# Using Deep Learning to Ensure the Safety of Patients Who Cannot Remain Still During Ultra-High Field MRI

Katherine Anna Blanter

A thesis submitted for the degree of Doctor of Philosophy

Cardiff University

September 2025

**Summary**

Ultra-high field (UHF) MRI may enable improved image resolution and contrast but has the potential to introduce tissue-heating-related patient safety concerns, approximated by specific absorption rate calculations (SAR). In an effort to prioritize patient safety, the full potential of UHF MRI is impeded, limiting clinical utility. This becomes more pressing when patients cannot remain still. Rigid motion during imaging alters the location of local SAR in an unpredictable way. Traditional safety models are not responsive to or reflective of these dynamics. This thesis investigates the use of deep learning to predict the effect of motion on SAR distributions during parallel radio frequency transmission (pTx) MRI at 7 Tesla.

The work begins by establishing the foundational physics of MRI and the mechanisms underlying radio frequency power deposition. The limitations of conventional SAR simulation methods are discussed. Deep learning is introduced as a method capable of mapping non-linear spatial relationships between field data and anatomy.

The first study chapter provides background to the pipeline development process. The objective here is to ensure reproducibility and caution against practices in deep-learning-based research which may hinder project reproducibility. Later chapters describe two deep learning models, Conditional Generative Adversarial Networks (cGANs) (Chapter 4) and U-Nets (Chapter 7), trained to estimate motion-induced changes in local SAR and SAR-related datatypes. Based on the initial cGAN implementations in Chapter 4, it is hypothesized that the U-Net would yield more accurate results. Several data formats and preprocessing strategies are assessed to determine which pipeline produces the most reliable estimates (Chapters 4 and 5 ). The work examines how neural network structure and pre- and post-processing decisions affect prediction accuracy. As a result, smoother datatype representations and simpler pipelines are expected to yield better estimations in subsequent investigations (Chapters 6 and 7). The overarching research objective was to design a proof-of-concept pipeline for deep-learning-driven SAR prediction during head movement during UHF MRI.

The cumulative results offer guidance for deep learning pipeline development and suggest that such models may support safer scanning of patients who move during imaging. The findings support the potential of deep learning to extend safe imaging capabilities, reduce conservative safety constraints, and improve scan efficiency without compromising patient safety.

# Acknowledgments

Like many PhDs, this one was also turbulent. I was hoping mine would be special, but it was as turbulent or maybe even more so than any other. I've at times regretted the undertaking but at other times had a blast.

Without further ado, I thank my first supervisor, Emre Kopanoglu, for his patience, brutal honesty, totally weird analogies, and dedication to the project and my well-being. I came to the field a complete novice, but somehow managed to go to some conferences and write this thesis and maybe publish a paper. That can't have been easy to guide.

I'd like to thank my second supervisor, Kevin Murphy, for his calm and kind understanding.

Thanks loads to Scot from CUBRIC's IT team for always taking the time to teach us everything ever about relevant hardware and software.

Thank you to the people at CUBRIC who, at least while we were all there, made the workplace a home: Alix, Elisa, Veronica, Ryan, Kajal, Malwina, James, and Phil.

Finally, I'd like to thank Isabella and Renzo who convinced me to power through, the folks at Tartu Raamatukogu for the ping-pong and much-needed distraction, and Liz, Martin, Yoana, and Karol, who, through their own PhD-related wins and misfortunes, always offered kindness, support, and encouragement.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# MRI Physics and Safety Foundations

## 1.1 Overview

Magnetic resonance imaging (MRI) field strengths range from sub-Tesla (ultra-low-field MRI) to above 4 Tesla, referred to as ultra-high field MRI (UHF MRI). Unlike X-ray or computed tomography (CT) imaging, MRI does not involve ionizing radiation. Instead, it uses magnetic fields and radiofrequency (RF) pulses to produce high-contrast and detailed images. This makes MRI a valuable diagnostic method and a relatively safe, noninvasive option for patients.

Although MRI is well established in clinical use, its full potential is yet to be reached. For this reason, many aspects of the technology, including hardware design, imaging sequences, and safety procedures, continue to be developed. To provide context for the work that follows, this chapter introduces the basic principles of MRI and explains how increased field strength affects signal-related elements such as the net magnetization vector (NMV), precessional frequency, and resonance. A brief summary of pulse sequence design and safety concerns is also included.

The second part of the chapter focuses on RF safety, including electromagnetic interactions and the potential for tissue heating, particularly in relation to UHF MRI. These subjects provide the necessary background for the material that follows, which discusses how deep learning can be used to estimate and reduce safety risks related to unintended subject motion in 7 Tesla (7T) parallel transmit

systems.

## 1.2 MRI Foundations

### 1.2.1 Net Magnitization Vector

Magnetic resonance (MR) imaging depends fundamentally on the interaction between the static polarizing magnetic field ($B_0$) and the intrinsic properties of MRI-active nuclei [11]. Superconducting magnets are the typical systems used in clinical practice because of their superior diagnostic utility and efficiency compared to alternative systems such as permanent and resistive magnets [12]. In superconducting systems, the static magnetic field $B_0$ is generated by electric currents passing through superconducting coils. The behavior of the resulting electromagnetic fields is governed by Maxwell's equations, which describe how electric and magnetic fields are generated and altered by charges and currents [13]. According to these equations, time-varying electric currents and changing electric fields produce magnetic fields, while a time-varying magnetic field induces a perpendicular electric field, as described by Faraday's Law of Induction. The strength and uniformity of $B_0$ depend on both the magnitude of the applied current and the design of the coil system.

Among the various MRI-active nuclei, hydrogen is the most widely used for its high natural abundance in biological tissues and its strong magnetic resonance signal [14]. Nuclei suitable for MRI must have a non-zero magnetic moment, which allows them to align with or against an external magnetic field. Hydrogen nuclei, which are spin-$\frac{1}{2}$ particles, can align either parallel (low-energy state) or antiparallel (high-energy state) to the $B_0$ field. In thermal equilibrium, slightly more hydrogen nuclei occupy the parallel state, resulting in a small imbalance. This imbalance produces a net magnetic moment, referred to as the net magnetization vector (NMV), which points in the direction of the external field. The NMV is the vector sum of the individual magnetic moments of all hydrogen nuclei within a given volume. As the strength of the magnetic field increases, the imbalance between nuclei aligned with and against the field becomes greater, resulting in a larger net magnetization vector (NMV) and improved signal-to-noise ratio (SNR) in the resulting MR images.

### 1.2.2 Precession

A higher magnetic field strength also increases precessional frequency, which describes the circular motion of a magnetic moment around the axis of the external magnetic field. This motion is neither a spin nor an orbital path, but rather is analogous to the wobbling motion of a spinning top as it slows down. Precessional frequency is described by the Larmor equation, which states that the frequency of precession ($\omega_0$) is directly proportional to the $B_0$ and the gyromagnetic ratio ($\gamma$):

$$\omega_0 = B_0\gamma \tag{1.2.1}$$

The gyromagnetic ratio is a constant specific to each nucleus. For hydrogen nuclei, it is approximately 42.57 MHz/T. Therefore, the precessional frequency of hydrogen is approximately 63.86 MHz at 1.5 T and 297.99 MHz at 7 T [15].

### 1.2.3 Resonance

Although the range appears wide, the precessional frequencies of hydrogen nuclei at both 1.5 T and 7 T fall within the radiofrequency (RF) range of the electromagnetic spectrum. For this reason, RF pulses, referred to as $B_1$ fields, are used to excite the nuclei. These fields are applied perpendicular to the $B_0$. The process of energy transfer is known as resonance and occurs when the frequency of the RF pulse matches the Larmor frequency of the nuclei. RF energy is delivered through a device called a transmit coil. The most common transmit coils are quadrature (or circular polarization) coils, which use two orthogonal coils operating 90° out of phase to generate a rotating $B_1$ field. These coils can transmit and receive signals, and when a single device performs both functions, it is referred to as a transceive coil [16]. There are also systems composed of transmit and receive coils which perform the transmission and reception functions separately. Common types of quadrature coils include the birdcage coil and the transverse electromagnetic (TEM) coil. TEM coils are especially well-suited for use at frequencies above 200 MHz, such as in ultra-high field MRI, due to their ability to provide more uniform RF excitation [17].

### 1.2.4 Radio Frequency Transmission

Resonance allows efficient energy transfer from the RF pulse to the targeted nuclei, rotating the NMV away from its alignment with the $B_0$. This rotation is quantified by the flip angle, which depends on the amplitude and duration of the RF pulse and is measured in degrees relative to $B_0$. A 90° flip, for example, moves the NMV entirely into the transverse plane. During this process, the individual magnetic moments of nuclei begin to precess in phase, resulting in a coherent transverse magnetization that can be detected to form the MR signal.

After RF excitation, the signal generated by this coherent transverse magnetization induces a voltage in a receiver coil, which contributes to the final MR image.

Following excitation, the MR signal decays and magnetization gradually returns to equilibrium through relaxation processes. T2 relaxation refers to the dephasing of transverse magnetization, while T1 relaxation describes the recovery of longitudinal magnetization. The rates of these processes depend on the chemical environment of the nuclei, for example, whether they are in water or fat [15].

### 1.2.5 Bloch Equations

The Bloch equations, from Felix Bloch, are a set of time-dependent differential equations that describe the behavior of the net nuclear magnetization vector ($\vec{M}$) in a magnetic resonance system [18]. They model how nuclear magnetic moments respond to external magnetic fields and relaxation processes, including both longitudinal (T1) and transverse (T2) relaxation [19].

Briefly, the cumulative angular momentum of millions of hydrogen nuclei can be represented with one vector, ($\vec{M}$), which rotates in space and is composed of transverse ($M_{x(t)}$ and $M_{y(t)}$) and longitudinal ($M_{z(t)}$) components. The simplest depiction of the interaction between M and B is of M precessing in a cone around B, with $M_{z(t)}$ as a constant, and $M_{x(t)}$ and $M_{y(t)}$ rotating in the z-plane at $\omega$. This causes a current in the receive coil.

These equations provide a model for the evolution of magnetization during and after RF excitation, forming the theoretical basis for MRI signal.

## 1.3 RF Transmission and Sequence Design

### 1.3.1 RF Pulses

RF transmission in MRI is typically delivered as short bursts, known as RF pulses, rather than continuous waves. These pulses, when combined with gradient magnetic fields, form structured pulse sequences. Pulse sequences vary in amplitude, duration, interval, and repetition to manipulate tissue contrast and spatial encoding.

Gradient fields are linear magnetic field variations applied in the orthogonal x, y, and z directions. They modify the $B_0$, thereby altering the precessional frequency of hydrogen nuclei along the gradient direction. This frequency variation enables spatial encoding through three mechanisms: slice selection, frequency encoding, and phase encoding [20].

The acquired signal is stored in k-space, a domain representing spatial frequency information rather than direct anatomical structure. A discrete Fourier transform is applied to reconstruct the image from k-space data [15]. A Fourier transform is a computation that breaks down a signal into its basic frequency components. Since, importantly, k-space is organized based on frequency and phase information, not spatial coordinates, the Fourier transform is used to convert spatial frequency data from k-space into an image.

RF pulse sequences are generally categorized into those that rephase dephase spins using RF pulses (e.g., spin echo) and those that use gradient reversal (e.g., gradient echo) [21]. These sequences are embedded into imaging protocols, with the timing parameters called repetition time (TR) and echo time (TE) controlling T1 and T2 weighting, respectively.

TR, the time between successive RF excitations, influences the extent of T1 relaxation at the time of signal readout. TE, the interval between the RF pulse and signal peak, affects the degree of T2 signal decay captured in the image.

A range of RF coil configurations are used to optimize signal detection and transmission. When paired with specific pulse sequences, coil design can improve SNR, reduce acquisition time (e.g., with parallel imaging, discussed imminently), and enhance the ability to image specific tissues, functions, or pathologies.

### 1.3.2 Basic RF Pulse Sequences and SAR

The most basic and clinically ubiquitous RF pulse sequences are the spin echo (SE) and gradient echo (GRE) sequences. SE begins with a 90° excitation pulse and is followed by one or more 180° rephasing pulses to generate an echo. This versatile sequence offers robust image quality and sensitivity to pathology but requires relatively long acquisition times, as only one line of k-space is acquired per TR. For a typical 256×256 voxel image, 256 phase encoding steps, and thus 256 TRs, are required.

In contrast, GRE imaging uses just one RF pulse which is usually less than 90° to excite the spins, followed by dephasing and rephasing with gradient field reversals. GRE does not use an RF refocusing pulse. It should be noted that GRE is a general term for the method and can be used in multiple ways that are tailored to the desired end goal. Compared to SE, GRE is faster because it uses shorter TR. While it is sensitive to $B_0$ inhomogeneities, the improved technology in modern scanners compensates GRE's methodological deficits [22].

More advanced pulse sequences are designed to reduce scan time, improve SNR, or better visualize specific pathologies. One example is single-shot fast spin echo (SS-FSE), which acquires nearly all of k-space in a single TR using rapid successive echoes. This is often combined with a half-Fourier technique that exploits k-space conjugate symmetry by acquiring slightly more than half of k-space, and leaving the rest to be reconstructed algorithmically [21]. However, this method reduces the SNR in the reconstructed portion by a factor of $1/\sqrt{2}$ [23]. This is an example of the give-and-take often found in MRI methods development: there is constantly a tradeoff between image quality, imaging time, cost, patient comfort, and patient safety.

SS-FSE requires multiple successive 180° RF pulses, which increases RF power deposition. Since RF energy is absorbed by tissues, this leads to heating and is quantified using the specific absorption rate (SAR), measured in watts per kilogram (W/kg). In other words, SAR is a metric of how much power from a deposited RF field is absorbed by a specified amount of tissue. SAR depends on tissue conductivity, subject size, RF pulse duty cycle, and scales with the square of both the RF frequency and the flip angle (e.g., a 180° pulse delivers four times the energy of a 90° pulse) [24].

### SAR Averaging and Limits

SAR is physically defined by:

$$\text{SAR} = \frac{\sigma |E|^2}{\rho} \tag{1.3.1}$$

where $\sigma$ denotes the electrical conductivity of the tissue (S/m), $|E|$ is the root-mean-square electric field magnitude (V/m), and $\rho$ is the mass density of the tissue (kg/m$^3$).

Different SAR types are specified by regulatory bodies, namely the International Electrotechnical Commission (IEC) and Food and Drug Administration (FDA). Those SAR types are delineated by means of averaging types [25]: whole body, partial body, head, extremity, local, and short-term. The SAR in W/Kg which is safe to deposit into the patient mass differs based on those six averaging types as well as whether volume or local coils are used to transmit the RF and the mode of scanner operation. There are three operating modes which are informed by potential subject risk: normal, first level controlled and second level controlled. While normal mode is for routine level (safe, $B_0 \leq$ 3.0 T), first level describes imaging where the patient may experience some discomfort (potential for some risk, $B_0 \leq$ 8.0 T), and second level is for experimental level functions (potential for significant risk). Only normal and first mode have regulatory SAR limits specified by the IEC, while second level limits are left to be determined by institutional ethics committees. The SAR limits are as follows:

1. Whole body: SAR averaged over a total patient body over 6 minutes of RF transmission while using a body transmit coil
   (a) Volume transmit coils
      i. normal: 2 W/kg
      ii. first level: 4 W/kg
2. Partial body: SAR averaged over the part of the patient which is exposed to the RF over any 6 minutes. This is scaled by a ratio of RF-affected anatomical region to whole patient mass.
   (a) Volume transmit coils
      i. normal: 2-10$^a$ W/kg, where $a$ is [10 - 8 x patient mass ratio]

ii. first level: 4-$10^b$ W/kg, where $b$ is [10 - 6 x patient mass ratio]

3. Head: SAR averaged over the head areas over any 6 minutes.
   (a) Volume transmit coils
      i. normal and first level: 3.2 W/kg
   (b) Local transmit coils
      i. normal: 10 W/kg
      ii. first level: 20 W/kg
4. Extremity: SAR averaged over an extremity over any 6 minutes.
   (a) Local transmit coils
      i. 20 W/kg
      ii. 40 W/kg
5. Local: SAR averaged over 10g of tissue over any 6 minute period.
6. Short-term: SAR averaged over any 10 seconds during the scan.

The IEC also set local and core temperature limits at 39 °C and 40 °C for normal and first level modes. At normal mode, the temperature may not rise by more than 0.5 °C, while at first level this limit increases to 1 °C [25].

**Thermal Effects**

The thermal effects of an applied and changing RF field arise from the coinciding changing electrical field (E-field) component, as per Maxwell's Laws [13]. The E-field causes both resistive and dielectric heating, the former of which is more responsible for injury during scanning. In resistive heating, existing ions in tissue produce a current from being accelerated by the changing E-field. Upon collision with each other, they produce thermal heat. Dielectric heating occurs from the direction of polar molecules in tissue aligning and changing in response to the changing E-field [26]. This mechanism also leads to collision and subsequent kinetic energy production (ie. heat) [27]. Since to date there is no practical method to measure E-fields directly during MRI scanning, researchers and scanner manufacturers use SAR as a proxy measure [26]. Though SAR is a surrogate measure for tissue heating, it is accessible and practical to calculate. For this

reason, it is standard practice to adhere to SAR limits when considering heating-related MRI safety.

At higher field strengths (typically 3 T and above), SAR limitations become more restrictive, requiring adjustments to pulse sequences that may reduce image quality or SNR to ensure patient safety.

Advanced and energy-intensive RF pulse sequences, especially at high field strengths, raise concerns about tissue heating, making it necessary to model and monitor thermal effects using models like the Pennes Bioheat Equation.

### 1.3.3 Pennes Bioheat Equation

The Pennes Bioheat Equation [28] is a mathematical model used to predict temperature changes in biological tissues. It was first introduced by C. S. Pennes in 1948 and has since become a tool in bioheat transfer modeling and thermal therapy research.

The model incorporates several mechanisms of heat transfer in biological systems, including thermal conduction, metabolic heat generation, and heat exchange via blood perfusion.

Like SAR measurement, the Pennes Bioheat Equation can be used to estimate temperature during MR imaging, but it is not entirely reliable because it excludes subjects with atypical perfusion since it assumes constant perfusion rates [29].

For example, a simplified estimate of tissue heating can be obtained by assuming no perfusion. Under these assumptions, the temperature rise $\Delta T$ is given by

$$\Delta T = \frac{\mathrm{SAR}\, t}{C}, \tag{1.3.2}$$

where SAR is the specific absorption rate in W/kg, $t$ is the exposure duration in seconds, and $C$ is the specific heat capacity of brain tissue.

For brain tissue, a commonly used value is

$$C = 3600 \text{ J/°C/kg} \tag{1.3.3}$$

In MRI safety standards (IEC 60601-2-33, normal operating mode) [25], the local

SAR limit for the head is

$$\mathrm{SAR}_{\mathrm{local}} = 10 \text{ W/kg}, \tag{1.3.4}$$

averaged over 10 g of tissue and over a 6-minute interval.

For an exposure duration of

$$t = 6 \text{ min} = 360 \text{ s}, \tag{1.3.5}$$

the resulting temperature rise is

$$\Delta T = \frac{(10 \text{ J/(s kg)})(360 \text{ s})}{3600 \text{ J/(kg °C)}} \tag{1.3.6}$$

$$= \frac{3600}{3600} \tag{1.3.7}$$

$$= 1°\mathrm{C}. \tag{1.3.8}$$

Without perfusion, a local SAR of 10 W/kg applied for 6 minutes would produce an estimated temperature rise of about 1°C in the pertaining tissue.

## 1.4 UHF MRI

### 1.4.1 The Utility of UHF MRI

Despite the associated safety and image-quality related issues described ahead, UHF MRI is valued because it enables the acquisition of more detailed images of tissue. The increased $B_0$ raises the SNR owing to greater net magnetization (ie. more available signal) [30]. In addition to increased SNR, UHF MRI improves contrast-to-noise ratio (CNR) which is useful when researchers and clinicians need to image small structures or detect subtle differences between tissue types [31].

CNR is the difference in signal intensity between adjacent tissues relative to background noise. While SNR affects overall image clarity, CNR determines whether anatomical boundaries or pathological structures can be distinguished in the resulting image [21]. UHF strengths improve CNR by increasing the separation in tissue relaxation properties and susceptibility effects, both of which affect signal behavior [32]. In ultra-high field MRI, this results in clearer delineation of gray and white matter, better visualization of subcortical structures, and improved detection of lesions or edema. CNR enhancement has utility in fMRI, T2*-weighted

imaging, and susceptibility-weighted imaging, where small differences in local tissue composition are relevant to function and pathology.

These benefits are relevant to neuroimaging, where UHF MRI reveals features that are difficult to detect at lower field strengths. UHF MRI is applicable to functional MRI and spectroscopy, both of which benefit from the higher spectral resolution and increased sensitivity provided by stronger fields [32].

### 1.4.2 RF Transmission at UHF

RF transmission during UHF MRI is more nuanced than at lower field strengths due to the nature of RF fields at higher Larmor frequencies. At UHF (e.g., 7 T), the RF wavelengths used for excitation are significantly shortened, often approaching or even falling below the dimensions of the human body and the MRI bore [33]. This leads to non-uniform distributions of the transmit and receive RF magnetic fields ($B_1^+$ and $B_1^-$), which degrade both image quality and safety margins.

Despite the theoretical increase in SNR at higher $B_0$, the inhomogeneity of the $B_1^+$ field causes spatial variations in flip angles, leading to spatially inconsistent image contrast and even total signal dropout in some regions. These effects, known as dielectric artifacts, manifest as bright and dark non-anatomical regions in the MR image and are caused by constructive and destructive interference patterns within the body. These patterns result from interactions between the RF electric field and the body's dielectric properties [34].

Furthermore, these interactions can exacerbate $B_1$ field inhomogeneity and lead to localized RF energy deposition, resulting in higher SAR and potential tissue heating [31].

### 1.4.3 Parallel RF Transmission

To address the image quality degradation and safety challenges inherent to UHF MRI, parallel RF transmission (pTx) was developed. In pTx, multiple transmit channels are used, each driven by its own RF amplifier and capable of independent amplitude and phase modulation with microsecond-level temporal resolution. Each channel contributes a tailored waveform, and the combined field is designed to optimize flip angle homogeneity or decrease SAR-related safety concerns [35].

### Static Parallel RF Transmission

The simplest implementation of pTx is static pTx, where the transmit settings are pre-calculated and fixed throughout the scan. While static pTx can improve $B_1^+$ uniformity (a process known as RF shimming), the complex superposition of electric fields from different channels can introduce localized SAR hotspots. This makes localized SAR a more pressing concern in UHF MRI than global or eigenvalue SAR, the latter referring to the maximum possible local SAR from any linear combination of transmit channels.

In practice, achieving uniform $B_1^+$ often requires trade-offs with SAR safety. Accordingly, many SAR-aware pTx pulse design strategies have been proposed to balance imaging performance with patient safety [35].

### Dynamic pTx

Dynamic pTx enables spatiotemporal control of RF excitation, allowing dynamic modulation of flip angles across space and time. This flexibility helps mitigate $B_1^+$ inhomogeneity at ultra-high field strengths. Modeling the full nonlinear transverse magnetization response under such conditions is computationally intensive. To simplify the problem, the small tip angle (STA) approximation is frequently used, which linearizes the Bloch equations under the assumption of small excitation flip angles. Under this approximation, the transverse magnetization becomes linearly related to the applied RF excitation, in contrast to the generally nonlinear behavior described by the full Bloch equations. This linearization significantly reduces the computational burden of RF pulse design in dynamic pTx.

Unlike conventional static pTx or single-channel excitation, dynamic pTx fills excitation k-space (the Fourier space of the desired excitation pattern) using tailored trajectories. One common dynamic pTx strategy is the spoke pulse method. In this technique, each spoke corresponds to a point in $k_x$–$k_y$ excitation space and represents a slice-selective RF pulse applied with spatially varying amplitudes and phases across channels. These pulses are designed to account for local $B_1^+$ inhomogeneities [35].

Spoke pulse sequences typically use between 1 and 7 spokes, depending on the size and complexity of the anatomical region. For instance, three spokes may suffice for brain imaging, while torso imaging might require up to seven to ensure uniform flip angles. Increasing the number of spokes improves uniformity but

increases RF pulse duration and SAR [35].

Because spoke pulses under-sample excitation k-space, they allow for more efficient transmission and reduced sensitivity to relaxation and off-resonance effects. The accuracy and order of spoke placement in k-space are determinant for achieving effective excitation patterns, and optimizing these parameters is essential to balance homogeneity, SAR, and sequence timing [36].

The interaction between electromagnetic fields and biological tissue, especially at high field strengths, is relevant to developing SAR-aware pulse designs. The following sections summarize computational methods used to model and mitigate SAR in pTx MRI.

## 1.5 SAR and pTx Design

### SAR Calculation

One caveat in local SAR reduction is that SAR arises from the E-field of the RF electromagnetic field, which cannot be directly measured during MRI. As a result, local SAR estimation is restricted to pre-scan calculations using numerical simulations [37]. These simulations apply Maxwell's Equations to either patient-specific anatomical models (which are time-consuming to obtain) [38] [39] [37] or to high-resolution generic body models within electromagnetic simulation software (as discussed in section Safety Modeling) [40].

From these simulations, the E-field distributions are used to compute a set of matrices known as Q-matrices [41, 42]. Each Q-matrix describes how a linear combination of transmit channels contributes to SAR in a specific voxel or region. Q-matrices for a specified tissue area (global, 10g, etc) can be calculated by

$$\mathbf{Q}_{m,n}(\mathbf{r}) = \frac{1}{2\rho(\mathbf{r})}\left[\mathbf{J}^H_{x,n}(\mathbf{r})\ \mathbf{E}_{x,m}(\mathbf{r}) + \mathbf{J}^H_{y,n}(\mathbf{r})\mathbf{E}_{y,m}(\mathbf{r}) + \mathbf{J}^H_{z,n}(\mathbf{r})\mathbf{E}_{z,m}(\mathbf{r})\right] \quad (1.5.1)$$

where $\rho(\mathbf{r})$ is the tissue conductivity, $\mathbf{E}_x, \mathbf{E}_y, \mathbf{E}_z$ and $\mathbf{J}_x, \mathbf{J}_y, \mathbf{J}_z$ are the 3D E-field and current density distributions, $m$ and $n$ represent the pTx coil's channel indices, and $H$ is the Hermitian conjugate. Because SAR must be evaluated both within and beyond the region of interest (ROI), this leads to a large number of Q-matrices across the body.

To make optimization tractable, only a subset of these matrices, called Virtual Observation Points (VOPs), are used. VOPs compress the full set of Q-matrices by preserving only those that bound the worst-case SAR outcomes within a specified error margin. These VOPs are then incorporated into the RF pulse design process as constraints or objectives. While limiting SAR is commonly a constraint, optimization may also target both SAR reduction and improved $B_1^+$ homogeneity in specific ROIs [35].

### SAR, pTx, and Motion

Motion is problematic during MRI scanning. While issues regarding motion at lower field strengths are related to hampered image quality, at UHF imaging with pTx, SAR safety becomes a major concern. Patient motion may arise due to long scan times, discomfort, pediatric non-compliance, or neurological disorders that impair motor control. Motion is typically classified into six degrees of freedom: three translations (left–right, anterior–posterior, superior–inferior) and three rotations (pitch, roll, yaw) [40].

Because pTx depends on accurate spatial relationships between the patient and coil elements, any movement invalidates the precalculated RF channel weights and $B_1^+$ field maps. $B_1^+$ maps are used to characterize the magnitude and phase response of each transmit channel across space, enabling personalized pTx pulse calculation and SAR estimation. However, acquiring accurate $B_1^+$ maps both lengthens total scan time and requires cooperation from the participant.

Overall, motion leads to unpredictable flip angle distributions, degraded image quality, and potential localized SAR increases in unpredictable locations. For instance, decreased distance between the head and coil can cause unexpected hotspots, while increased distance requires more RF power to achieve adequate excitation [33]. Even small deviations from planned positioning can cause local SAR to increase more than fourfold [43].

Increased SAR may occur during scanning for a few different reasons: the body's contact with the bore, the body's contact with a conductive object, and skin-to-skin contact. This is all because the human body is conductive, and when an RF field is applied, it causes eddy currents, and therefore Jule heating (the conversion of electrical energy into heat when an electric current flows through a resistive material). When the eddy currents concentrate at contact points, they induce local heating which may be further exacerbated for a resonant conductive loop.

This does not explain burn injuries occurring from the body's contact with the scanner bore, since in that case there is no loop [44]. In the case of body-to-bore contact, heating is likely caused by the strong electric field at the capacitors of the RF transmit coil [45].

During multi-spoke excitations using pTx, motion has been shown to increase local SAR by up to 3.1-fold, and up to 2.1-fold during quadrature excitation. Importantly, local SAR and whole-head SAR do not change in lockstep with motion, meaning that monitoring only one may miss important safety information. On the other hand, eigenvalue SAR provides a theoretical upper bound on SAR but can be overly conservative, potentially sacrificing image quality without actually providing additional safety benefits [40].

Moreover, SAR safety limits are typically based on simulations using static body models positioned within the RF coil. During actual scanning, the system does not account for patient motion, meaning the safety model may not reflect the subject's real-time position [46]. Coil manufacturers can either implement a multi-position safety model that accounts for worst-case scenarios, prioritizing safety at the expense of image quality, or use a single-position model, which may offer better image quality but fails to ensure safety if the subject deviates from the modeled pose. The former sacrifices imaging performance to guarantee safety, while the latter risks overheating if the subject falls outside the model's assumptions. For this reason, an optimal approach would incorporate a motion tracking mechanism paired with a tool that models the impact of motion on SAR in real time and communicates this information to the scanner to enable safety-preserving adjustments.

To begin to address this, recent work has introduced near-real-time pTx pulse design strategies that aim to meet safety and performance constraints within sub-second timescales [47]. These methods incorporate electromagnetic field simulations, VOPs, and precomputed libraries to iteratively optimize excitation pulses in a motion-aware framework.

### Near-Real-Time pTx

Safety and image quality issues associated with involuntary participant motion at UHF MRI have led to the development of near-real-time (sub-second) pTx pulse design algorithms. In work by Kopanoglu et al. (2018 and 2024) [47] [48], a computational framework is introduced to enable SAR-aware and system-

constrained pTx design in under one second.

The method jointly optimizes slice selection and excitation spokes using a set of precomputed data inputs, including electromagnetic field simulations of $B_1^+$ and electric fields, Q-matrices for SAR estimation, and coil-specific spatial sensitivity profiles. A precomputed library of flip-angle slice-selective pulse envelopes is also incorporated to reduce real-time computation.

During the design process, spokes in excitation k-space are iteratively selected and assigned RF weights across transmit channels. Each candidate spoke is evaluated for excitation fidelity and compliance with system constraints (e.g., gradient limits, peak voltage) and safety constraints (e.g., local SAR). The algorithm continues iterating until a user-defined number of spokes is selected with minimal excitation error.

This approach is a step toward making pTx pulse design clinically viable while accommodating subject motion during planning or calibration.

Motion is a significant and unresolved hurdle for pTx at UHF strengths. Because pTx pulse designs are highly sensitive to the spatial relationship between the body and transmit coils, even small involuntary displacements can invalidate precalculated RF waveforms and introduce unacceptable local SAR risks. Simulating every possible motion state using full EM solvers is computationally infeasible, especially when patient safety must be ensured in real time. This motivates the use of machine learning models that can generalize across motion states by learning to predict the electromagnetic consequences of displacement. In the following chapters, deep learning is explored as a strategy to model the effects of head motion on SAR-related distributions.

## 1.6 Safety Modeling

### 1.6.1 Body Models

As previously discussed, ensuring SAR safety during UHF MRI with pTx involves electromagnetic simulations using realistic human body models before scanning. These models used in this thesis are designed to represent a broad spectrum of typical human anatomies and are part of the Virtual Population (ViP), developed by the IT'IS Foundation and implemented in the Sim4Life simulation platform [49].

The Sim4Life body model library includes anatomically detailed representations of infants, children, adolescents, adults of various BMIs (including obese individuals with BMI $> 30\,\mathrm{kg/m^2}$), a pregnant woman, geriatric individuals, and more [50]. This diversity is essential, as energy absorption and SAR distribution vary significantly depending on body size, age, sex, tissue composition, and fat content.

Each model includes over 300 tissue types, segmented with high anatomical resolution, and is associated with frequency-dependent electromagnetic properties, including conductivity, permittivity, mass density, and in some cases anisotropy. These parameters are essential for accurate SAR estimation and temperature prediction through bioheat modeling. Bioheat modeling involves solving biothermal transport equations, such as the Pennes Bioheat Equation, to quantify spatial and temporal temperature distributions in tissue resulting from RF-induced power deposition, accounting for thermal conduction, perfusion, and natural body heat.

By enabling simulations of local SAR and thermal effects under varied anatomies, the use of detailed body models aids subject safety during MRI scanning.

### 1.6.2 Simulation

Electromagnetic simulations can be performed using many methods including the Finite Difference Time Domain (FDTD) method, method of moments (MoM), and finite element method (FEM).

Both the FEM and FDTD methods are solved in the volumetric domain. While FDTD is solved by means of differential equations, FEM is solved through variational form. The material types they are applied to are typically nonlinear and anisotropic [51].

FDTD applies finite differences to Maxwell's equations. It supports a wide range of frequencies and can be applied to nonlinear materials. On the other hand, it struggles with computational efficiency when applied to complex geometries because it applies a high resolution mesh in unnecessary locations. This issue can be solved by varying the mesh size. FDTD can also be susceptible to numerical dispersion error because of approximation assumptions it imposes on Maxwell's equations. In other words, the finite nature of the method yields compounded inaccuracies [51].

FEM on the other hand is more suitable to complex geometries and is used to solve partial differential equations. Because it is applied in the frequency domain, it does not succumb to dispersion error. Overall, FEM is more difficult to use because of the necessity to generate accurate volumetric meshes. Moreover, it does not perform well on homogeneous materials [51].

In contrast, the MoM is solved by means of integral equations and is discretized with surface currents. Its utility is for linear and homogeneous materials. While it is simpler to administer and is more computationally efficient than the other two methods, it is unsuitable for body model simulations since bodies are not homogenous and linear [51].

The work in this thesis uses data simulated in Sim4Life [52] with the FDTD method. FDTD is suitable because it is relatively computationally efficient while still being suitable for complex geometries (ie. human and other animal bodies).

**Simulation parameters particular to the work presented in this thesis**

Each of the five body models used in the following chapters were simulated according to the protocol described here, as well as those applied in Refs. [8, 43]. The configuration of the eight numbered channels used throughout this thesis is illustrated in Figure 1.6.1. Numbering the channels provides a clear spatial reference and facilitates consistent interpretation across the figures and analysis.

For the simulations, the transmit loops are defined as perfect electrically conducting (PEC) structures with a height of 220 mm. These loops are arranged equidistantly in a 360° ring configuration, a setup chosen to enhance generalizability to real-world coil designs. Resistance (Ohm, Ω) and tuning and matching capacitors (pico Farads, pF) are specified based on the individual body model used. Resistance is how much the flow of charge is opposed, while capacitance describes a capacitor's ability to store electric charge in terms of the ratio of charge to voltage. These electrical properties influence the reflection coefficients (in decibels, dB) at the 7 T Larmor frequency of 298 MHz, which indicate the proportion of the RF signal reflected due to mismatched resistance.

To simulate varying degrees of participant motion, the pTx coil array is shifted relative to the stationary body model, rather than moving the body itself. This approach avoids artifacts caused by voxel redefinition, like partial volume effects, which can occur when the body model's pose is altered. Partial volume effects

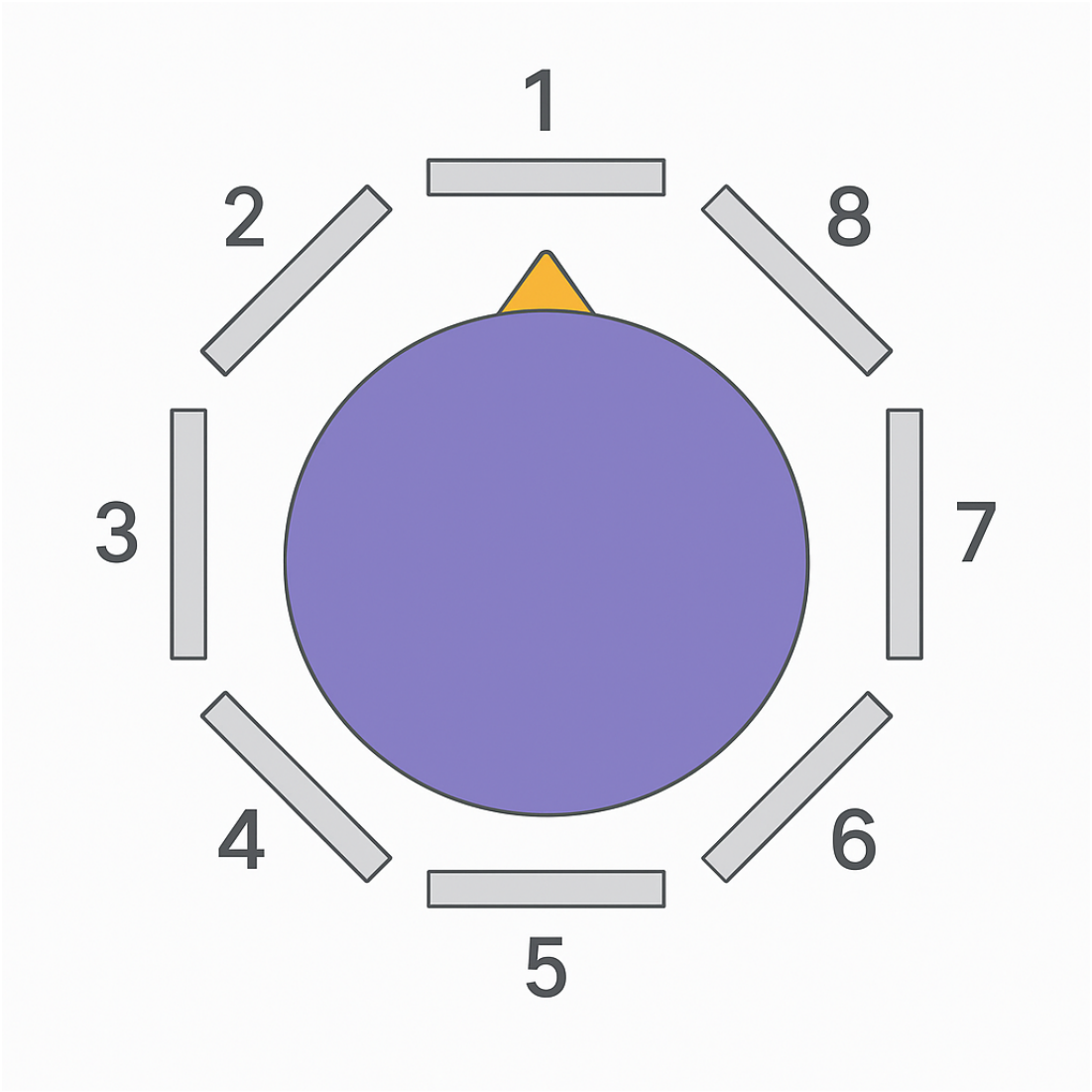


Figure 1.6.1: Numbered 8-channel pTx coil configuration used throughout this thesis. The triangle indicates the position of the nose, providing orientation for the spatial layout of the transmit channels.

refer to small changes in the boundaries between tissue types that arise when voxel dimensions do not align consistently during simulated movement.

Each body model is imported into the 3D simulation space and centered within the eight-channel coil ring. Key parameters influencing simulation and processing time, as well as output data size, include grid resolution, padding, and the number of simulated periods. Although Sim4Life supports a voxel resolution as fine as 0.5 mm, such high resolution is computationally expensive. Therefore, voxel sizes are varied depending on the anatomical region, with finer resolution reserved for areas where accuracy is needed most.

In this study, the pTx coils require the highest resolution. When simulating coil movement, coarse voxelization can cause artificial discontinuities in coil geometry, potentially compromising simulation accuracy. The voxelized simulation domain (bounding box) extends from the top of the head to just below the shoulders. This range ensures accurate modeling of RF energy absorption, as prior studies suggest that SAR hotspots can occur in the shoulder region during 8-channel pTx neuroimaging [53]. Additionally, including the shoulders improves simulation accuracy [54].

Finally, because the interaction between the $B_1^+$ field's E-field and the body's dielectric tissues contributes significantly to local SAR, accurate E-field simulation is essential for reliable safety assessments.

## 1.7 Summary and Outlook

Overall, this chapter has introduced the electromagnetic basics of MRI with a focus on issues introduced by pTx at UHF strengths. Motion-induced variability, $B_1^+$ inhomogeneity, and SAR hotspots all make it difficult to design safe and effective imaging protocols in clinical practice and research settings. These hurdles motivate the purpose of this thesis which develops a proof-of-concept deep learning pipeline designed to mitigate the effects of unintended motion on SAR-relevant electromagnetic parameters. The next chapter introduces the machine learning framework and related literature before presenting the deep learning models developed, applied, and evaluated in the remainder of this thesis.

# Chapter 2

# Deep Learning for SAR Prediction in MRI

## 2.1 Brief Deep Learning Overview

Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL) are related but distinct concepts within computer science. AI is the broadest term, ML is a subset of AI based on the idea that machines can learn from data without being explicitly programmed, and DL is a subset of ML that employs neural networks (layered architectures composed of interconnected nodes that process input data and adjust internal parameters to learn complex patterns) with many deep layers to model patterns in large datasets which humans would otherwise be hard-pressed to discern. DL performs notably well in image recognition and natural language processing, when given an abundance of data [55–57].

In basic terms, deep learning networks typically consist of an input layer, multiple hidden layers, and an output layer, through which information is propagated and transformed. Each connection between nodes is assigned a weight, and nodes apply an activation function to determine whether signals should continue forward. The concept is loosely inspired by neurophysiology and the pertaining action potential responsible for determining whether a signal is propagated. DL models are trained using large datasets, where the network iteratively adjusts its weights based on discrepancies between predicted and actual values [55]. This process is supported by backpropagation and optimization algorithms like stochastic gradient descent [58].

### 2.1.1 Training, Validation, and Testing

When designing a DL pipeline, the dataset is typically split into three parts: training, validation, and testing. The training data is what the model actually learns from. Validation data is separate from the training set and is used during training to evaluate how well the model is generalizing. Testing data is only used at the very end, after training is complete. It gives an unbiased measure of how the model performs on completely unseen data. This separation helps prevent overfitting, where the model does well on training data but fails to generalize to new inputs [55, 56, 59].

### 2.1.2 Parameters

Parameters are values within the network that are learned from data during training. They influence the model's predictions. These parameters are the weights and biases associated with the connections between neurons across the network's layers. Weights and biases determine how input data is transformed as it passes through the model. Each connection between neurons is assigned a weight, which scales the input and determines the relevance of that connection. The weights are adjusted during training to minimize prediction error. In other words, weights control how strong an influence a change on the input will have on the output. In addition to weights, each neuron also includes bias, which acts as an offset added to the weighted value. Bias values represent how close the predictions are from the intended output value. Weights and biases are continuously updated during training to improve network performance. During training, the goal is to minimize the loss function. The final set of parameters encodes what the model has learned [55, 56, 59].

### 2.1.3 Loss Function

A loss function is used to measure how far off a model's predictions are from the actual target values. It provides a quantitative measure of the model's prediction error, thereby guiding the optimization process during training. The lower the loss, the closer the model's predictions are to the correct answers. During training, the model uses this information to adjust its internal settings so that it performs better with each iteration.

### 2.1.4 Backpropagation

Backpropagation allows neural networks to adjust their parameters to reduce prediction error over iterations. It operates by comparing the model's predicted output to a known ground truth, calculating a loss that represents the total error. This loss is then propagated backward through the network, from the output layer to the input layer, using the chain rule to compute partial derivatives of the loss with respect to each individual weight. In doing so, the network identifies which parameters contributed most to the error. The weights are then updated incrementally using these gradients to reduce the loss during subsequent iterations. Each iteration, or epoch, brings the predicted output closer to the true target by reinforcing the weight configurations that produce more accurate outcomes [55, 56, 59].

### 2.1.5 Gradient Descent

Gradient descent is an optimization method used to minimize a loss function by iteratively adjusting the parameters of a model in the direction of steepest descent, as defined by the negative gradient. A valid visualization of gradient descent typically involves a parabola on a grid, where the lowest possible point in the valley is marked. The principle goal is to reach that lowest possible value in an optimal way. In less basic terms, the goal is to evaluate how a small change in each parameter affects the overall loss, and then to update each parameter in the opposite direction of this gradient to reduce error. The loss function evaluates the difference between predicted outputs and the true labels, and the gradient is computed with respect to each weight in the network. This process is repeated across multiple iterations, refining the network's weights to reduce prediction error. Traditional or batch gradient descent computes the gradient using the entire training dataset at each step, ensuring an accurate but computationally expensive update. This method produces a stable descent toward a minimum but becomes inefficient with large datasets because each update requires processing all of the examples before proceeding [55, 56, 59].

### 2.1.6 Stochastic Gradient Descent

Stochastic gradient descent (SGD) [60] is a more efficient method which is often employed in DL architectures. At each step, rather than computing the gradient of the loss over the entire training dataset (as in batch gradient descent), SGD

estimates the gradient using only a single randomly selected data point or a small subset (mini-batch). SGD often allows the model to escape shallow local minima. After each forward and backward pass, the network's weights are nudged according to the computed gradients and a user-defined learning rate, which determines the size of the step. Over many iterations, SGD progressively refines the model parameters, moving toward a configuration that minimizes the average loss across the training set [55, 56, 59].

### 2.1.7 Hyperparameters

Parameters and hyperparameters are both properties that influence how a neural network performs, but they operate at different levels. Parameters, eg. weights and biases, are part of the internal structure of the model. These values are learned directly from the training data through iterative updates during optimization. In contrast, hyperparameters are settings that are defined before training begins. They control how the learning process unfolds but are not always updated during training (though methods to update learning rates, for example, are available). Examples of hyperparameters include the learning rate, the number of layers, the size of each layer, the batch size, and how many times the model sees the data during training (epochs). Because hyperparameters are set manually, choosing effective values often involves trial and error (see Chapter 5). Selecting the most effective combinations of hyperparameters influences how well the network learns and how well it performs on new data [55, 56, 59]. Optimization methods ensure prediction accuracy by controlling how, how much, and when the parameters are changed in the model from updates informed by the loss function [55, 56, 59].

#### Learning Rate

The learning rate controls how steep each step is in gradient descent by determining how much the model's parameters are adjusted in response to the calculated error. A higher learning rate causes the model to take larger steps through the parameter space, which can speed up training but also increases the risk of missing the point where the error is minimized. A lower learning rate results in smaller, more gradual updates, which can help the model settle into a lower error region more precisely but may require more time and iterations to reach a useful solution. The learning rate does not change on its own during training unless a schedule (called a learning rate scheduler) or adaptive method is used [55, 56, 59].

### Mini Batch Size

The mini batch size determines how many individual steps in gradient descent are combined together into 'batches' and processed simultaneously for computational efficiency purposes. Instead of updating the model's parameters after every single example, or waiting until the entire dataset has been seen, the training data is divided into smaller batches. Each batch produces an estimate of the gradient based on its subset of data, which is then used to update the model. When the model updates its parameters using smaller mini batches, the estimate of the gradient (the direction and size of the update) is based on fewer data points. Because of this, the estimate is less stable and can vary more from step to step. This variation is noise in the gradient. The random fluctuations caused by small mini batches can push the model away from narrow or shallow areas in the parameter space and help it find better solutions. In other words, it keeps the model from getting stuck too early in spots that are not useful. However, because these updates oscillate more, it may take longer for the model to settle into a stable, low-error configuration (ie. cause a longer training time). That is what is meant by slower convergence. Larger mini batches produce smoother, more stable updates but require more memory and can reduce the number of total updates per epoch. The choice of mini batch size can influence how quickly the model trains and how well it generalizes to unseen data [55, 56, 59].

### Epochs

Since networks are expected to learn patterns and structure from the training data, it is not sufficient to pass the data through the network only once. During a single pass, the model typically makes large errors and has not yet had enough exposure to the data to make useful adjustments to its parameters. The number of epochs refers to how many times the entire dataset is presented to the network during training. With each epoch, the model updates its weights based on the gradients computed from the training examples, gradually refining its mapping and reducing the overall loss. Multiple passes through the data reinforce data information, correct earlier mistakes, and converge toward a more accurate mapping between inputs and outputs. On the other hand, using too many epochs can lead the model to overfit, meaning it performs well on the training data but poorly on new, unseen data. The choice of how many epochs to use is typically determined by monitoring performance on a separate validation set and stopping training once improvement plateaus or begins to reverse [55, 56, 59]. To develop

the content of this thesis, TensorBoard [61] was used to track training and validation loss over epochs.

### Activation Functions

The functions that guide how the network transforms input data as it moves through the layers are called activation functions. These are applied to the output of each neuron after the input values have been combined using weights and biases. Activation functions introduce nonlinearity, allowing the network to model relationships that would not be captured by linear operations alone. Without them, a deep network would behave like a single-layer linear model and fail to learn more nuanced patterns. The most commonly used activation function is the rectified linear unit, or ReLU. It outputs zero for any negative input and returns the input itself if the value is positive. This simple behavior allows the model to introduce non-linearity while preserving positive signal flow, which helps it learn patterns where the effect of one input may depend on the presence or strength of another. ReLU is also efficient to compute and helps maintain useful gradient values during training. Other activation functions, such as sigmoid (binary) or softmax (multiclass), are used in different parts of the network [55,56,59]. Further explanation of ReLu can be found in Chapter 5.

### Hidden Layers

The number of hidden layers depends on the intricacy of the network. Each hidden layer is made up of hidden units, also called neurons, and the number of these units determines how densely connected the layers are. More hidden layers allow the network to perform a greater number of sequential transformations on the input data, allowing it to detect and combine increasingly detailed patterns. The appropriate number of hidden layers depends on how much structure the model needs to uncover in the data. For example, in an image classification task, a shallow network might be sufficient to recognize simple features like edges or shapes, but deeper networks are often needed to detect and combine more detailed patterns. In the case of an image containing horses, pigs, grass, and a barn, a deeper network with more hidden layers could first identify each of those individual elements, then combine them across layers to understand the broader context of the scene [55, 56, 59].

The layered architecture makes it so that DL models can extract increasingly

nuanced features from raw data. Early network layers may model basic edges or textures, while deeper layers recognize more complicated information [59].

## 2.2 Deep Learning in MRI

DL is applied as a tool for MRI development with increasing frequency. Its applications range from image reconstruction [62], to $B_0$ correction [63], MRI image synthesis [64], RF pulse design [65], anatomical modeling [66], $B_1^+$ map synthesis [67], and 10g-averaged SAR assessment [39]. Since numerous literature reviews [62,68–70] already provide an updated status of DL for MRI, after a brief description of the most common network architectures, the focus here will remain with DL for SAR-related applications.

### 2.2.1 Common Deep Learning Architectures in MRI

#### Network Architectures

A neural network architecture is the arrangement of the layers, units, and connections that make up a neural network. This includes how many layers the network contains, how many neurons are in each layer, how those neurons are connected, and what types of operations occur at each stage. The architecture defines how information flows from the input to the output and determines how the network transforms and learns from data. Different architectures are suited to different tasks. For example, convolutional neural networks are often used for computer vision tasks (and are therefore the most abundant choice in MRI literature), while recurrent or transformer-based architectures are used for sequential data. The architecture determines how much capacity the model has to extract information from the data, which influences its performance and its susceptibility to underfitting or overfitting [55, 56, 59].

#### Convolutional Neural Network

Convolutional neural networks (CNNs) [1,71] use convolutional layers that apply filters to small, local regions of the input. As shown in Figure 2.2.1, these filters slide across the input image (ie. sliding window) and detect spatial features like edges, textures, or shapes. As the data passes through deeper layers, the network can combine these simple patterns to represent more elaborate structures [59].

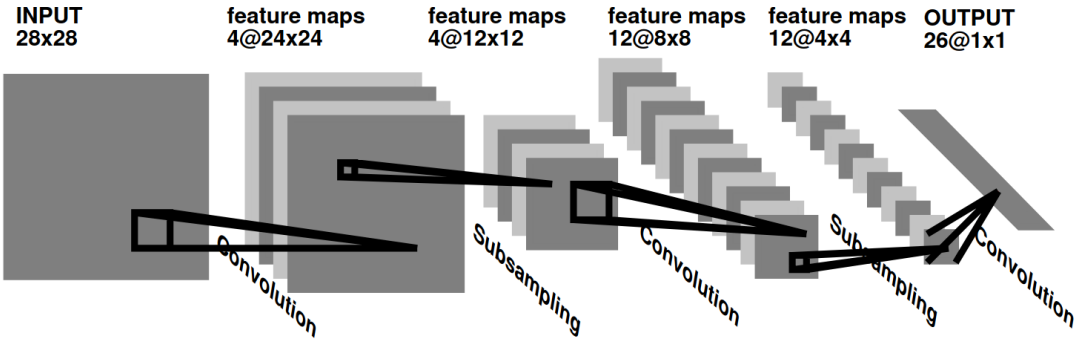


Figure 2.2.1: Illustration of a simple Convolutional Neural Network (CNN) that receives a 28×28 pixel input image and processes it through successive layers to generate feature maps. The image first undergoes convolution using a sliding window, followed by subsampling (pooling), and then another round of convolution and subsampling. This hierarchical feature extraction continues until the output layer, which yields 26 features suitable for classification. Figure retrieved from LeCun and Bengio (1995) [1].

The use of CNNs is ubiquitous in MR literature. For instance, they were used in one study to design static pTx pulses for 7T MRI [65] and in another [72] to correct motion artifacts in fetal brain MRI scans.

### U-Net

U-Nets [2] are a type of CNN originally developed for biomedical image segmentation. As shown in Figure 2.2.2, the architecture is shaped like a "U," with a contracting path on the left and an expanding path on the right. In the contracting path (convolution, or downsampling), the network applies a series of convolutional and pooling layers to reduce the spatial dimensions of the input while learning increasingly abstract representations of the features. In the expanding path (transposed convolution, or upsampling), the network recovers spatial resolution and produces an output that matches the size of the original input. U-Nets also have skip connections, which link layers in the contracting path to their corresponding layers in the expanding path. These connections allow the network to reuse spatial information that would otherwise be lost during downsampling, improving the accuracy of the final output.

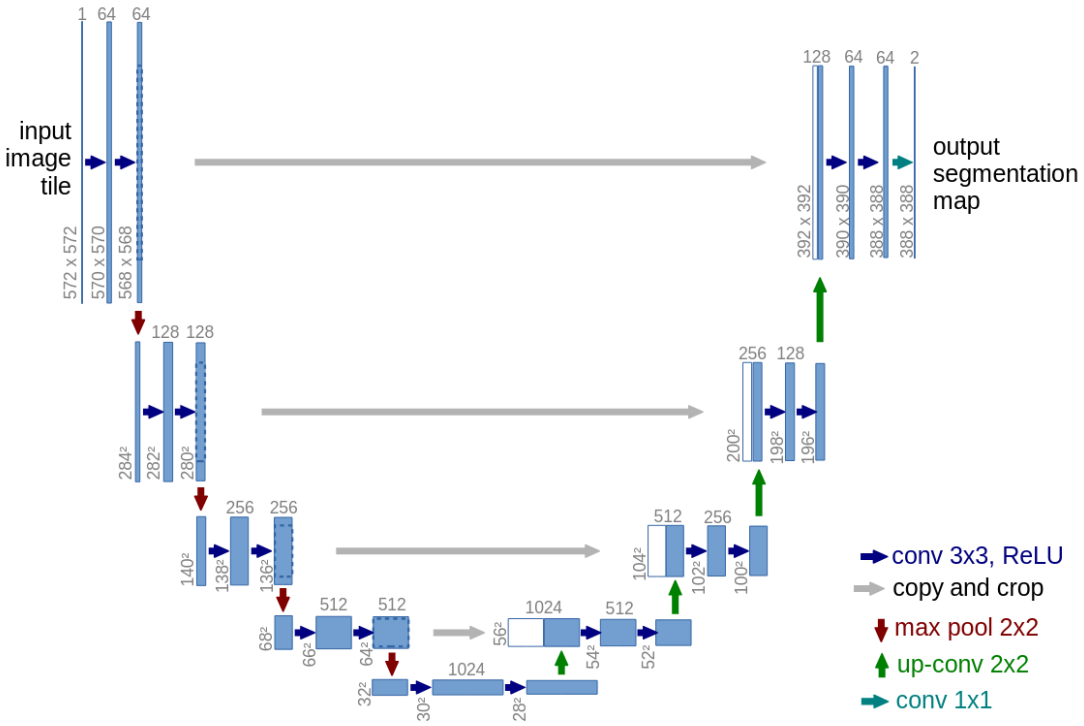


Figure 2.2.2: Illustration of U-Net architecture. The blue boxes are multi-channel feature maps. The values at the top of each box correspond to the number of channels. In the lower left edge, the x-y-size is listed. The white boxes are copied feature maps. The operations which are denoted by the arrows are described in the legend. Notably, "copy and crop" is equivalent to the aforementioned skip connections. This figure was retrieved from the original U-Net publication from Ronneberger [2].

Many variations of this basic U-net architecture exist, such as the 3D U-Net [3]. 3D U-Nets are designed for volumetric data, meaning they can process 3D inputs. Standard U-Nets operate on 2D images and apply convolutions in two dimensions, so they process each slice of a volume independently. This ignores information along the third axis, which can be a limitation when the structure of interest spans multiple slices. As illustrated in Figure 2.2.3 3D U-Nets replace 2D convolutions with 3D ones, allowing the network to extract spatial features across all three dimensions. This added depth increases memory demands but results in better segmentation performance for volumetric data. The network is trained using annotated volumetric datasets, with the aim of producing accurate segmentations of anatomical structures. 3D U-Nets can interpret the spatial context of neighboring voxels along all three axes, which improves performance when dealing with volumetric medical data.

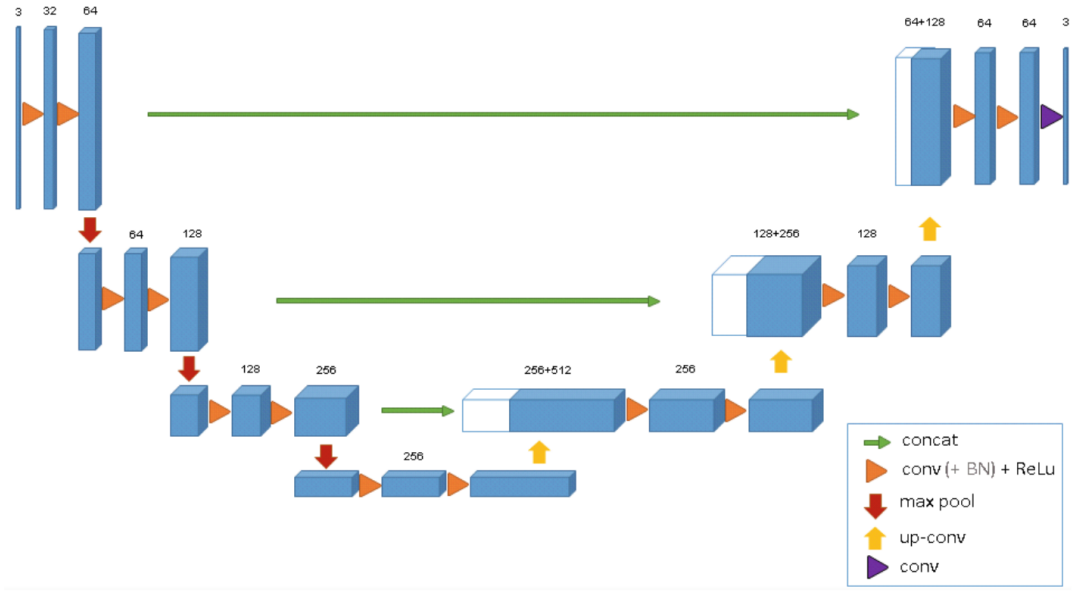


Figure 2.2.3: Illustration of 3D U-Net. The diagram is nearly identical to the one found in Figure 2.2.2, except that the data is volumetric. **BN** stands for batch normalization. This figure was retrieved from Cicek et al., (2016) [3].

In recent work, 3D U-Nets have been adapted to applications besides segmentation. For example, they have been used to predict subject-specific $B_0$ field changes due to head motion during 7T MRI [73], as well as to estimate spatially resolved SAR distributions in subject-specific anatomical models [74].

### Generative Adversarial Network

Generative adversarial networks (GANs) [75] generate new data that resembles a given training dataset. A standard GAN is composed of a generator and a discriminator, which are two separate networks. During training, the networks compete with one another. The generator uses random noise to produce data that look like the ground truth while the discriminator receives both ground truth data and generated data and is tasked with determining which is which. During training, the generator becomes better at fooling the discriminator, while the discriminator becomes better at detecting fakes. This back-and-forth dynamic pushes both networks to improve over time. The simplified architecture is illustrated in Figure 2.2.4.

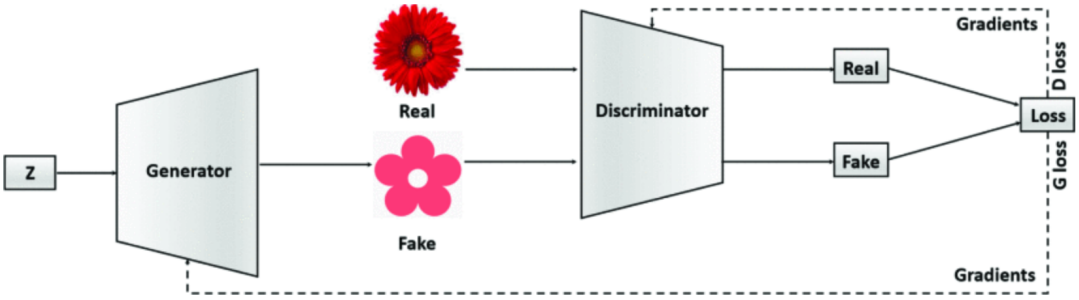


Figure 2.2.4: Illustration of a Generative Adversarial Network (GAN). $Z$ is a random noise vector used by the Generator to produce fake images (e.g., the fake flower). Both the fake image and a real image (e.g., photorealistic flower) are fed into the Discriminator, which is tasked with distinguishing between real and fake inputs. The Discriminator outputs a probability or binary judgment indicating whether an image is real or fake. *D loss* (Discriminator loss) encourages the Discriminator to correctly classify real versus fake images, while *G loss* (Generator loss) incentivizes the Generator to output images that the Discriminator is inclined to label real. Gradients from the loss function flow back to update both networks. The diagram is retrieved from Krichen [4]

.

The original GAN creators [75] apply an art forger and art critic analogy to illustrate the interplay. Once trained, the generator can produce new data samples that closely match the distribution of the original dataset. Like with U-Nets, variations rather than standard versions of the GAN are more common in the MRI literature, such as the CycleGAN, StyleGAN, and conditional GAN.

### CycleGAN

CycleGANs [5] are a variation of the standard GAN architecture. Standard GANs are often referred to informally as vanilla GANs. CycleGANs learn mappings between two image domains without requiring paired examples. Unlike vanilla GANs, which rely on direct input-output pairs, cycleGANs use a cycle-consistency loss (Figure 2.2.5) to enforce structure in unpaired translation tasks. The architecture includes two generators and two discriminators. One generator learns to translate images from one domain to another, and the other generator does the reverse. If an image is translated from one domain to another and then back again, the result should resemble the original input. This cycle-consistency constraint makes the network preserve relevant content while changing style or domain.

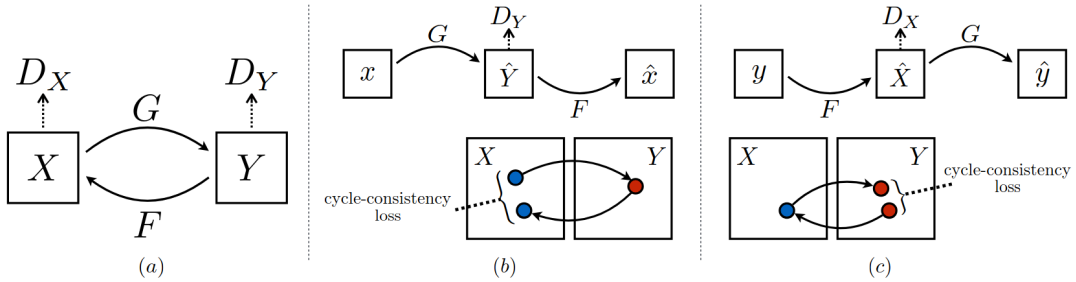


Figure 2.2.5: Illustration of a cycleGAN architecture retrieved from Zhu et al., (2017) [5]. A: Architecture overview where $X$ and $Y$ are two domains, $G$ and $F$ are two generators, and $D_x$ and $D_y$ are two discriminators. While $G$ takes an image from domain $X$ and produces a synthetic image that resembles an image from domain $Y$, $F$ performs the reverse. $D_y$ distinguishes real $Y$ images from those generated by $G$, and $D_x$ distinguishes real $X$ images from those generated by $F$. In the end, the goal is for $G$ and $F$ to generate believable images in their respective domains while preserving the image content. B: The cycle consistency concept where $x$ is converted to a fake image $\hat{y}$ in domain $Y$ using $G$. This is followed by $F$ converting $\hat{y}$ back to $\hat{x}$ in domain $X$. The purpose of the cycle consistency loss is to check whether $x$ is close to $\hat{x}$. Closeness would indicate image content preservation during translation. C: Cycle consistency in terms of using $F$ to generate $\hat{x}$ from $y$ (in domain $Y$). Then $G$ is used to turn $\hat{x}$ back to $\hat{y}$ in domain $Y$.

CycleGANs have been applied in medical imaging to improve data quality and enable cross-modality translation. For example, Czobit et al. [76] used cycleGANs to map 1.5T MRI images to 3T MRI images. Various other studies used cycleGANs to create MRI images from CT scans [77–79]. These applications are particularly popular in contemporary MRI trends which are focused on democratizing MRI, and spreading its unrivaled utility to countries which otherwise do not have access to such expensive equipment and the resources to operate it [80].

### StyleGAN

In StyleGANs [6], mapping networks first transform a latent vector into a style space. In the context of machine learning, latent space is a compressed, abstract representation where each point corresponds to a different possible output. It is the domain from which random vectors are sampled and passed to the generator. Each point in this space maps to a specific image, and small shifts within the space ideally produce gradual changes in the generated output. A latent vector

is a single point in latent space. It is analogous to a set of instructions for the generator, though these instructions are abstract and not directly interpretable. The model learns how different values in the vector correspond to visual features, but those associations are not labeled. Each latent vector encodes a mix of properties like shape, texture, or color, and slight changes in the values lead to smooth variation in the final image. Rather than feeding the latent vector straight into the generator, a StyleGAN passes it through several fully connected layers to produce intermediate vectors. A fully connected layer is a type of neural network layer where each node is connected to every node in the previous and next layer, allowing the model to combine and transform input features without preserving spatial structure. Intermediate vectors modulate the convolutional layers using adaptive instance normalization (AdaIN). AdaIN adjusts the mean and variance of feature maps at each layer based on the style vector. This gives the model control over image attributes at different levels of detail, from overall structure to fine detail. An illustration of the StyleGAN architecture is presented in Figure 2.2.6.

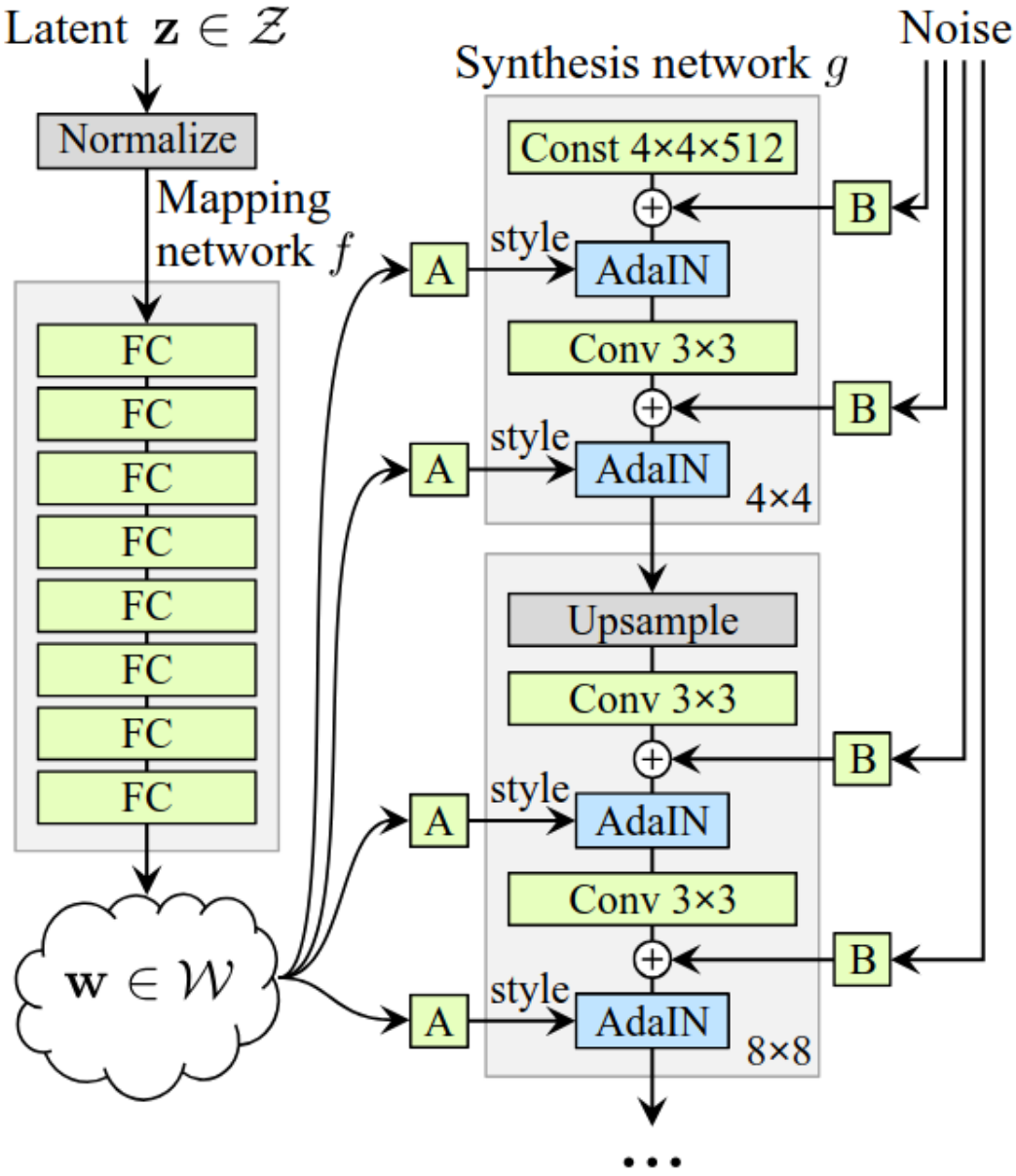


Figure 2.2.6: The StyleGAN architecture. Left side: $z$ is a latent vector sampled from a standard normal distribution, which is normalized before being passed through 8 fully connected layers that transform $z$ into $w$ (the intermediate latent space). $w$ is then sent to the synthesis network via affine transformations ($A$) to control the style of the image. Right side: The network starts from a learned constant tensor of size $4 \times 4 \times 512$. Each block receives style information from $w$ through AdaIN layers, which adjust the mean and variance of the image features. Image diversity is introduced using per-pixel random noise inputs ($B$). The network grows progressively from low ($4 \times 4$) to higher resolutions (e.g., $1024 \times 1024$) through upsampling. Each resolution block includes convolution and AdaIN layers. Image adapted from Karras et al. [6].

For example, StyleGANs were used by Fetty et al. [81] to also map CT scans to MRI scans and by Gullmar et al. [82] to track disease progression from T1-weighted 3T MRI scans.

### Conditional GAN

In conditional generative adversarial networks (cGANs) [83], both the generator and the discriminator receive additional input that conditions the generation process. This conditioning variable can be a class label, an image, or any other information that guides the model toward producing more targeted outputs (Figure 2.2.7). In the generator, the conditioning input is combined with a noise vector to produce samples that match the desired condition, while the discriminator evaluates whether a given sample is both realistic and consistent with the conditioning input. cGANs generate a specific type of output based on a known input, like image-to-image translation.

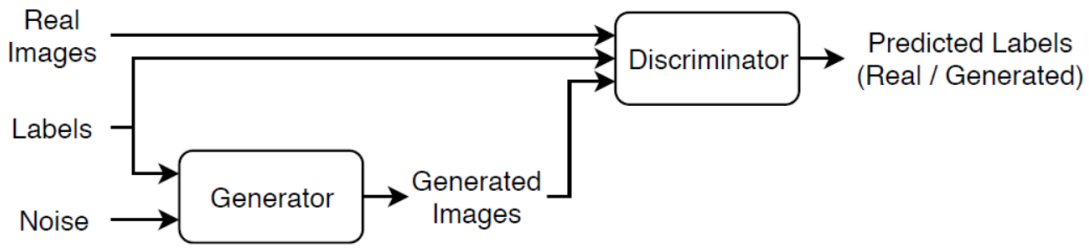


Figure 2.2.7: cGAN network architecture. The concept is largely similar to the GAN architecture, with the exception that labels are included in the training process. The real and generated images are both presented to the discriminator with labels. The discriminator is tasked with evaluating the realness of the images as well as whether they match the given label. Diagram retrieved from Ref. [7].

cGANs were used by Ref. [8] to predict the effect of motion on $B_1^+$ maps and by Ref. [39] for subject-specific 10g-averaged local SAR assessment.

## 2.3 Deep Learning and SAR

Efforts have been focused on improving participant safety while maintaining scanner utility during UHF MRI by implementing DL methods alongside $B_1^+$ mapping [39]. Meliado et al. [39] used DL to predict 10g-averaged local SAR distributions during 7T MRI. Since traditional SAR estimation methods rely on

generic body models and extensive electromagnetic simulations, which can be time-consuming and may not accurately reflect individual anatomical differences, the authors trained a cGAN using simulated data to learn the relationship between subject-specific $B_1^+$ maps and corresponding local SAR distributions. Because $B_1^+$ maps are complex valued, and the proposed network is not designed for complex data inputs, the authors split the $B_1^+$ maps into real and imaginary parts and combine them with a binary body mask containing tissue and air. The resulting model was able to provide fast and subject-specific SAR assessments which are designed to reduce the need for conservative safety margins and improve the efficiency of MRI protocols.

With a similar goal towards more efficient but safety-centered MRI protocols, the same group [84] applied a probabilistic model to improve how safety margins are applied to 10g-averaged local SAR predictions in 7T MRI. Standard methods use a fixed linear safety factor to avoid underestimating SAR, but this might inflate SAR estimates which dampen scanner utility. The authors introduce a conditional safety margin that adjusts based on the estimated SAR value, using the conditional probability distribution between predicted and ground truth SAR. The method was applied to five different SAR estimation strategies, including a CNN trained on simulated $B_1^+$ maps. In this case, the $B_1^+$ maps were split into magnitude and phase parts in order to be suitable for the neural network input. The deep learning method performed best. This approach is also inspired by the MRI research community's motivation to allow safety margins to be more tailored and less conservative.

Other studies have expanded on this idea. Gokyar et al. [85] developed a multi-channel 3D CNN to predict 10g-averaged local SAR distributions using both $B_1^+$ maps and anatomical MRI data. The model was trained on a large simulated dataset to learn the relationship between subject-specific anatomical features and local SAR distributions under various RF shim settings. The CNN architecture incorporated a $B_1^+$ map, a phase-reversed $B_1^+$ map, and structural MR images. This way, the authors did not have to split the $B_1^+$ into real and imaginary or magntitude and phase parts. Instead, the phase information was implicitly encoded in the combined data through the phase-reversed $B_1^+$ map. Overall, this approach enabled rapid and more accurate subject-specific SAR assessments compared to conventional methods.

The way in which the three above-mentioned studies handle complex data in-

formed the way in which complex data are handled in Chapters 4 and 7, where complex data types were also split into magnitude and phase as well as real and imaginary parts.

To optimize the prediction pipeline even further, Brink et al. [86] developed a ForkNET [87] to predict subject-specific 10g-averaged local SAR distributions for 7T pTx MRI using a single T1-weighted scan. Basically, a ForkNET is a type of CNN which starts with a common set of layers and then splits into several branches, each of which is responsible for predicting a different target. These branches can learn separate tasks while benefiting from shared feature extraction in the early layers. The authors trained the ForkNET to generate personalized body models from T1-weighted images, which were then used to estimate local SAR distributions.

Most recently, Wang et al. [88] tasked a 3D Attention U-Net with predicting subject-specific SAR distributions in 7T MRI. The network is trained on a large synthetic dataset generated by a high-dimensional model representation (HDMR) method, which approximates the relationship between tissue dielectric properties and E-field distributions. The 3D Attention U-Net takes five input channels (relative permittivity, conductivity, and three components of the E-field) and outputs three channels corresponding to the predicted E-field magnitudes, which are then used to compute voxel, 10g-averaged, and peak spatial SAR. This avoids the need for full electromagnetic simulations for each subject, reducing prediction time. The purpose of the approach is to enable rapid SAR estimation during coil design or real-time safety monitoring.

While recent advancements in local SAR prediction using DL are numerous, none of the studies mentioned focus on the effects of patient motion on local SAR distributions and related subject safety concerns. Motion remains a major gap in DL-based SAR modeling.

### 2.3.1 Deep Learning, SAR Considerations, and Motion

A study [8] from the same group as the present author used cGANs to correct for motion-induced RF field changes during 7T MRI scanning using pTx. The authors trained a set of cGANs to predict subject-specific complex $B_1^+$ maps following rigid in-plane head displacements. In this study, the complex data was split into magnitude and phase parts. Each cGAN was trained on simulated data

using a U-Net-based generator and a convolutional discriminator. The input to the network was a centered $B_1^+$ map (magnitude or phase), and the output was the predicted $B_1^+$ map (corresponding to centered input, magnitude or phase) at a displaced position. By doing this, the authors aimed to enable real-time tailored pTx pulse redesign based on the predicted fields rather than static maps, which are often invalidated by patient motion. They showed that excitation errors caused by motion could be reduced when pulses were designed using the predicted maps, with improvements in flip-angle homogeneity. This paper did not focus on SAR as a design constraint and prioritized mitigating compromised image quality at UHF strengths. The results state that the approach did not increase 10g-averaged local SAR in most head slices, and therefore did not significantly worsen existing patient safety concerns.

The trajectory of this thesis commenced with a continuation of the work from Ref. [8]. Since the $B_1^+$ map prediction approach was successful, it was deemed appropriate to expand it to address subject safety. For this reason as well, as the reader will discover, the magnitude-phase split was applied to begin with.

## 2.4 Summary and Outlook

This chapter introduced basic concepts in DL relevant to the experiments presented in the remainder of the thesis, with a focus on network architecture, optimization, and how and why deep learning has been applied to local SAR prediction in MRI.

The rest of the thesis is based within this research domain. It begins by detailing the process and obstacles of recreating the Python environment from Ref. [8], along with an attempt to reproduce their results (Chapter 3). The research objective here is to caution against practices which may hinder project reproducibility in similar projects, as well as to encapsulate the current project's pipeline for validity and reproducibility purposes. The following chapter applies that same cGAN to investigate how rigid head motion affects data types that feed into SAR estimation (Chapter 4). The objective in this chapter is to discover the optimal datatype for the overarching objective of designing a SAR-estimation pipeline for head motion during UHF MRI applications. An unexpected artifact observed during that work prompted a deeper investigation into its origins (Chapter 5). These investigations were conducted to shed light on the effects and causes of

this artifact for the development of the current project as well as for future use by the related scientific community. This artifact investigation in turn led to a more successful use of a U-Net, which proved effective at predicting the effect of motion on local SAR distributions (Chapter 6). This chapter is the climax of the thesis, where the proposed proof-of-concept pipeline for using machine learning to estimate the effects of head motion on local SAR distributions is ultimately introduced and evaluated by pTx application. The final chapter revisits the original datatypes, applying the U-Net in a new context (Chapter 7). The objective here is to compare two networks and pipelines, and to evaluate the utility of more SAR-related datatypes for the aforementioned deep learning pipeline development research goals. Taken together, the project demonstrates the usability of DL tools like cGANs and U-Nets as well as the limitations imposed by model design choices, data preprocessing, and reproducibility barriers. The overall goal of the project is to lay the groundwork for a pipeline which guarantees patient safety while ensuring that the benefits of UHF MRI can be used to their fullest potential. Relevant information is presented through affirmative and null results, both of which are of value to the pertinent research community.

# Chapter 3

# Blood Sweat and Tears: Reproducing the Python Environment and Previous Deep Learning Results from the Lab

## 3.1 Reproducibility Issues in Python for Deep Learning Applications

### 3.1.1 Limitations of Open Source Software

Open source software like Python offers many benefits. Most importantly, it is free, which means that for the purposes of the present project and the lab's preceding endeavors, the authors did not have to seek extra funding. Python also offers a myriad of applications, especially those dedicated to deep learning. Python comes with a vast standard library that provides ready-to-use modules and packages for common tasks, reducing the need to write code from scratch. Moreover, Python has a large and active community of developers who contribute to enhancing the language, creating libraries, and providing support and resources for users.

While the authors benefited massively from the aforementioned aspects of Python's advantages, there were a few catastrophic setbacks which are directly attributed to the nature of open source software in itself, and cost the present project over

half a year of stagnation.

In the literature, it is evident that environment and compatibility-related issues are common. One reported issue involves versioning and backward compatibility, where updates to Python and its libraries frequently introduce changes that break existing code [89]. For example, discrepancies between NumPy's (a Python library) handling of data types across versions [90] have been noted as sources of error in reproducibility. There is evidence of insufficient documentation for older versions of specific libraries, limiting the ability to trace results [91, 92]. A lot of these issues can be attributed to the lack of standardized workflows for setting up reproducible environments in Python, such as Docker or Conda environments [93]. Open-source community support, comprehensive documentation, and robust environment management tools are useful for ensuring reproducibility in machine learning projects.

The work presented in the next three chapters began with reproducing published work by a researcher in the same lab at the research center. The work is a pipeline designed to predict the effects of rigid head motion on $B_1^+$ maps at 7T MRI scanning using pTx [8]. The following complications were discovered while reproducing this pipeline.

### 3.1.2 Brief overview of methods for environment creation

Environments can be created anew or by copying an existing environment created by another programmer. To copy an existing environment, users must use `.yml` files or `.txt` files. A `.yml` file, short for YAML (Yet Another Markup Language), is a human-readable data serialization format commonly used for configuration files and environment management in software development. It is used to define dependencies, specify software packages, and manage runtime settings [94]. YAML files use indentation-based structure, making them easy to read and write, while also supporting key-value pairs, lists, and nested elements [95]. In machine learning and deep learning workflows, `.yml` files are relied on to ensure that computational environments remain consistent across different systems, reducing dependency-related errors and compatibility issues.

`.txt` files, or a simple text document, can be written to contain the environment requirements as well. `.txt` files are universally compatible and have no formatting requirements. They can be downloaded directly through an existing Python

installation using Conda.

Besides creating environments from an existing `.yml` file, they can be created anew by the programmer. To create environments in Python from scratch, required libraries can be downloaded from many available package management systems. The first is Conda, an open-source package management and environment management tool originally created for Python. It facilitates the installation, dependency resolution, and management of software packages across multiple platforms. Unlike traditional package managers such as Pip (described below), Conda is capable of handling both package dependencies and binary dependencies, ensuring compatibility across different operating systems. It enables users to create isolated environments, allowing multiple versions of libraries and software to coexist without conflicts. This is particularly useful for reproducing work like in the case presented here. Conda also provides an extensive repository of precompiled packages through Anaconda and Conda-Forge, which streamline installation processes and optimize performance. Users can install packages using brief commands such as `conda install package_name`, while entire environments can, in theory, be exported and shared using `.yml` files. When successful, Conda's cross-platform design and efficient dependency resolution make it a valuable tool for managing multiple software-based projects [96].

The work discussed here also uses Anaconda, a commercial distribution of Python and R that includes Conda as its package and environment manager. It contains a repository of pre-compiled, stable, and validated packages. The Anaconda repository, the default Python channel, is maintained by Anaconda, Inc., and is designed to contain stable package versions created by professionals, rather than the open-source public software community. The main draw of Anaconda is that it offers a streamlined package installation method [97].

Conda-Forge is a community-driven, open-source repository that provides a broader selection of packages than Anaconda's default repository. It is maintained by a volunteer community, ensuring more frequent updates, wider package availability, and support for diverse platforms. Because it is open source, Conda-Forge sees continuous contributions from developers, allowing the latest versions of packages to be available more quickly than in the Anaconda repository. The downside is that there is a higher potential for compatibility issues resulting from the rapidly evolving software available on the channel [98].

We also trialed the use of Pip for our environment creation purposes. Pip is

used to install, update, and manage software packages from the Python Package Index (PyPI). Pip was designed to simplify dependency management by automatically resolving and installing required libraries and frameworks. To install packages with Pip, users must call `pip install package_name`. Like Conda, Pip is meant to provide virtual environment compatibility, and enable projects to maintain isolated dependencies. Unlike Conda, which manages both packages and their dependencies across multiple languages like R and Java, Pip is specifically designed for Python environments. Pip can struggle with dependency conflicts, requiring manual resolution when multiple packages depend on different versions of the same library [99]. An efficient but often damaging feature of Pip is that installing packages through it automatically changes the builds and versions of packages previously installed in the environment. This can cause a snowball effect of dependency issues which are difficult to trace [100, 101].

### 3.1.3 Version Incompatibilities Between Conda-Produced .yml File and Actual Version Number in Environment

Identical scripts to Ref. [8] were used to preprocess data in Matlab for training, validation, and testing input to the cGAN. The only difference between the two pipelines was the body models (from the Virtual Population [102]) used to create the datasets. Since the research lab discovered issues in the data extracted from Dizzy, it was discarded to ensure that whatever error arises during reproduction cannot be attributed to the data. Ref. [8] used Duke and Ella for training, Billie for validation and testing (alternating slices), and Dizzy for testing. The reproduced pipeline used Ella, Fats, and Glenn for training, Billie for validation, and Duke for testing. This decision was made because, in any case, a completely faithful reproduction was not possible without access to Dizzy. It was therefore reasonable to take advantage of the newly acquired body models and increase the training dataset with the expectation that it would improve the deep learning network's prediction accuracy. Moreover, it was assumed that since the published work from Ref. [8] was intended to be versatile and have applications across anatomies and demographics, that altering the training, validation, and testing dataset configurations would not be damaging (more on this in Chapters 5, 6, and 8).

After preprocessing, a series of environments were created to run the network. At

first, to ensure full compatibility between the reproduced and original work [8], the same OS (computer operating system) (Windows 10) and a similar GPU (NVIDIA's GeForce GX 1050 vs. the original GTX 1050ti) were used to run the networks in the same environment. The environment (from here forward called $env_{orig}$) was sent from the original PC to the new PC as a `.yml` file and opened on the new PC.

While transferring $env_{orig}$ with the `.yml` file, a peculiar incompatibility was observed: the versions of different packages in the `.yml` file differed from the actual version of the packages in $env_{orig}$. For example, the TensorFlow [1] version displayed using the

```
print("TensorFlow version:", tf.__version__)
```

command in Spyder [2] on the original setup used in [8] differed from the version displayed in the `.yml` file (version 1.14.0 in .yml vs. version 2.3 in Spyder).

Discrepancies such as incompatible package version reporting between IDEs and `.yml` files have been described in the scientific community. For example, a study has found that Python dependency management tools, such as Conda, can fail to guarantee consistency between environments due to mismatches in package versions and OS dependencies [103].

Installing packages via both Conda and Pip within the same environment can lead to version discrepancies, as `conda list` (ie. the command used to list all of the packages in a given environment) may not reflect the most recent package versions installed by Pip [96]. Dependency conflicts can also arise when different packages require incompatible versions of the same dependency, a situation colloquially referred to as "dependency hell" [104].

A simple environment transfer, as suggested by the Conda cheat sheet [105], was not reliable. The Conda Cheat Sheet is a brief and functional reference guide that provides commonly used Conda commands for package management, environment management, and configuration settings.

[1] Tensorflow is the machine learning framework developed by Google on which the pipeline from Ref. [8] was developed.

[2] Spyder is an open-source integrated development environment (IDE) for Python, designed for scientific computing and data analysis. It contains an interactive environment that combines editing, debugging, and data exploration features. Its main draw is its similarity to the MATLAB interface, a benefit for for the current pipeline which is built in both MATLAB and Python languages.

The same issue arose when attempting to transfer $env_{orig}$ as a `.txt` file, and then downloading the content of the `.txt` file through Conda using

`conda create -n` $env_{new}$ `--file` $env_{orig}$`.txt`

This phenomenon has been previously observed by the Python community, such as in Ref. [106], where the Python version specified in an original environment (3.8.5) differed from the version of Python transferred by the `.yml` file (3.8.13). The user eventually resolved the issue by uninstalling their base Python version (3.10) from which the environment was originally created, and replacing it with Python 3.9. For context, in order to create a Python environment, Python (any version) must already be installed on the OS. The reason for why this was the solution remains a mystery. Such reproducibility-related inconsistencies have inspired the development of containerization tools like Docker. [3] Unfortunately, the researchers were unaware of Docker during the development of the current and foundational project, and therefore did not make use of it.

Creating a local environment from the inaccurate `.yml` file was nonetheless unsuccessful, with initial attempts using the commands from [105] resulting in

`metadata-generation-failed`

among other errors. Many trials using various commands suggested by the expansive online Python community [109] were employed to no avail.

After numerous attempts in the above-specified Windows OS, the same was tried in CentOS Linux 7 (Community ENTerprise Operating System version 7), since the preceding project from which $env_{orig}$ was being transferred was also partially implemented in CentOS Linux 7. Remarkably, in 2019 during the production process of Ref. [8], the author [110] reported that the exact same environment and exact same code produced different results while predicting the effects of head motion on the phase of complex $B_1^+$ maps. While in Windows 10, the network predictions had little error, the same method run in CentOS Linux 7 did not produce the lowest error generator weights necessary for the evaluation pipeline

[3] Docker is a tool that allows developers to package applications along with all their dependencies into a 'container'. It provides a way to create stand-alone environments where Python code can run consistently across different machines. Instead of installing Python and all required libraries manually, a researcher can create a Docker container that includes everything the application needs, such as the correct Python version and required packages to avoid dependency conflicts [107]. While Docker can encapsulate dependencies and their versions, ensuring consistent results across different systems, it can have limitations, like when working with hardware-specific drivers or OS configurations [108].

and produced predicted images composed of exclusively NaNs (not a number(s), which represent undefined or unrepresentable numerical values). Magnitudes of $B_1^+$ maps were predicted with equal success in Windows 10 and CentOS Linux 7.

### 3.1.4 Build and Dependency Incompatibility Issues Preventing Identical Environment Creation From Scratch

Since the `.yml` file transfer was unsuccessful, the next step was to recreate $env_{orig}$ from scratch. The new environment's suitability was evaluated by 1) whether the cGANs could run error-free and 2) the quality of the cGAN-predicted images. In this process, considerable effort was devoted to replicating not only the specific software dependencies but also the exact CUDA and cuDNN versions required for hardware acceleration. As noted in prior work, mismatches in CUDA or cuDNN [4] can result in unexpected behavior or errors when running deep learning frameworks including TensorFlow and PyTorch [114].

A total of 18 new environments were created to test the neural networks. New iterations of the environments were advised by resulting error messages in the terminal during environment creation or cGAN script execution. Only some of the resulting errors are documented here, as not all were recorded over three years ago. The exact methods to create five iteration of the environment which preceeded the final one used for the remainder of the thesis are described in A**2**.

*env1-5* environments were trialed in the Windows 10 OS with a GeForce GTX 1050, Windows 10 OS with a GeForce GTX 1050ti, CentOS Linux 7, and on the original PC used for Ref. [8]. In all of the above scenarios, it was possible to open Spyder and set up the training network, but it was not possible to recreate the results from Ref. [8].

One primary issue encountered during the experiments was OOM (out-of-memory) errors, which are likely inaccurate and not indicative of the actual underlying cause of the failed script, considering the similarity of the hardware used for the original and reproduced projects. Examples of OOM errors can be found in A**3**.

[4] CUDA (Compute Unified Device Architecture) and cuDNN (CUDA Deep Neural Network Library) are technologies developed by NVIDIA [111] to accelerate computing, notably in deep learning and general-purpose GPU applications. While CUDA is more general purpose (simulations, crypto currency mining, game development, etc.), cuDNN is designed for machine learning applications. Both methods increase computational speed and run programs more efficiently by means of parallel computing, which, as the name suggests, is a method where multiple processes are executed simultaneously [112, 113].

OOM errors in TensorFlow have been previously reported to stem from inefficient memory allocation or memory fragmentation rather than genuine hardware limitations [115].

The second main issue encountered was GPU recognition failures (for which extensive mitigation measures were taken, including upgrading GPUs, downloading different CUDA versions, altering bash scripts and BIOS, and adding lines of script to the neural network). All efforts were based on suggestions from the online Python community, software developers, and the research center's bespoke IT team. Examples of GPU recognition errors are listed in A**4**.. These issues are consistent with reports indicating that GPU recognition errors often arise from mismatched TensorFlow, CUDA, and driver versions [116].

The third primary issue was premature training termination, such as the kernel restarting without user intervention or the Spyder window closing unprompted after fewer epochs than specified. The last primary issue presented as network-predicted images containing all NaNs after 2, 5, 10, or 60 training epochs. Both issues are indicative of subtle configuration mismatches between Python libraries and their runtime dependencies.

Overall, it must be reiterated that the neural network script and data used for this process were identical to the one successfully applied for the work done in Ref. [8], which was implemented in the same research center. Especially considering the case where the same PC was used with a slightly altered Python environment, since identical recreation of the environment using `.yml` file transfer and manual package loading was not possible, it is plausible that the environment itself rather than any hardware differences is responsible for the vast contrast in cGAN results or code execution.

### 3.1.5 Google Colab Sanity Check

In an attempt to bypass the need to create environments at all, Google Colab [117] [5] was trialed as a potential solution.

[5]Google Colab is a free online platform that lets users write and run Python code in a browser. It supports Jupyter Notebooks and provides access to GPUs and TPUs. A Tensor Processing Unit (TPU) is a type of hardware developed by Google specifically for accelerating machine learning tasks. It is designed to handle large-scale numerical computations, with a focus on deep learning models. TPUs are optimized for operations commonly found in frameworks like TensorFlow and can perform many calculations in parallel, which increases efficiency when training or running models

When using Google Colab, users can store their work in Google Drive and run code in small sections, which helps with testing and adjustment. One major downside is that the platform only permits 5GB of free space, and exceeding that allotment requires a monthly subscription. For this reason, the platform was used for trouble shooting and sanity-checking without intention for full integration, since the full project requires terabytes of data.

In Google Colab, creating separate environments is generally not required. Each session runs in a temporary and isolated workspace that is reset when the runtime is restarted. This default setup helps avoid package conflicts and ensures a clean environment for each use. Results from the Google Colab experiments are summarized in A**5**.

### 3.1.6 The successful environment: $env^*$

After months of trial and error spent creating environments, loading packages, and running cGANs, a sufficiently functional environment was finally discovered in CentOS Linux 7. This was not merely the sixth environment created. The previous five environments described in the appendix were environments for which the author has full documentation. In total, about 18 failed environments are stored on the system, and many others have been deleted during troubleshooting.

$$env^*$$

was created with the following commands:

```
Python=3.7.11
conda install spyder matplotlib==3.4.3 scipy==1.7.1
scikit-image==0.18.3
conda install -c anaconda tensorflow-gpu keras==2.4.*
```

| **conda install** | conda install -c anaconda |
|---|---|
| spyder | tensorflow-gpu |
| matplotlib == 3.4.3 | keras == 2.4.* |
| scipy == 1.7.1 | |
| scikit-image == 0.18.3 | |

Table 3.1.1: The installed packages (rows) and corresponding installation methods (columns) for *env**. Changes from *env*5 (A**2**) are highlighted in yellow. The specified Python version was 3.7.11.

The version of TensorFlow-gpu available from the anaconda repository at the time was 2.4.1, therefore this was the unspecified version downloaded into the environment. The downloaded version of keras was chosen to accommodate for the TensorFlow-gpu version. Additionally, spyder was downloaded first, and the version was also unspecified. The following versions of packages were chosen based on the ones used originally in Ref. [8]. Cudatoolkit and cudann were not specified during environment creation. Instead, the cuda version (11.4) was coordinated after the environment creation by the center's IT team. This was based on requirements within the center's computing systems (CentOS Linux 7 and NVIDIA v100 GPU). The cuda version was later specified in the Bash scripts which deployed the cGANs. A Bash script is a file containing a series of commands written in the Bash (Bourne Again Shell) scripting language. It automates tasks in Linux.

As referred to before, while successful in Windows 10, Ref. [110] was unable to use the same cGAN in the same original environment when attempting to predict the effects of motion on the phase of $B_1^+$ maps when implemented in CentOS Linux 7. In CentOS Linux 7, the network results contained only NaNs. When implementing the same script in the *env** environment, the results were more favorable than any other implementation so far, and so *env** was used for all subsequent investigations in this thesis. Based on this exploration, two things are apparent: one is that the Python environment is responsible for the implementability of a given neural network in different operating systems; the second is that the implementability of a script is entirely dependent on the content of the Python environment in which it is being run.

If Python scripts are so heavily reliant on the environments in which they are being run, and the reproducibility of Python environments is unstable, how re-

producible are many of the Python projects available on the internet and globally across research centers? This question is especially relevant as machine learning research increasingly relies on intricate stacks of dependencies, where even minor changes in package versions or system settings can lead to non-reproducible results [118, 119]. `.yml` environment files offer partial solutions but may still fail to capture nuances in system-level dependencies, making reproducibility an ongoing challenge.

### Comparing results to previous work

The network-estimated magnitude and phase-jittered $B_1^+$ maps using

$$env^*$$

were compared to those produced by Ref. [8] using normalized root mean squared error (nRMSE). The nRMSE between the ground truth and network-estimated magnitude maps were was calculated by

$$nRMSE = 100 \times \frac{\sqrt{\overline{(GT - y)^2}}}{\overline{GT}} \tag{3.1.1}$$

where $GT$ is the ground truth map and $y$ is the network-estimated map. This metric allows a normalized comparison of estimation error relative to the magnitude of the true data, offering a quantitative measure of network performance [120]. While the error from Ref. [8] was 5.4% (compared to 11.5% motion-induced error), the error using $env^*$ was 1.8%, compared to an identical motion-induced error. Higher worst-case network estimation error is observed in Figure 3.1.1 compared to Figure 3.1.2. This reduction in error demonstrates that improvements in training data and environment configurations can significantly alter the network's effectiveness. It is important to note that it could be the case that the training data was the sole cause of the different network estimation error results. No subsequent analysis was performed to evaluate what the underlying reason was for the reproduced results' lower estimation error values.

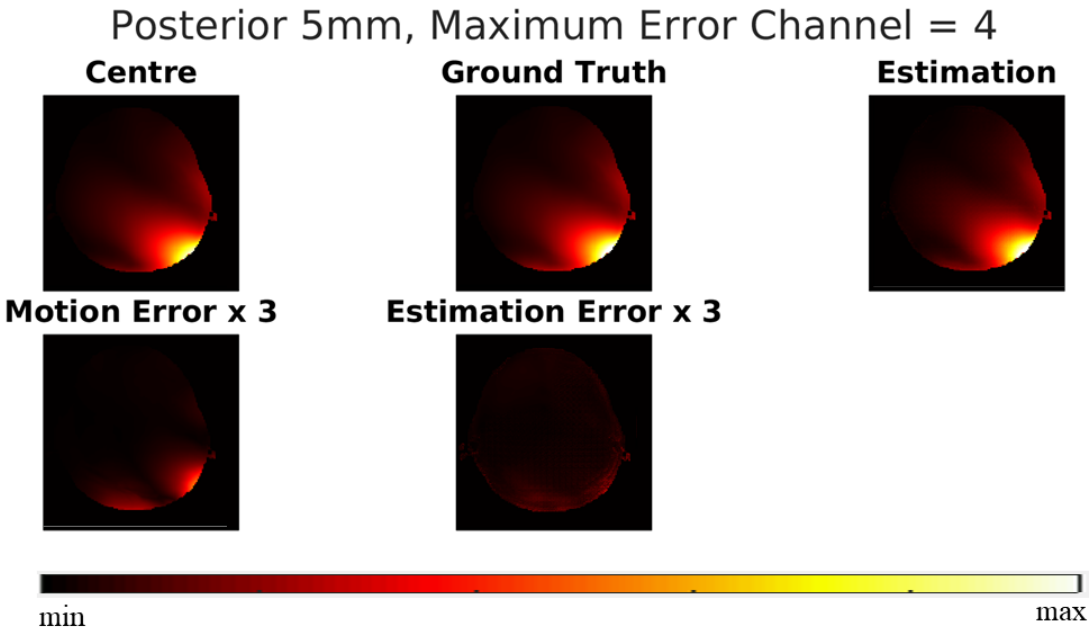


Figure 3.1.1: Maximum intensity projection along the z-axis for the pTx channel producing the worst error in the estimated $B_1^+$ maps from Ref. [8].

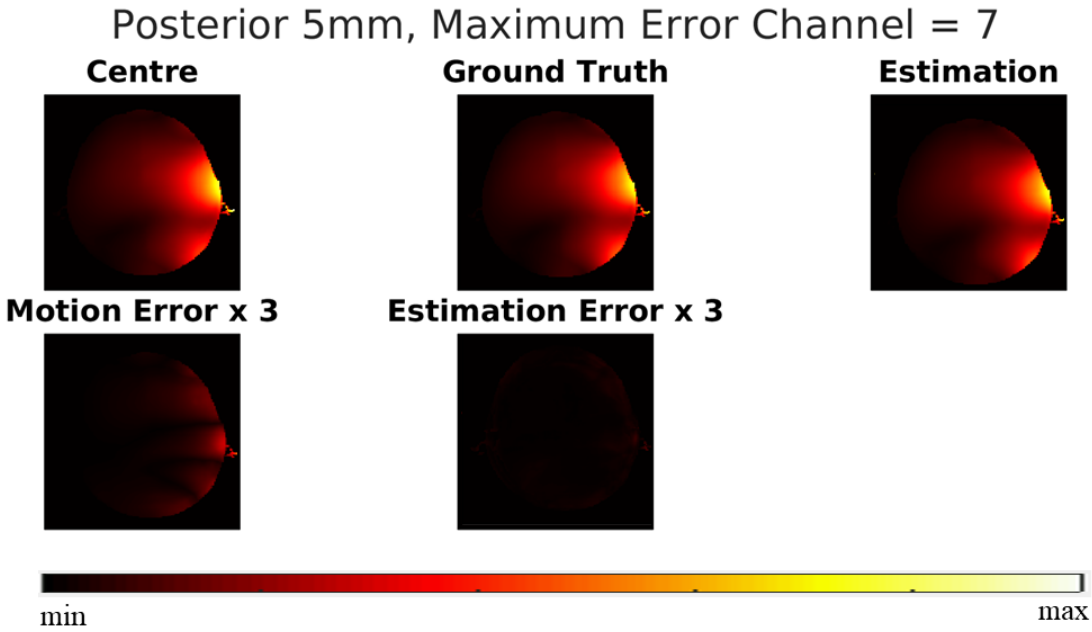


Figure 3.1.2: Maximum intensity projection along the z-axis for the pTx channel producing the worst error in the reproduced estimated $B_1^+$ maps. Compared to Figure 3.1.1, less estimation nRMSE is apparent.

The large discrepancy in the nRMSE of estimated magnitude $B_1^+$ maps could be at least partially attributed to the training dataset content and size. While Ref. [8]

trained on 1,040 slices, the reproduced pipeline trained on 2,754 slices. This increase in training data likely contributed to better network generalization and reduced overfitting, as the larger dataset provides more diverse representations of input-output mappings for the network to learn [121] . The discrepancy could also result from the different environment used to reproduce [8]'s pipeline. The specific differences in TensorFlow library dependencies and GPU drivers between the environments may have subtly influenced numerical computations within the network. There is likely no way to verify either claim.

To maintain dimension, the phase error was calculated by the overall difference between the ground truth and network estimated $B_1^+$ maps, across slices and channels:

$$\angle error = \frac{180}{\pi} \times |\angle e^{(i(\angle GT - \angle y)}| \tag{3.1.2}$$

where $\angle GT$ is the phase of the ground truth $B_1^+$ map, and $\angle y$ is the phase of the network estimated $B_1^+$ map. This formulation quantifies phase discrepancies in degrees, making it intuitive to interpret the angular error. While the error from Ref. [8] was 2.3° (compared to 2.4° motion-induced error, or minimal improvement), the error using $env^*$ was 2.2° (equal to motion-induced error, therefore showing no improvement). This marginal improvement suggests that either phase prediction is inherently more complicated for the network or less influenced by environmental factors compared to magnitude estimation.

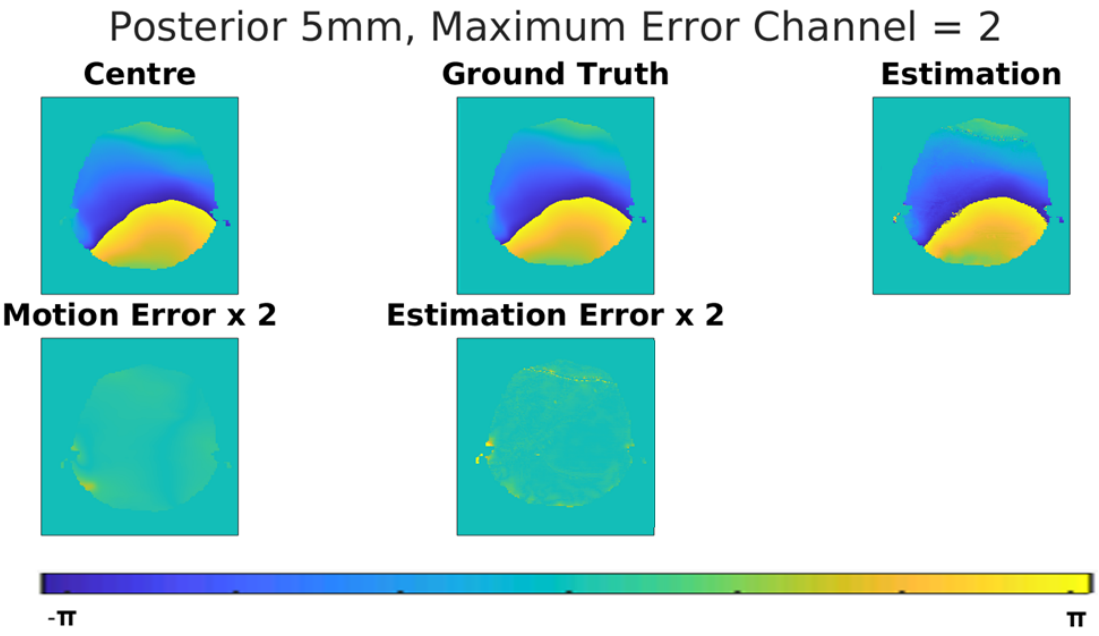


Figure 3.1.3: Slice selected from the center of the head for the pTx channel producing the worst error in the estimated $B_1^+$ maps from Ref. [8]. For emphasis, error is doubled.

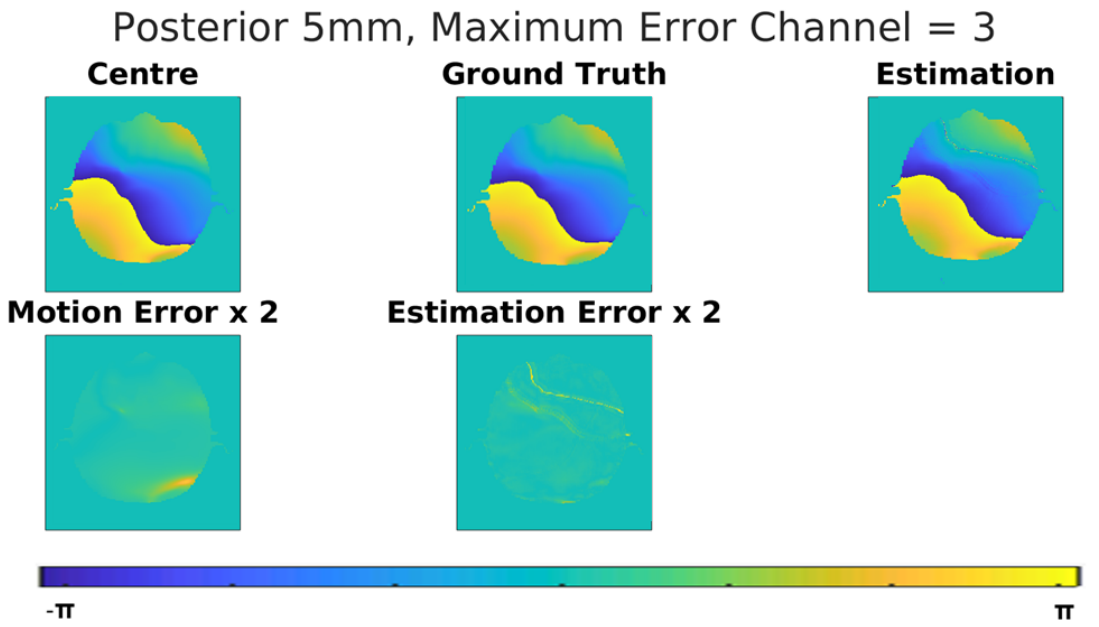


Figure 3.1.4: Slice selected from the center of the head for the pTx channel producing the worst error in the reproduced estimated $B_1^+$ maps. For emphasis, error is doubled. Compared to Figure 3.1.1, about the same amount of error is apparent.

Overall, *env** was successfully used to reproduce the results from Ref. [8]. Major

discrepancies in the magnitude network results could be attributed to the training data and differences in Python environments. Minor discrepancies in the phase network results could be attributed to the general volatility of cGANs and the random shuffling of training data in the preprocessing stage. Randomized training data ordering can result in slight variations in optimization trajectories, especially in GAN-based networks where training is inherently unstable [122]. Importantly, it was confirmed that the results from Ref. [8] are reproducible despite differences in Python environments, and therefore it is valid to apply the methodology to other datatypes and MRI-safety related research goals. Nevertheless, the inability to reuse the original Python environment from Ref. [8] was a major setback in being able to conduct a completely faithful reproducibility investigation. To address this, future work should consider containerized solutions which encapsulate the entire software environment and ensure exact replication across systems. Research centers should adopt standardized methodologies to prevent this issue from recurring.

### *env** reproducibility in terms of environment creation

Two years later, two methods were tested to reproduce the *env** environment: 1) following the above Conda commands and 2) creating and downloading a `.yml` file. The former resulted in an everlasting (48h+) `solving environment` after committing the second line shown above. This prolonged `solving environment` issue is often caused by dependency conflicts or outdated metadata in Conda channels. Over time, as packages are updated or removed, Conda can struggle to resolve complex dependency trees, especially in older operating systems like CentOS Linux 7. This failure was replicated on two other users' [123] [124] CentOS Linux 7-imaged PCs in the same research center. It could be that CentOS Linux 7's use of older libraries and compilers exacerbates these dependency issues. This failure is likely a result of package, channel, and repository updates over the last two years. For example, major updates to TensorFlow and Keras since 2021 have shifted to newer CUDA and cuDNN versions, dropping support for older GPU environments [125]. Furthermore, the scipy and scikit-image library versions specified may no longer align with the older TensorFlow-gpu versions originally used in *env**.

The latter method proved successful. The fact that a `.yml` file transfer was successful (and all package versions remained the same as in the original) indicates that when possible, a `.yml` transfer is the safest environment reproduction option.

Using a `.yml` file preserves the exact package versions, theoretically guaranteeing reproducibility and circumventing the need for Conda to resolve dependencies. But as previously described, even `.yml` files can fail if underlying system-level libraries (e.g., glibc) are incompatible. Therefore, best practice should likely involve documenting both the Python environment and the system environment as a necessary step to take for researchers interested in maintaining their projects' validity. The following information outlines an example of environment-relates issues outside of the current research center.

**STARGAN-v2 environment installation: an example of reproducibility issues**

Package locations changing from channel to channel over time complicates Python project reproducibility. A**6** contains the instructions posted by the GitHub page's authors which can be used recreate the environment required to run the second version of STARGAN [126].

The third line (`torchvision=0.5.0`) causes a `PackagesNotFoundError` which suggests a change of location for torchvision version 0.5.0. This is a common issue encountered by users of Conda and Pip repositories, as package versions may be deprecated or moved to different channels over time [127]. Though it was an intuitive approach, providing the 'recipe' for the Python environment was not suitable for ensuring that other users can reproduce Ref. [126]. The lack of a fully reproducible environment may have been caused by evolving package repositories and their configurations, making it difficult for future users to replicate the exact environment used in the original work.

## 3.2 Discussion

The obstacles of ensuring reproducibility in Python environments for deep learning are a reflection of the fragility of open-source software tools. Version inconsistencies, dependency conflicts, and hardware-specific limitations pose significant hurdles to creating identical environments across systems and subsequently reproducing research results for machine learning investigations. These issues stress the delicacy of reproducibility when projects, like the one developed in this thesis, rely on a large stack of Python libraries and specific configurations.

One finding of this investigation is the inability of environment-sharing tools

such as Conda's `.yml` files to guarantee exact reproductions. This limitation may arise from discrepancies between declared and installed package versions, evolving repositories, and hidden dependencies tied to system-level libraries. The exploration revealed that even slight mismatches in CUDA, cuDNN, or Python library versions could produce vastly different results, a phenomenon widely documented in the literature and by users [107, 128–131].

This chapter has several limitations. First, completely faithful reproduction of Ref. [8] was not possible because the Dizzy body model's simulation data was unavailable at the time of replication. For this reason, study reproduction results and subsequent interpretations are approximate. Second, it is possible that, since the author of Ref. [8] was still in the process of learning Python while developing the published pipeline, some beginner-level errors may have occurred during the creation or transfer of the Python environment to other users, accidentally causing the reproduction process to fail from the beginning. Finally, it is the case that the present author is not a computer scientist and was a novice at the time of commencing the content presented in this chapter. As a result, there may have been errors committed which caused more "blood sweat and tears" than would have been encountered by a seasoned professional during the replication process.

This study demonstrated that a significant portion of the observed performance variability could be attributed to differences in data preprocessing and environment configurations. For instance, the successful $env^*$ environment allowed for lower nRMSE in $B_1^+$ map predictions compared to previous implementations [8], though the non-identical training datasets prevent the certainty of this claim. However, the persistent setbacks in recreating $env^*$ across systems two years later reveal the ephemeral features of Python environments, where even seemingly minor updates can disrupt reproducibility. Overall, these findings call for standardization of practice within research groups.

The processes and issues described in this chapter indirectly serve to advance solving the SAR-motion problem undertaken in the remainder of this thesis. This is because the ability to reproduce the project relies on being able to reproduce the environment. Moreover, any subsequent developments to the pipeline will also likely rely on either the same or a different, bespoke collection of python packages retrieved from specific repositories which are constantly changing and advancing. If in the future a tool suitable for clinical implementation is developed, it will likely too rely on a particular environment in order to function as intended, and

as a result the preservation of the environment will be integral to the utility of the tool. This chapter serves as a call for researchers to be maximally mindful of the environments in which they build their python packages, as well as an explicit guide to how the environment for the thesis's pipeline was developed.

## 3.3 Conclusion

This work has recounted the difficulties of achieving reproducibility in Python-based deep learning projects, especially in environments with many intricate dependencies. Despite months of experimentation, the inability to precisely recreate the original environment from prior research reveals a limitation in Python's ecosystem. The successful implementation of the $env^*$ environment, while a notable achievement, further demonstrates the fragility of dependency management in open-source projects.

Moving forward, reproducibility in Python can be enhanced through several measures. First, using containerization tools can encapsulate dependencies and eliminate inconsistencies arising from system-level variations. Second, comprehensive documentation of both the Python and system environments is essential to preserve the exact conditions for replication. Third, adopting tools to infer and lock dependencies more effectively could mitigate many issues experienced in this work. Finally, promoting the use of tools like Google Colab, which do not rely on complex dependency setups, could also be helpful, but only if access remains free or if researcher centers establish contracts similar to those formed for article submissions to academic journals.

As machine learning applications become more ubiquitous across disciplines and within the author's home research center, safeguarding reproducibility is essential for the integrity and scalability of Python-reliant research.

# Chapter 4

# Exploring The Suitability of Data Types for cGAN Estimation

## 4.1 Transparency Foreword

The work described in this chapter represents a refined version of the initial efforts, produced during the final year of the PhD project. Alix Plumley wrote the original pipeline that was adapted for the following investigations. Plumley's pipeline included the preprocessing, training, evaluation, and postprocessing procedures. The scripts were altered as deemed appropriate by the author. Error metrics were altered.

The data used in this chapter was simulated and the codes written to process the simulation data were written by Emre Kopanoglu. The scripts were used in their original, unaltered form.

The suggestion to forward map Q-matrices to local SAR matrices was put forward by Shaihan Malik. The algorithm used to do so was published in Ref. [132].

This revision was necessitated by the misplacement of the original data due to storage constraints and the learning process associated with data and storage organization. While largely consistent with the original implementation from 2022, several modifications were made to ascertain whether previous errors and subsequent insights could yield a more accurate dataset. These modifications are as follows:

1. The number of slices used for network testing increased from 7 intermittent

slices across a head volume slab to include every slice (140 total) from the shoulders to the apex of the head, including air space above the head

2. In the preprocessing stage, magnitude error was normalized on a per-channel basis, whereas previously, a single normalization factor was applied to the entire dataset

3. Phase data is now jittered during preprocessing rather than using the original discontinuities from the simulation

4. Originally the training, validation, and testing data were all created with reference positions (eg. paired rightward (R) 5mm with R10mm to represent an R5 mm translation is used in the same dataset as center to R5 mm). The testing and validation datasets used in this chapter are center-out, while training data is composed of all possible combinations which yield an R5 mm translation.

5. The configuration of the body models from the virtual population was altered. Originally, the training dataset contained Ella, Fats and Glenn, the validation dataset contained Billie, and the networks were ultimately tested on Duke. The body model configuration in the datasets used for this chapter are Billie, Fats and Duke for training, Glenn for validation, and Ella for testing.

The rationale for Change 1 stems from the educational journey involved in completing a PhD. The current project, based on reference [8], initially adopted 7 discrete slices from a central slab of the head for the testing dataset. While this approach is reasonable for $B_1^+$ map estimations, it is insufficient for SAR assessments. Given that E-fields and, consequently, SAR are three-dimensional, their distributions should be considered from the shoulders to the apex of the head to ensure maximal accuracy for future applications. Local SAR effects in the shoulders and neck also need to be accounted for [53].

Change 2 was implemented following investigations into gridding artifacts, revealing that using individual normalization factors per channel, rather than a single normalization factor for the entire dataset, resulted in lower cGAN estimation errors. This approach was applied in the current chapter because it is still in the interest of the author to discover whether, with improved understanding, the original methodology could be effectively applied in future research.

Change 3 arose after consultation with Alix Plumley who clarified that the term "phase jitter" in the inherited scripts was identified as "random offsets" in Ref. [8].

Change 4 was implemented to enable testing the networks with larger displacements (e.g. R20 mm) by cascading (process described in Ref. [8] and Chapter 6) the networks as necessary.

Change 5 was implemented since Duke's nasal profile exceeded the anatomical bounds of all of the other body models and ultimately caused the evaluation network to produce areas of high intensity error in the nose of testing images. This is a reason to speculate that using 5 body models for this project is overall insufficient, since the anatomical differences between body models are vast and unique. This is discussed further in the following chapters.

## 4.2 The Neural Network Architecture

The neural network used by Plumley et al. [8] (Figure 4.2.1) was largely based on the pix2pix network [83]. Originally, pix2pix was designed for tasks such as creating realistic images from image outlines and translating aerial map images to street map images. Ref. [8] altered the network to accommodate the task of predicting the effects of head motion on $B_1^+$ map data. The work in this chapter used identical networks to the ones described in Ref. [8]. The $B_1^+$ map estimation network architectures are detailed below.

### Generator

The conditional generative adversarial network (cGAN) consisted of a generator and discriminator. As described in Chapter 2, the generator was a U-Net, which derives its name from its graphical representation (Figure 4.2.1): it consists of an encoder which downsamples the inputs into classified feature representations, and a decoder which takes those features and upsamples them into an image identical to the input image using a semantic segmentation mask which allows for pixel-per-pixel image classification [2].

In this case, the encoding layers connected the input and output $B_1^+$ maps, each of which were activated by a ReLU function. The size of the filters in the CNN for magnitude and phase differed (4×4 for magnitude, as it is often associated with feature extraction in intensity-based images, and 8×8 for phase, as phase information tends to have more fine-grained patterns that require larger receptive

fields). It is uncertain why the network performed better with larger filters (bigger receptive field) for phase. It could be speculated that the differences between phase distributions between channels were less obvious and more intricate, therefore demanding finer feature detection. Another possible explanation could be that phase images exhibit higher spatial frequency content, which benefits from increased filter sizes to extract long-range dependencies (ie. parts of the image that are spatially distant from one another).

Magnitude and phase data were normalized using different methods. Normalization helps stabilize training by rescaling features from differing ranges into a defined interval. Phase data, but not magnitude data, responded positively to batch normalization, which standardizes each minibatch during training. This outcome indicates that magnitude data, being non-negative and bounded, may be less affected by scale variations than phase data, which has a circular distribution that requires more careful normalization. The entire cGAN was regularized using dropout layers, which randomly deactivate a percentage of nodes during training to reduce overfitting (a dropout rate of 0.5 was used here). Dropout has been found to improve generalization in deep networks by discouraging the co-adaptation of neurons [133].

### Discriminator

The discriminator was also a CNN made up of five layers and a leaky ReLU instead of a regular ReLU. The only difference between the two is that the former takes into account small negative values when the input is negative or the node is inactive, and allows for a small positive gradient [134]. This helps alleviate the "dying ReLU" problem, where neurons can become permanently inactive during training if the standard ReLU function outputs zero for all negative inputs. This is discussed in further detail in the next chapter.

Each of the networks were optimized over 60 epochs using Adaptive Moment Estimation (Adam), which iteratively adapts the learning rate during gradient descent, allowing for low training cost and high predictive accuracy [135]. Adam is a mix of two other optimization methods (AdaGrad [136] and RMSprop [137]), where the former performs well with sparse data and the latter handles changing learning conditions. This makes training faster and more stable.

The cGAN was further regularized by means of saving the weights resulting from the lowest overall L1 error. By monitoring validation loss, this method ensures

that the model does not overfit to training data while also reducing computational costs by stopping unnecessary iterations.

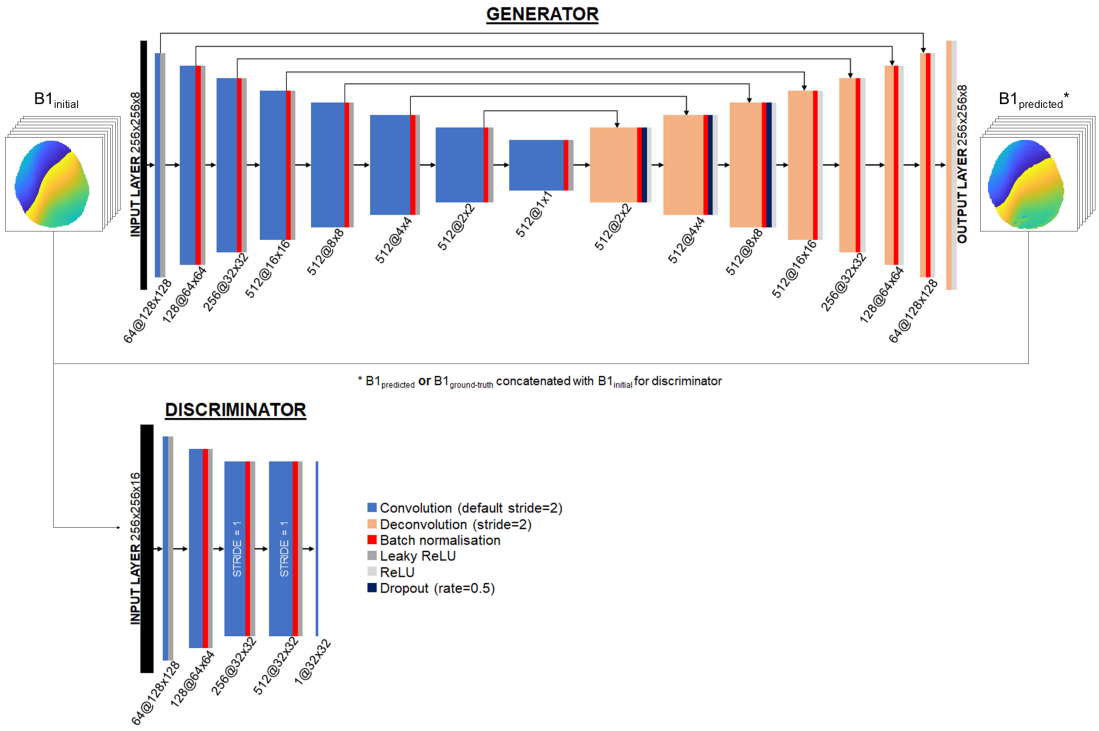


Figure 4.2.1: The cGAN architecture. The top graphic shows the generator network architecture, which consists of a convolutions, joined by skip connections to transposed convolutions. The bottom graphic shows the discriminator network architecture. Colored segments are described in the legend. The blue blocks represent convolution layers, which perform hierarchical feature extraction. The salmon blocks represent deconvolution layers, which expand the feature maps to ultimately create the full resolution image. The red strips represent batch normalization, which stabilizes training. The dark grey and light grey strips represents leaky ReLu and ReLu, respectively. These are both activation functions, which introduce non-linearity, helping the network learn patterns, prevent vanishing gradients, and improve the quality of generated images. The dark blue strip represents the dropout rate which helps prevent overfitting by randomly deactivating neurons during training, ensuring the model generalizes better and does not over-rely on specific features. In this case, the phases of $B_1^+$ maps are shown as paired input and output. Image courtesy of Ref. [8].

## 4.3 E-fields

### 4.3.1 Rationale

The electrical component of the electromagnetic RF field is responsible for tissue heating during MRI scanning [138]. Inconveniently, direct measurements of E-fields are complicated by hardware limitations. The first reason is that the rapidly oscillating RF pulses are high frequency and transient, making it difficult for standard sensors to respond in time to track the changing E-fields [24]. Second, since the MRI environment is sensitive to electromagnetic interference, introducing measurement devices could disrupt the fields required for image acquisition, and the devices themselves might not function correctly because of the magnetic and RF fields already present in the system. The next reason arises from safety concerns. Inserting conductive materials or sensors into the MRI scanner can pose safety risks, like heating or inducing currents that could harm patients or distort imaging results [139]. Finally, since MRI systems operate using a combination of static magnetic fields, low-frequency gradient fields, and RF fields, the cumulative environment makes isolating and measuring E-fields technically demanding [140].

Now that there was an approximately reproducible foundation for successfully predicting the effects of rigid head motion on phases and magnitudes of $B_1^+$ maps, the goal was to repurpose the same pipeline for alternative datasets for MRI safety. If successful, this pipeline would have been expanded to predict E-field variations during scan time, and subsequently be applied as RF pulse design constraints to create safer pTx sequences to control the distribution of E-fields more precisely. E-field predictions could help optimize the phase and amplitude settings of each transmitter to minimize SAR. By fine-tuning these parameters, power deposition can be more evenly distributed and reduce peak local SAR values and guarantee patient safety. Integrating E-field predictions into scanners to dynamically monitor and adjust RF power in real-time could also improve regulatory compliance by making sure that SAR thresholds are not exceeded under varying patient conditions.

E-field predictions could also be used to develop personalized safety protocols for individual patients. By tailoring MRI parameters based on the patient's specific anatomy and demographically-informed tissue properties, the risk of tissue heating could be minimized.

### 4.3.2 Methods

#### E-field Simulations

All body models for this and all subsequent sections in this chapter were simulated identical to the description provided at the end of Chapter 1. All of the data used in this section was already simulated by the lab. The E-fields were extracted by the author using Emre Kopanoglu's pre-existing scripts. Briefly - using finite difference time domain (FDTD) calculations [141], electromagnetic (EM) simulations were performed on Billie, Duke (scaled by 90%), Ella, Fats and Glenn from the Virtual Population (IT'IS, Zurich, Switzerland) [102] at one central position and 29 off-center positions using a generic 8 channel pTx coil at 7 T (295 MHz) in Sim4Life (ZMT, Zurich, Switzerland) [52]. The off-center positions were R 2, 4, 5, 10, and 20 mm and posterior (P) 2, 4, 5 and 10 mm from the center position, as well as all of their possible combinations. In order to ensure consistent voxelization of the tissues while simulating movement, the head coil was shifted in relation to the body. The three-dimensional E-field distributions from each channel and position of individual body models were imported to MATLAB (Mathworks, Natick, MA) [142] for preprocessing.

#### Training, validation and testing dataset preparation

We began the project with the individual x-direction field vector of the E-fields ($E_x$-field), and with a posterior 5mm (P5) displacement. Phase and magnitude input and P5 displaced target pairs were created for the training, validation and testing datasets. A leave-one-out approach was taken for the training, validation and testing dataset creation. In other words, separate body models were used for each dataset: Billie, Duke and Fats for training, Glenn for validation and Ella for testing. The validation and testing datasets were composed of 140 axial slices using only center-out distributions (eg. P0 to P5), while the training dataset was composed of 140 axial slices of both center-out and relative displacements (eg. P5 to P10 as well). To suit the needs of the cGAN, the data was normalized between 0 and 1. For the magnitude data, this was performed channelwise, which means that there were 8 separate normalization factors which were used per channel across slices. The phase data was normalized across the whole dataset and jittered (the whole phase distribution was randomly offset in each slice) in order to prevent the cGAN from overfitting the location of discontinuity and introducing an artificial error. The random phase offset is given by:

$$\theta_s = 2\pi\xi, \quad \xi \sim \mathcal{U}(0,1) \tag{4.3.1}$$

where $\theta_s$ is the phase offset and $\xi$ is a randomly generated number between 0 and 1.

The adjusted phases are:

$$\phi_{\text{init}} = \angle\left(e^{i(\phi_{\text{init}}+\theta_s)}\right) \tag{4.3.2}$$

$$\phi_{\text{tar}} = \angle\left(e^{i(\phi_{\text{tar}}+\phi\theta_s)}\right) \tag{4.3.3}$$

where $\angle()$ denotes the angle of the complex value within the parentheses, *init* is the initial position and *tar* is the target position. The normalized phases are given by:

$$\phi_{\text{init}} = \frac{\text{mod}\ (\phi_{\text{init}} + 2\pi, 2\pi)}{2\pi} \tag{4.3.4}$$

$$\phi_{\text{tar}} = \frac{\text{mod}\ (\phi_{\text{tar}} + 2\pi, 2\pi)}{2\pi} \tag{4.3.5}$$

where *mod* is the modulo function which returns the remainder of a division operation between two values. These equations normalize angular values ($\phi$) to a range of 0 to 1. They ensure that any given angle remains within a single cycle of a full rotation using the modulo operation.

This yielded a training dataset of 5,040 initial-position and displaced target pairs for each, magnitude and phase. The cGANs used for this section were identical to those in Ref. [8], but were implemented using slightly different Python packages due to the reproducibility issues described in the previous chapter. The scripts were run on a Linux CentOS 7 using a NVIDIA v100 DGX GPU.

### Magnitude network estimation error calculation

The magnitude results were evaluated by calculating the normalized root mean square error (nRMSE) between the centered and ground-truth $E_x$-field matrices and comparing it to the nRMSE between the network-estimated and ground-truth images:

$$\text{nRMSE} = 100 \times \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(GT_i - y_i\right)^2}}{\frac{1}{N}\sum_{i=1}^{N} GT_i} \tag{4.3.6}$$

where $N$ is the number of pixels, $GT_i$ is ground-truth and $y_i$ is either the centered or network-estimated $E_x$-field.

The magnitude network estimation error was post-processed using two different approaches: one with a Gaussian smoothing filter applied and one without. A Gaussian smoothing filter is a low-pass filter that reduces noise and smooths variations in an image by convolving it with a Gaussian function (Figure 4.3.1, effectively blurring the image in a controlled manner while preserving important structures and suppressing high-frequency noise. In the context of cGANs, applying a Gaussian smoothing filter can improve image output quality by reducing noise, enhancing spatial consistency, and mitigating artifacts including pixelation or checkerboard patterns caused by transposed convolutions. This filtering process results in cleaner, more visually appealing images with improved perceptual quality [9]. A Gaussian filter was initially applied to follow the existing protocol established by Ref. [8].

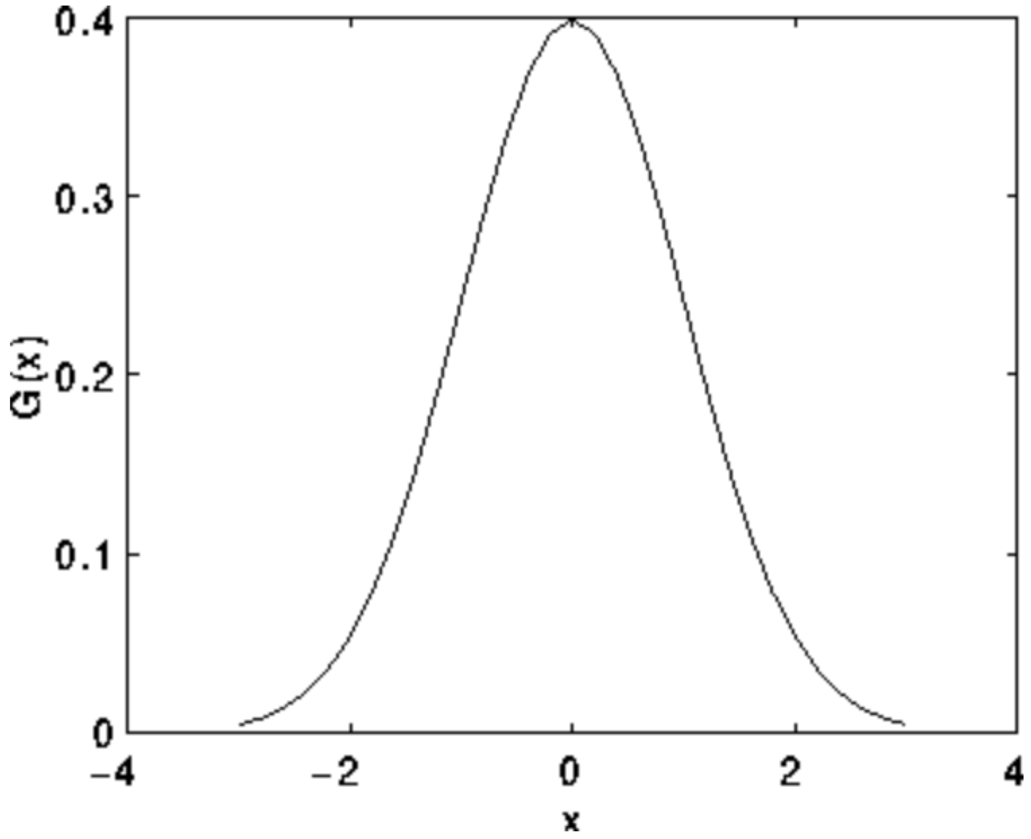


Figure 4.3.1: 1D Gaussian filter with mean = 0 and standard deviation of the distribution = 1. Figure retrieved from Ref. [9].

The error images seen in the figures displaying maximum intensity projections are not nRMSE, but rather reflect cases of maximum point error (worse case scenario for the differences between channels). This is the case for all of the following sections in this chapter.

### Phase network estimation error calculation

To maintain dimension, the phase error was calculated by the overall difference between the ground-truth and centered or network estimated $E_x$-fields, across slices and channels:

$$\epsilon_\phi = \frac{180}{\pi} \times |\angle e^{(i(\angle GT - \angle y))}| \tag{4.3.7}$$

where $\epsilon_\phi$ is the error of the phase values, $\angle GT$ is the phase of the ground-truth $E_x$-field, and $\angle y$ is either the phase of the initial-position or phase of the network estimated E-field.

The error images in Figure 4.3.8 were produced by plotting the absolute value of the phase error in Eq. 4.3.7.

### 4.3.3 Results

Each cGAN trained for approximately eight hours.

#### Magnitude

The magnitude motion-induced nRMSE was 10.7% while the network estimation image which was smoothed with a Gaussian filter was 17.3%. This network estimation nRMSE decreased to 15.2% after removing the Gaussian filter from the postprocessing pipeline. When the Gaussian filter was removed, the error improved because the network estimation in Figures 4.3.2 and 4.3.3 appeared smoother than the ground-truth image, since the ground-truth $E_x$-field patterns are highly detailed (Figures 4.3.4 and 4.3.5).

Despite the change, the estimation error was not lower than motion error (Figures 4.3.4 and 4.3.5).

The figures are generated to first display the greatest discrepancy between the ground-truth and centered position E-field distributions (GT - centered), followed by the greatest discrepancy between the ground-truth and network estimation images (GT - NE).

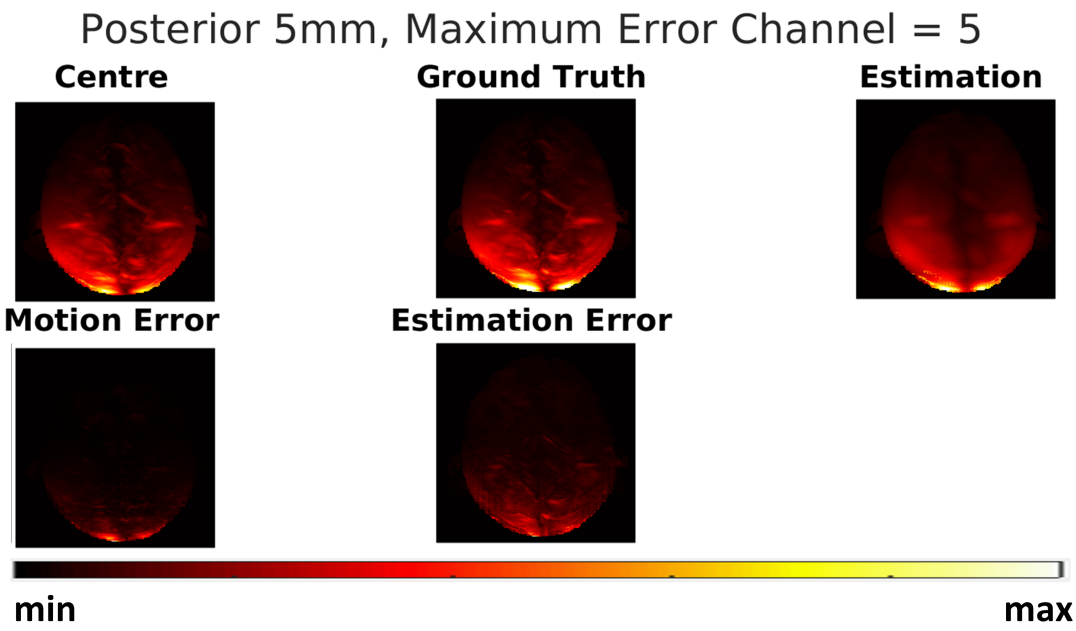


Figure 4.3.2: Error of $E_x$-field **magnitudes** exhibited as a maximum intensity projection along the z-axis (all 140 slices) for channel 5 which yielded the worst error between the ground-truth and centered images. These slices **have been smoothed** with a 5x5 Gaussian filter.

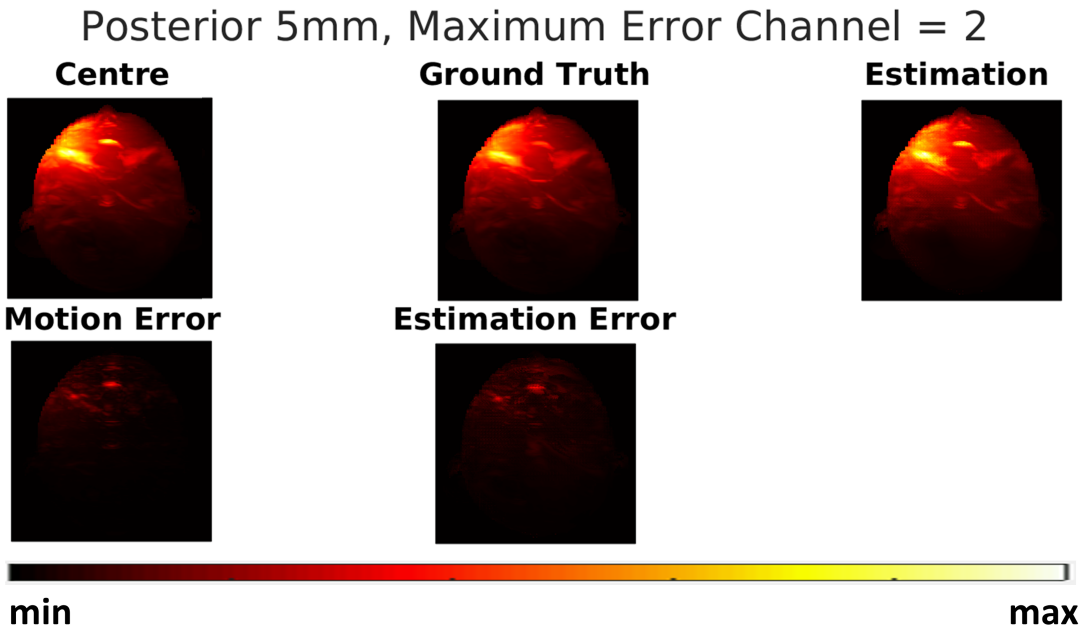


Figure 4.3.3: Error of $E_x$-field **magnitudes** exhibited as a maximum intensity projection along the z-axis (all 140 slices) for channel 2 which yielded the worst error between the ground-truth and network estimated images. These slices **have been smoothed** with a 5x5 Gaussian filter.

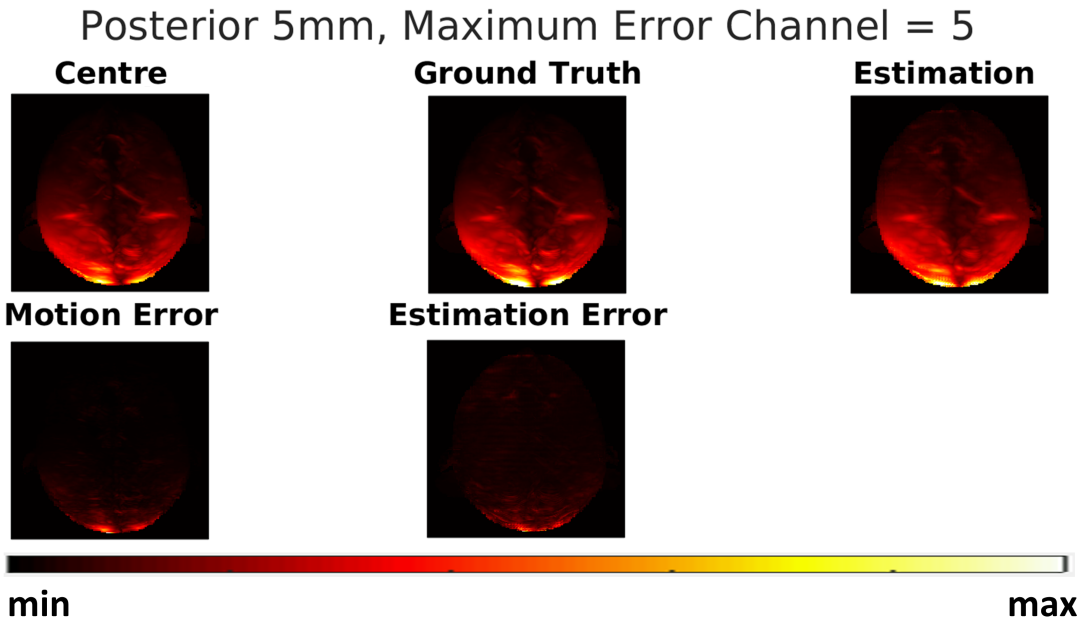


Figure 4.3.4: Error of $E_x$-field **magnitudes** exhibited as a maximum intensity projection along the z axis (all 140 slices) for channel 5 which yielded the worst error between the ground-truth and centered images. These slices **have not been smoothed** with a Gaussian filter.

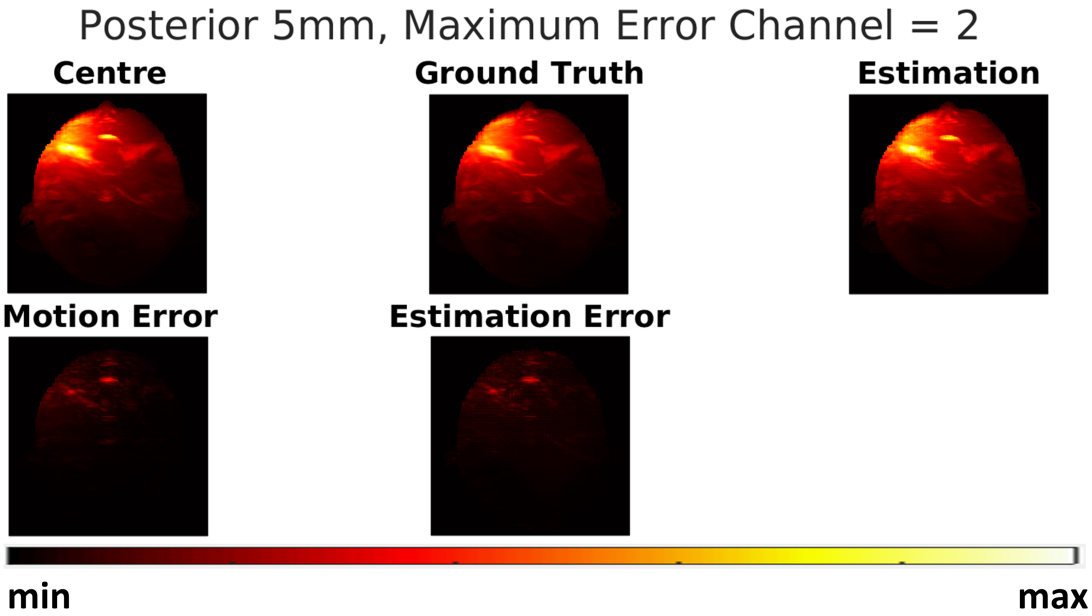


Figure 4.3.5: Error of $E_x$-field **magnitudes** exhibited as a maximum intensity projection along the z axis (all 140 slices) for channel 2 which yielded the worst error between the ground-truth and network estimated images. These slices **have not been smoothed** with a Gaussian filter.

The center and ground-truth images in Figures 4.3.2, 4.3.3, 4.3.4, and 4.3.5 exhibit smooth transitions but clear spatial variations.

Figures 4.3.6 and 4.3.7 show the nRMSE per pTx channel for the Gaussian smoothed and not Gaussian smoothed $E_x$-fields, respectively. Importantly, motion error in pTx channel 1 is higher than network estimation error. It is notable that the maximum intensity projection difference images and nRMSE indicated different channels which yield the maximum error.

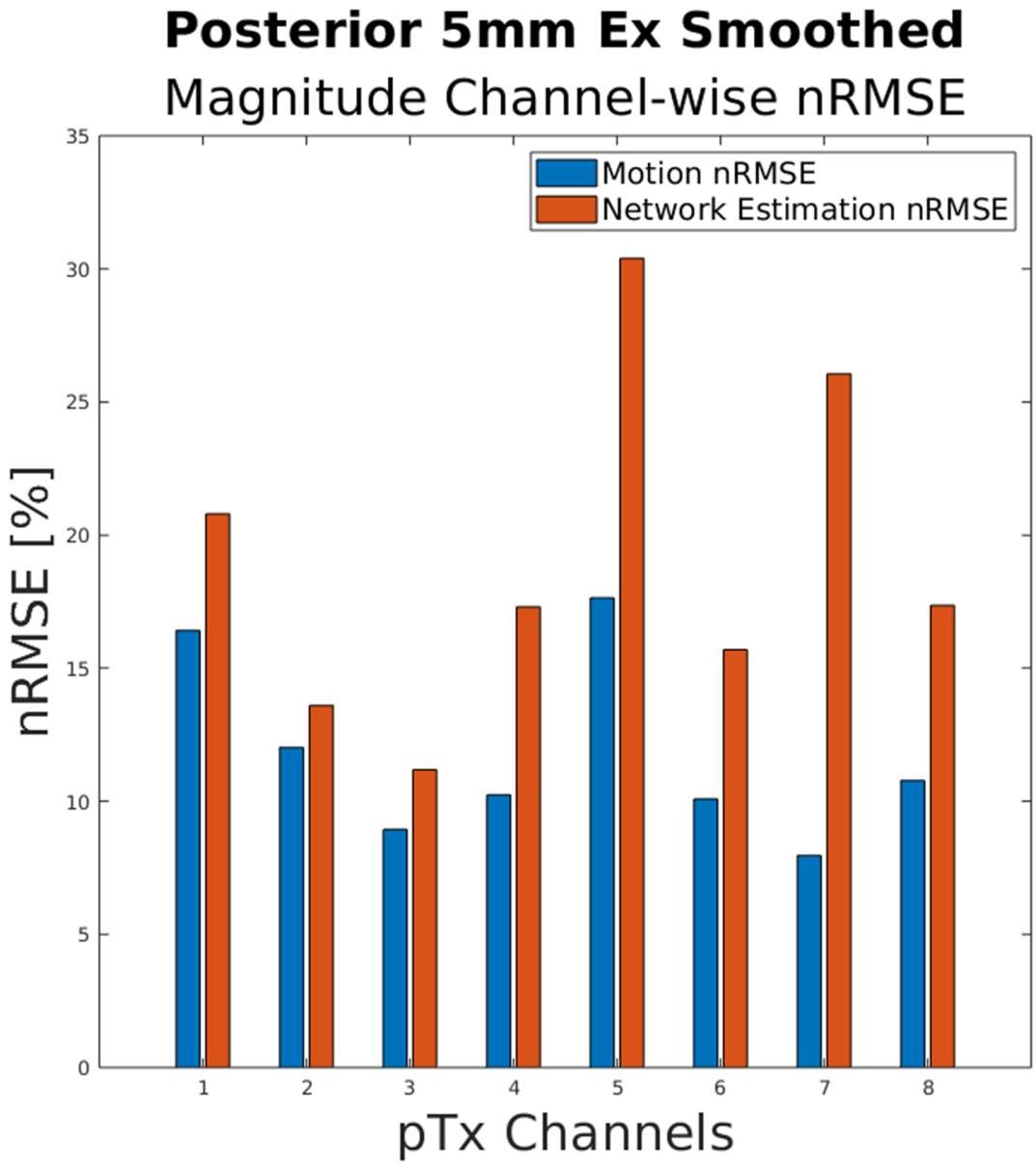


Figure 4.3.6: Motion and Network Estimated nRMSE per pTx channel for the magnitudes of $E_x$-fields. The Network Estimated fields **have been smoothed** by a 5x5 Gaussian filter. The blue bars are default-motion error, while the orange bars are network-estimation error.

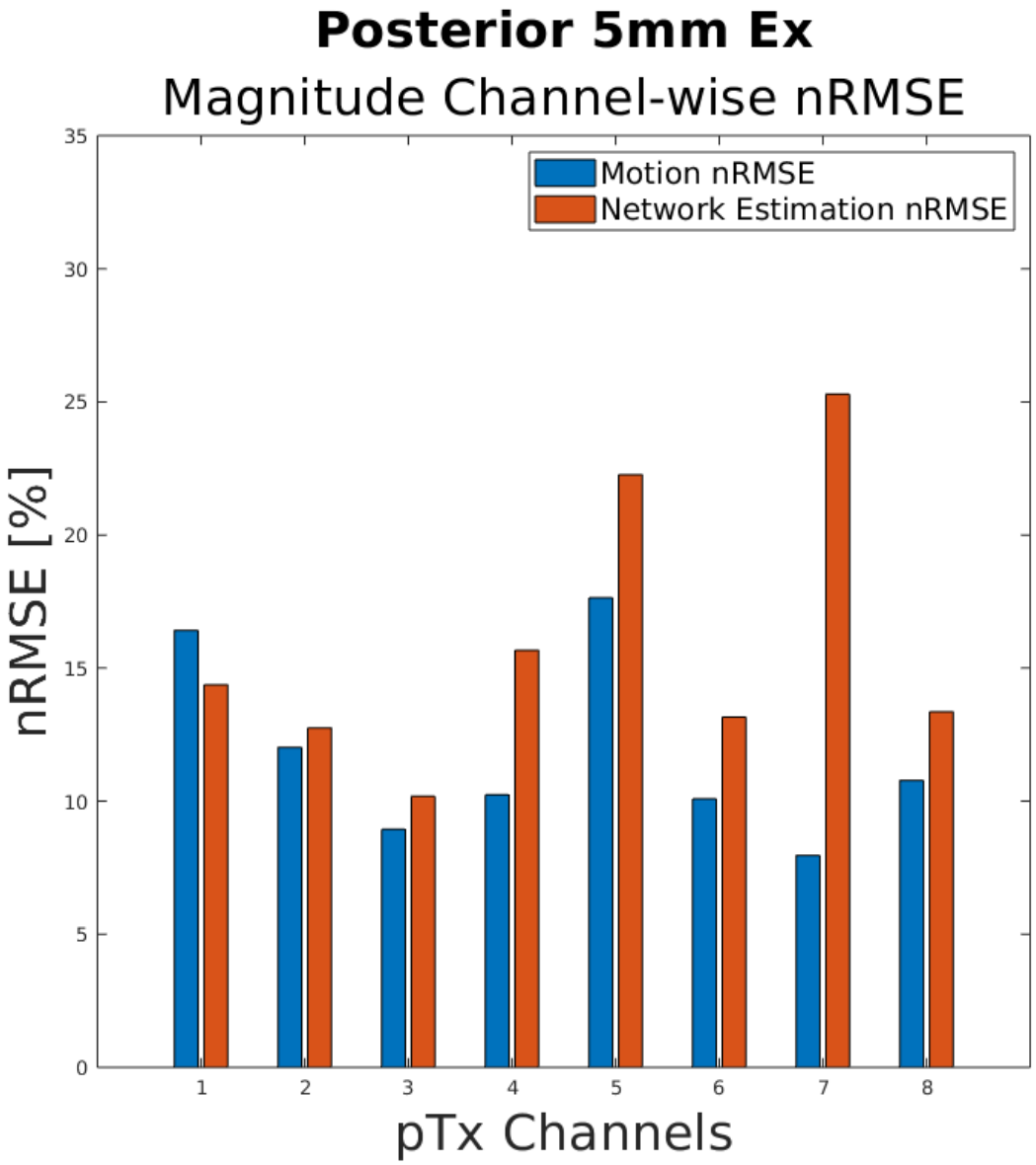


Figure 4.3.7: Motion and Network Estimated nRMSE per pTx channel for the magnitudes of $E_x$-fields. The network estimated fields **have not been smoothed** by a 5x5 Gaussian filter. The blue bars are default-motion error, while the orange bars are network-estimation error.

### Phase

The mean network estimation error was greater (2.3°) than the mean motion error (1.6°), across slices and channels. Channel 1 yielded the worst error. (Figure 4.3.8). Channel 1 saw the worst error for both network estimation and default motion error.

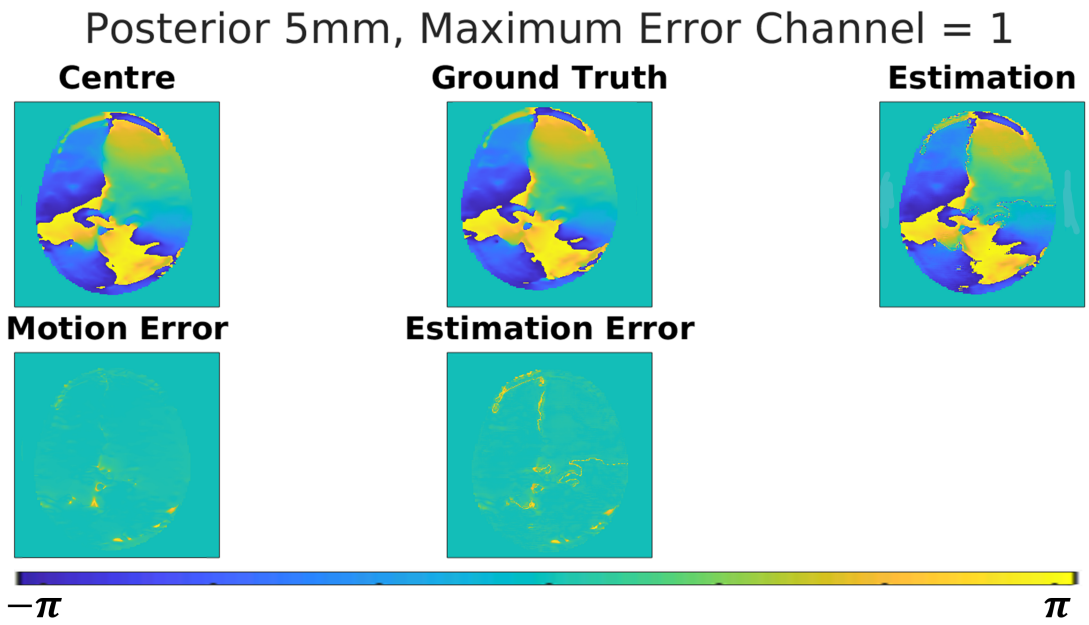


Figure 4.3.8: Error of $E_x$-field phases exhibited at a central slice for channel 1 which yielded the worst error between the ground-truth and both, network estimated and centered images. The scale is between $-\pi$ and $\pi$.

Figure 4.3.9 shows that the network estimation error, calculated using Equation 4.3.7, was greater than the default motion error in all channels. While channel 1 had the worst network estimation error, channel 5 had the worst motion error.

### Complex $E_x$ fields

The nRMSE of the magnitude and phase data was also calculated after combining the two parts into the full complex data. By combining the magnitude and phase parts together, it is less likely that the error metrics will penalize phase wrap boundaries. The full complex value accounts for the interaction between magnitude and phase, thereby providing a more complete evaluation of the data. The motion-induced nRMSE of the $E_x$-field magnitudes which were not smoothed with a Gaussian filter reunited with the $E_x$-field phases were 11.6% (channel-wise errors presented in Figure 4.3.11), while the network-estimated nRMSE was 21.8%. When smoothing was performed, the network-estimated nRMSE rose to 23.3% (channel-wise errors presented in Figure 4.3.10),.

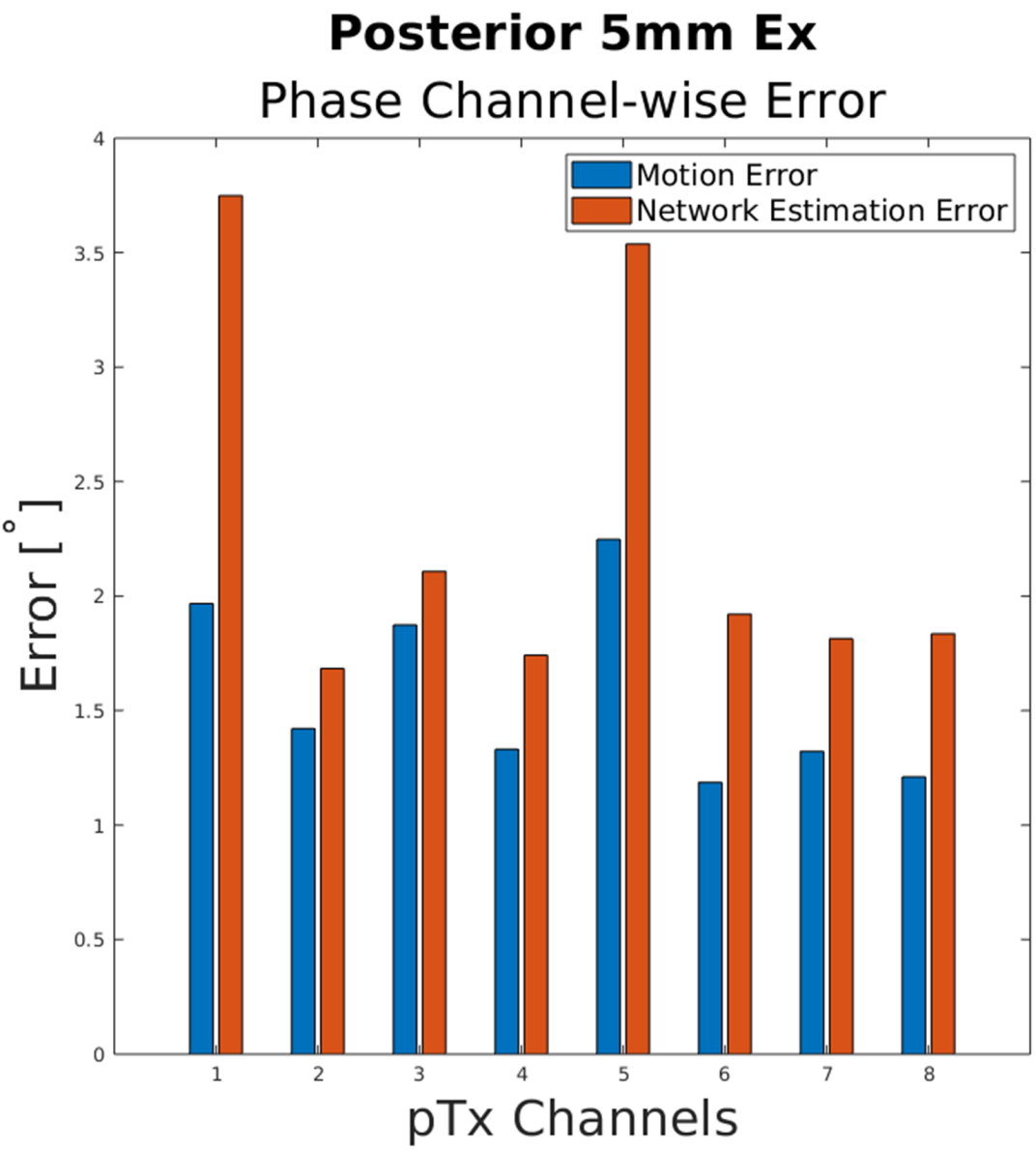


Figure 4.3.9: Motion and Network Estimated ° error per pTx channel for the phases of $E_x$-fields. The blue bars are default-motion error, while the orange bars are network-estimation error.

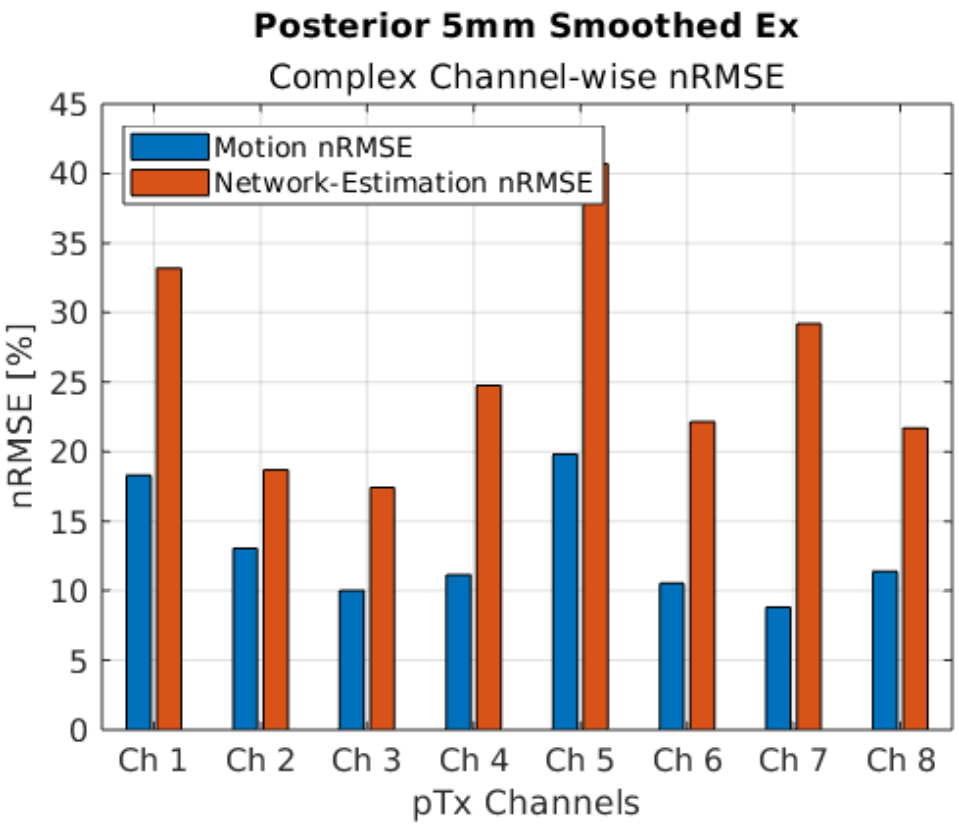


Figure 4.3.10: Motion and Network Estimated nRMSE error per pTx channel for the smooth complex $E_x$-fields. The blue bars are default-motion error, while the orange bars are network-estimation error.

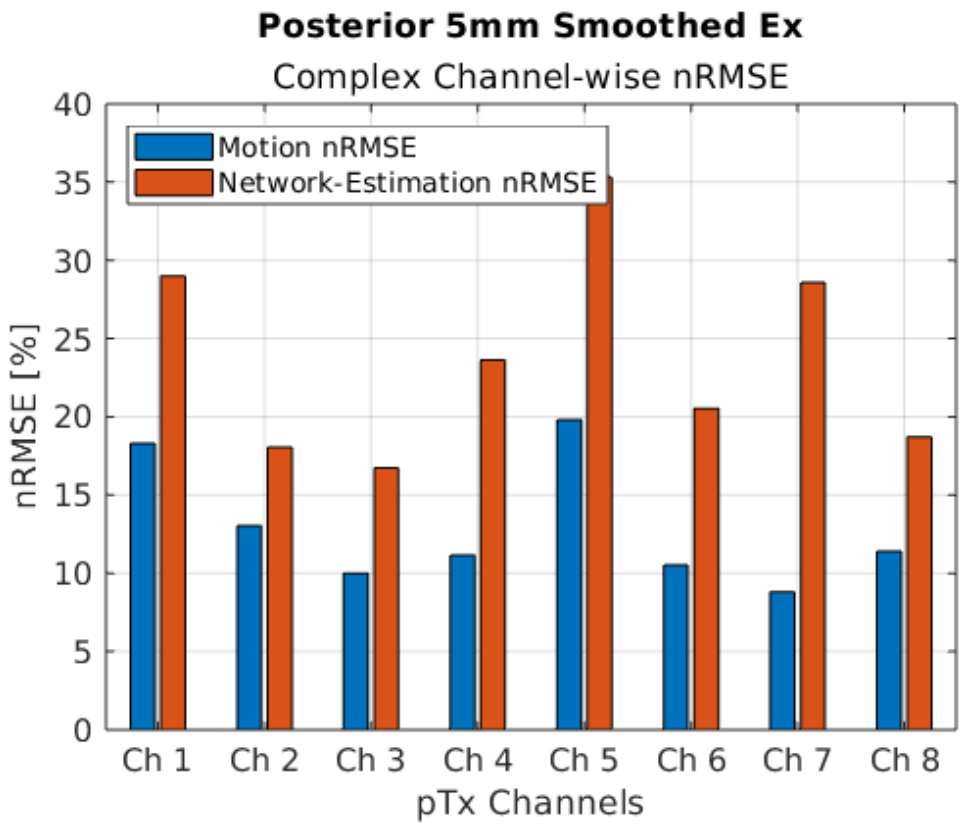


Figure 4.3.11: Motion and Network Estimated nRMSE error per pTx channel for the unsmoothed complex $E_x$-fields. The blue bars are default-motion error, while the orange bars are network-estimation error.

### 4.3.4 Section Discussion

Overall, it is evident that the preliminary $E_x$-field predictions were problematic, as neither the magnitude nor phase predictions were accurate or useful for the ultimate goal of creating a pipeline for MRI safety. While the network was able to produce structurally plausible outputs, the quantitative analysis revealed that both magnitude and phase predictions were not sufficiently accurate compared to the baseline motion-induced error. In particular, the network-estimated $E_x$-field magnitudes exhibited higher nRMSE than the motion-displaced fields, and even more so after post-processing with a Gaussian filter. Both data types encompassed intricate details which are likely problematic as inputs for image-to-image translation tasks. The spatial complexity of E-fields, driven by tissue heterogeneity and coil interactions, likely exceeds the representational capacity of the model in its current form.

With respect to the magnitude investigations, the fact that the worst case channel varies between the GT-centered and GT - NE cases indicates that in the case of $E_x$-field prediction, the cGAN does not yield the worst performance when the original motion error (GT - centered) is highest. The variation in these distributions is also expected due to tissue-dependent dielectric properties. For example, areas with higher water content (CSF, gray matter) have different permittivity and conductivity, affecting field penetration. The Motion Error images (Figures 4.3.2, 4.3.3, 4.3.4, and 4.3.5) confirm that motion alters the interaction between RF fields and tissue, leading to potential misregistration of energy deposition, if the current coil safety model does not account for head motion or positional variations. It is evident that the network estimation nRMSE is lower when not smoothing with a Gaussian filter. Importantly, motion error in pTx channel 1 is higher than network estimation error, indicating that the cGAN offered improvement in that one domain.

Combining the magnitudes and phases together into a complex variable did not paint a more favorable picture for the network estimation accuracy.

Consequently, the decision to pivot toward alternative target datasets, possibly with lower spatial complexity or better-behaved distributions, was well-justified.

## 4.4 Q-Matrices

### 4.4.1 Rationale

Predicting the effects of motion on 10g-averaged Q-matrices was the next logical step, given the ineffectiveness of applying individual E-fields. Q-matrices represent the RF power contribution of each pTx channel and can serve as constraints in RF pulse design. In terms of magnitude, Q-matrix images are less detailed, making their variations easier for neural networks to learn. Conversely, the phase of Q-matrices is complicated by singularities, posing challenges for neural networks. Singularities in phase arise from the multi-channel RF interactions, where abrupt phase discontinuities occur.

In the previous section about E-fields, a direct deep learning approach would have required running separate networks for different degrees of motion and motion types, phase, and magnitude, as well as for each individual field vector (x, y and z). This approach may have required extensive training time, potentially spanning months of individualized tweaking and testing, since it could have been the case that different network parameters were more suitable to different field vectors. Overall, using E-fields could have eventually necessitated at least 24 networks (unless of course networks trained on the x field vector could be tested with the y direction). It also had the potential to introduce inherent error buildup arising from multiple networks, each of which contribute their own inherent error, yielding elements of a single datatype. Stacking multiple predictions from independent networks can lead to error propagation, compounding inaccuracies over multiple processing steps (an effect of this is investigated in Chapter 6). In other words, this could have resulted in a combinatorial explosion of required models.

If successful, the Q-matrix prediction pipeline could have been combined with work which developed subject-specific models for single-position personalized SAR predictions [86]. Combining Q-matrix predictions with personalized SAR models could enable real-time safety monitoring, reducing the risk of RF-induced heating while optimizing transmission efficiency.

For clarity in all future references to Q-matrices in this thesis, Figure 4.4.1 illustrates how each single-value channel index corresponds to a specific channel combination $Q_{mn}$.

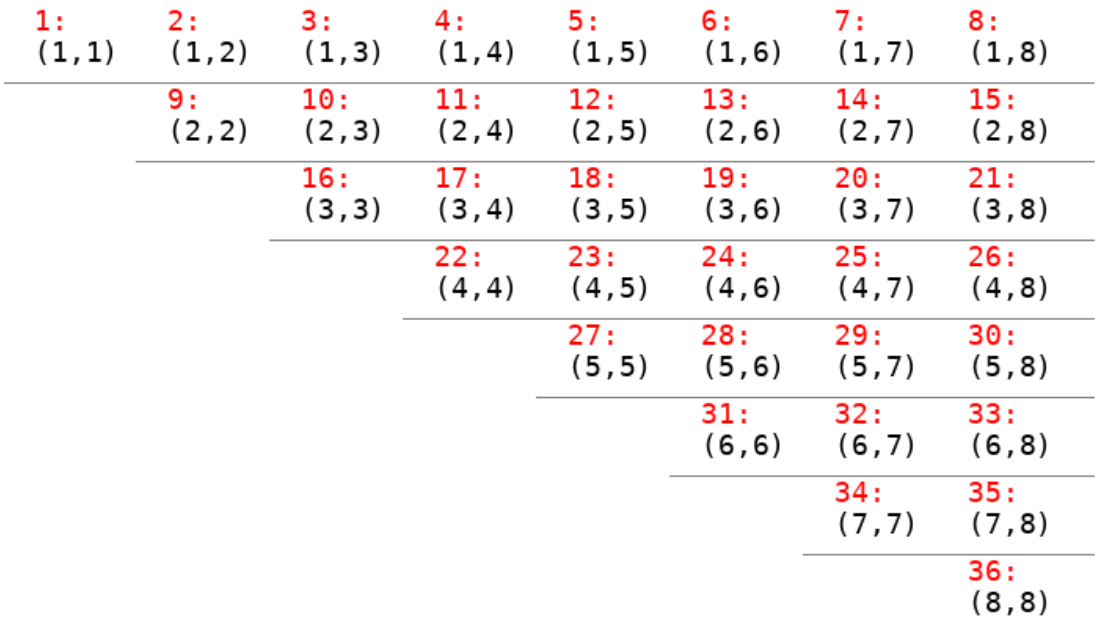


Figure 4.4.1: Diagram of Q-matrix channel indices (in red), organized by transmit channel combinations (in parentheses).

### 4.4.2 Methods - Magnitude and Phase

To calculate Q-matrices, the three-dimensional E-field distributions and current densities from each channel and position of individual body models were imported to MATLAB for preprocessing. The data were first masked using tissue data in terms of its mass density, electrical conductivity and averaging volume. The E-fields were then combined by the current density's Hermitian conjugates along individual field vectors to yield local 10g averaged Q-matrices, using

$$\mathbf{Q}_{m,n}(\mathbf{r}) = \frac{1}{2\rho(\mathbf{r})}\left[\mathbf{J}^{H}_{x,n}(\mathbf{r})\ \mathbf{E}_{x,m}(\mathbf{r}) + \mathbf{J}^{H}_{y,n}(\mathbf{r})\mathbf{E}_{y,m}(\mathbf{r}) + \mathbf{J}^{H}_{z,n}(\mathbf{r})\mathbf{E}_{z,m}(\mathbf{r})\right] \quad (4.4.1)$$

where $\rho(\mathbf{r})$ is the tissue conductivity, $\mathbf{E}_x, \mathbf{E}_y, \mathbf{E}_z$ and $\mathbf{J}_x, \mathbf{J}_y, \mathbf{J}_z$ are the 3D E-field and current density distributions, $m$ and $n$ represent the pTx coil's channel indices, and $H$ is the Hermitian conjugate. Because mutual interactions between two channels (i.e., $\mathbf{Q}_{m,n}$ and $\mathbf{Q}_{n,m}$ where $m, n$ are channels) are complex conjugates, 28 entries of the 8x8 Q-matrix are non-unique for each voxel. To prevent network overfitting, these 28 entries were discarded. From this point onward, the discretized version of $\mathbf{Q}_{m,n}(r)$ will be denoted as $Q_{m,n}$. All of the following investigations in the remainder of this thesis which use Q-matrices are conducted

on specifically 10g-averaged Q-matrices, which hereafter are referred to as simply "Q-matrices".

The cGAN used to predict the Q-matrices was nearly identical to that used for the $E_x$ fields, except for the changes made to accommodate 36 channels instead of 8, and therefore data sizes of 256x256x36. Since for the network to run smoothly, the number of filters must be a multiple of the number of channels, the magnitude networks used 144 filters, and the phase networks used 72 filters (changed from 128 and 64, respectively).

The results were post-processed and evaluated in the same way as for the Gaussian-smoothed $E_x$-fields.

### 4.4.3 Results - Magnitude and Phase

Each cGAN trained for approximately 14 hours.

#### Magnitude

Figures 4.4.2 and 4.4.3 show that with all of the changes applied to the network-prediction pipeline since its first implementation in 2022, the magnitude network estimation error was far lower (7.1%) than the motion error (18.8%). The center and ground-truth images display obvious error, while the ground-truth and cGAN estimation images are nearly indistinguishable. This is reflected in the two error images, displayed in the bottom half of the figures.

Figure 4.4.3 shows that the worst case network estimation error is notably low, while Figure 4.4.2 shows how the cGAN improves prominent default motion error.

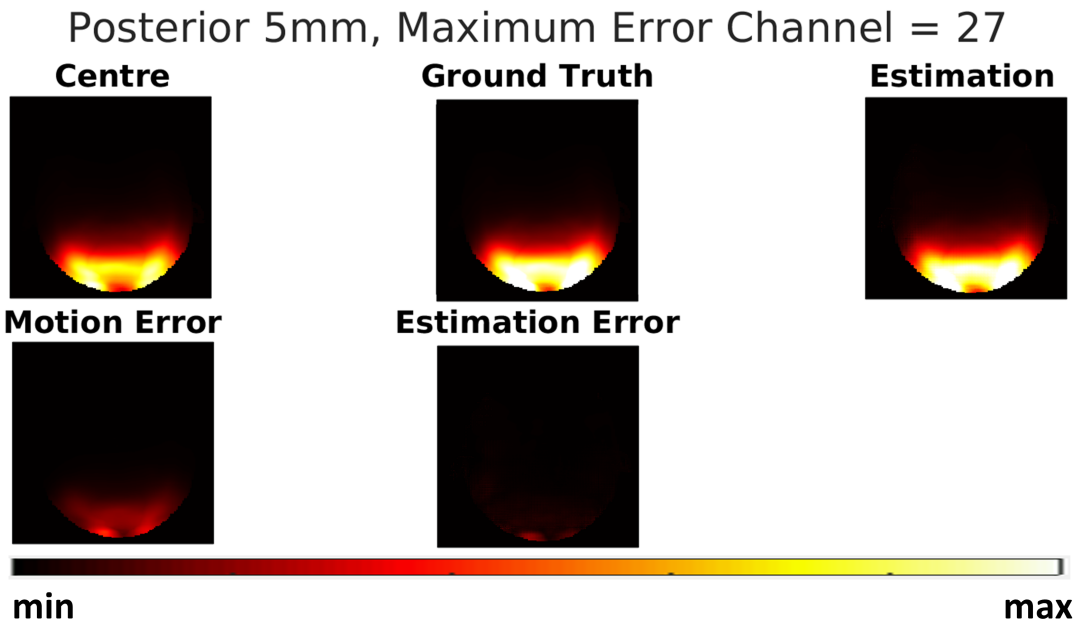


Figure 4.4.2: Error of Q-matrix magnitudes. This figure shows a central slice for channel 27 ($Q_{5,5}$) which yielded the worst error. In the final panel, it is clear that the network estimation error is far lower than motion-induced error.

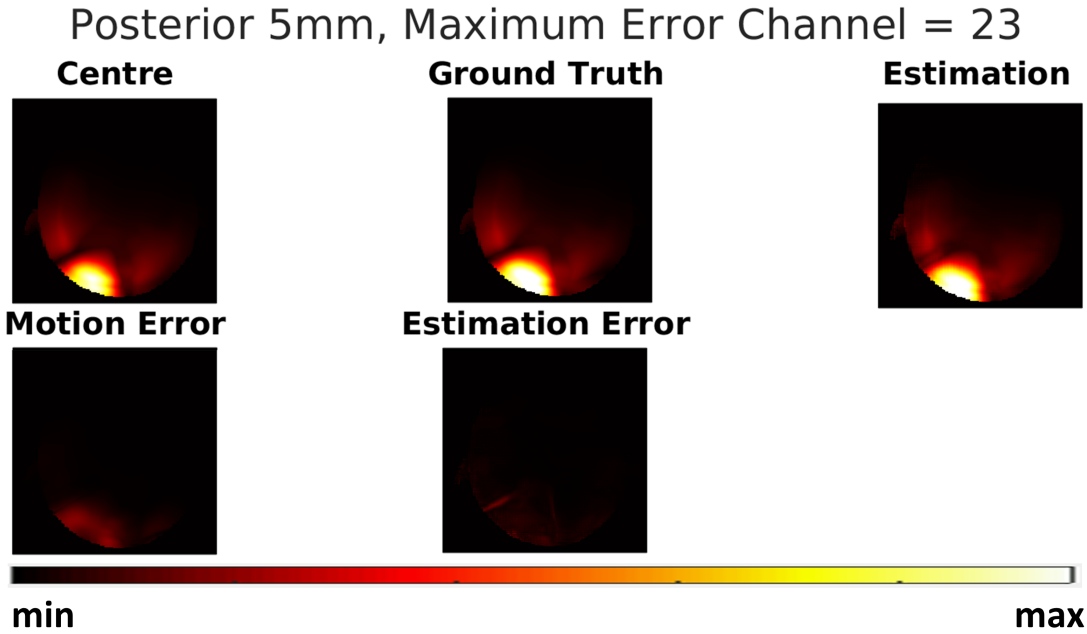


Figure 4.4.3: Error of Q-matrix magnitudes. This figure shows a central slice for channel 23 ($Q_{4,5}$) which yielded the worst error. In the final panel, it is clear that the network estimation error is far lower than motion-induced error.

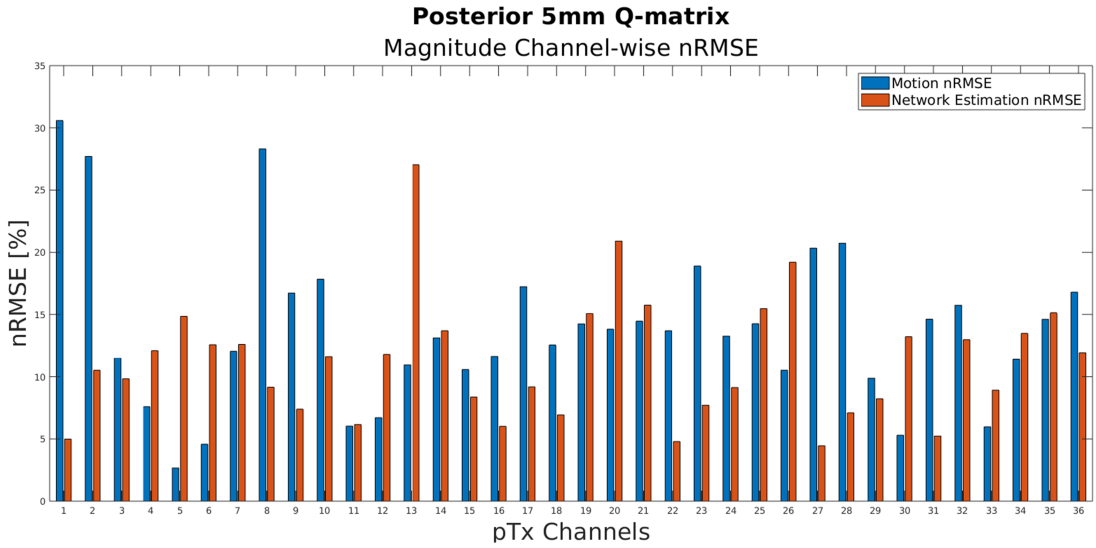


Figure 4.4.4: Network estimation and motion error of Q-matrix magnitudes per pTx channel. Network estimation is lower than motion error in 53% of the channels. The blue bars are default-motion error, while the orange bars are network-estimation error.

Figure 4.4.4 shows that over half of the channels experience lower network estimated nRMSE than default motion nRMSE.

### Phase

To depict Figure 4.4.6 and the following phases of Q-matrices, the data arising from same-channel interactions (eg. $Q_{1,1}$ or $Q_{5,5}$) were excluded because they are stored power. This is because same-channel interaction images appear as shown in Figure 4.4.5.

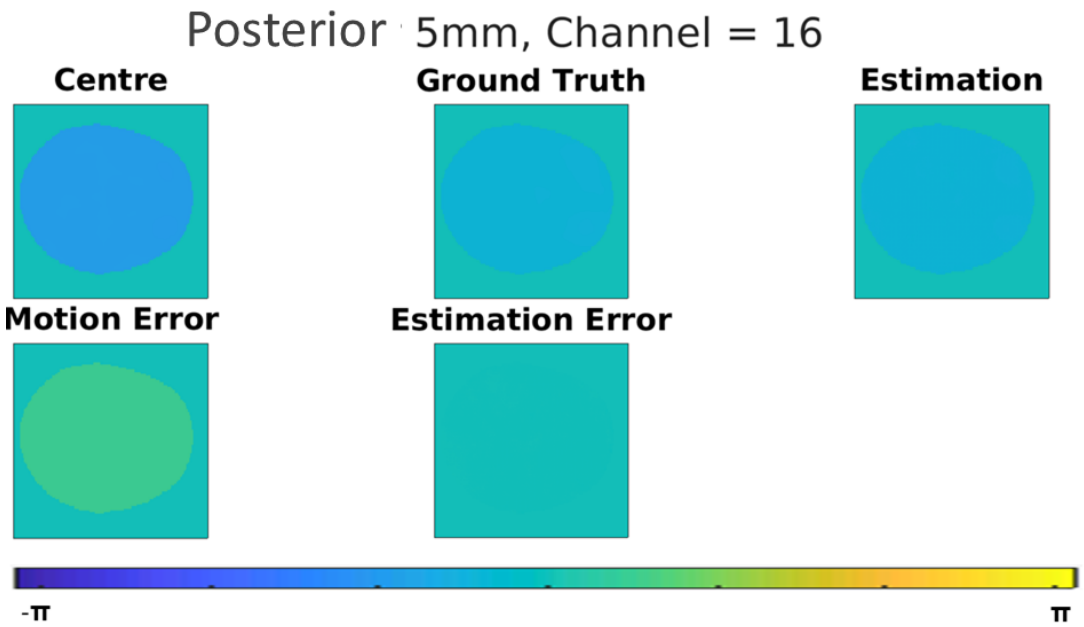


Figure 4.4.5: Same channel interaction effects on the phases of Q-matrices. The example channel depicted here is $Q_{3,3}$.

The phase network estimation error was 3°, while the motion error was 1.8°. The network estimation inaccuracy is depicted in Figures 4.4.6 and 4.4.7.

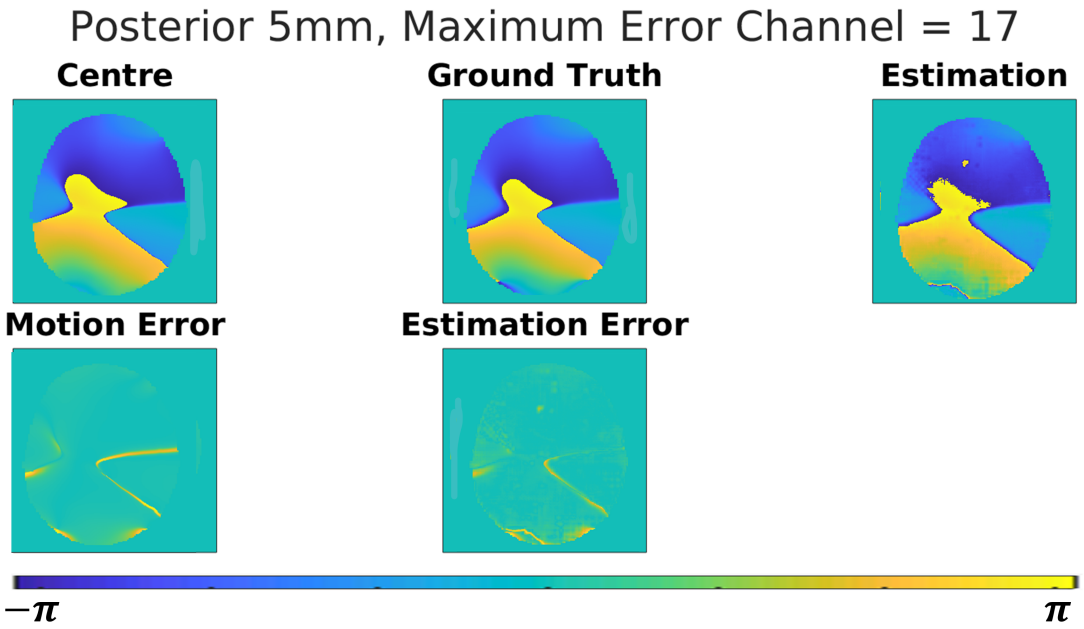


Figure 4.4.6: Error of Q-matrix phases. This figure shows a central slice for channel 17 ($Q_{3,7}$) which yielded the worst error between the ground-truth and centered images. The scale is between $-\pi$ and $\pi$. In the final panel, it is clear that the network estimation error is only marginally higher than motion-induced error.

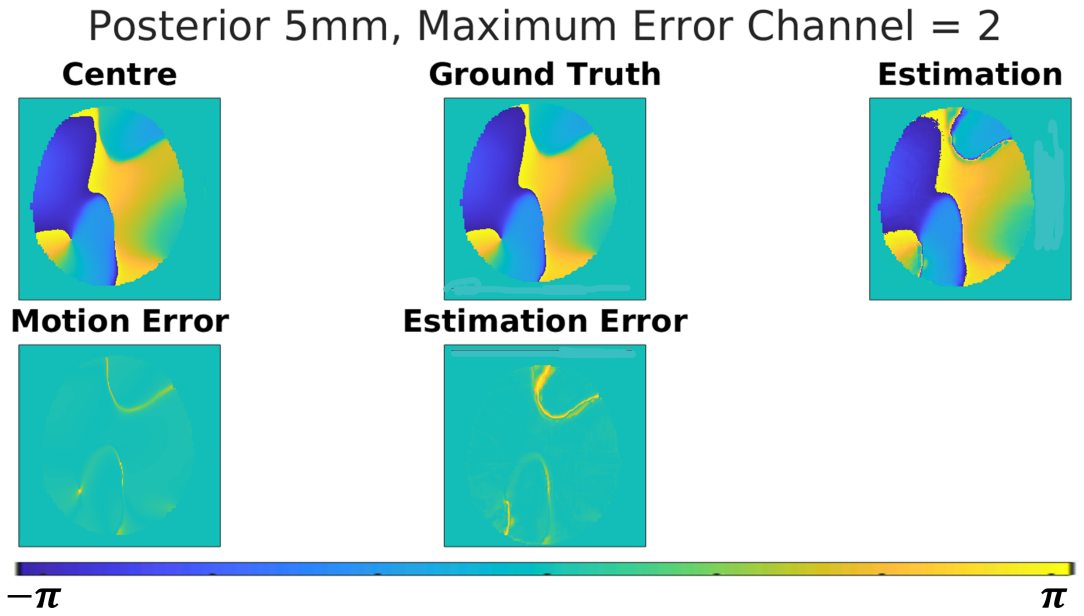


Figure 4.4.7: Error of Q-matrix phases. This figure shows a central slice for channel 2 ($Q_{1,3}$) which yielded the worst error between the ground-truth and network estimated images. The scale is between $-\pi$ and $\pi$. In the final panel, it is clear that the network estimation error is notably higher than motion-induced error.

Figure 4.4.8 shows that network estimation error was higher than motion error for all channels.

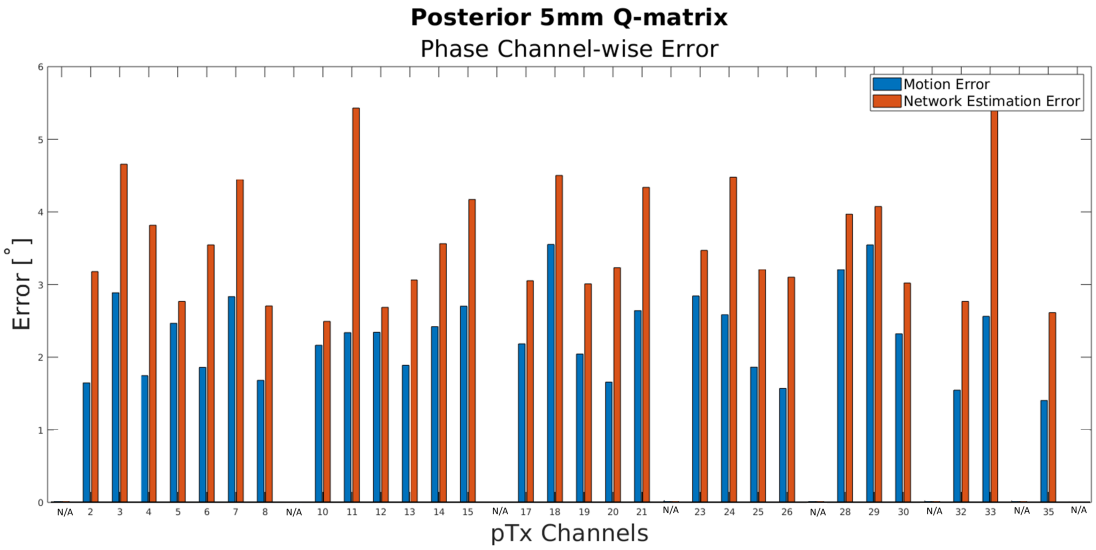


Figure 4.4.8: Network estimation and motion error of Q-matrix phases per pTx channel. Where channel $m$ and channel $n$ are identical, there is no motion error. This is excluded from error ° calculation results. The rest of the channels show higher network estimation error than motion error. The blue bars are default-motion error, while the orange bars are network-estimation error.

### Complex Magnitude-Phase Q-matrices

The motion-induced nRMSE of the magnitude-phase combined complex values Q-matrix was 19.4%, while the network-estimated nRMSE was 16%. Figure 4.4.9 shows channel-wise nRMSE values.

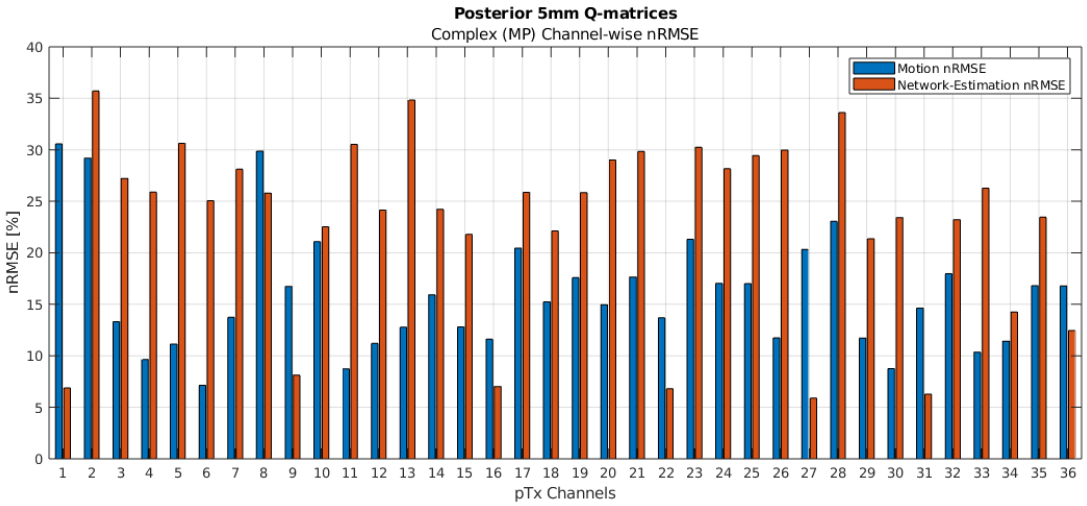


Figure 4.4.9: Network estimation and motion error of Q-matrix phases per pTx channel. Where channel $m$ and channel $n$ are identical, there is no motion error. This is excluded from error ° calculation results. The rest of the channels show higher network estimation error than motion error. The blue bars are default-motion error, while the orange bars are network-estimation error.

### 4.4.4 Section Discussion - Magnitude and Phase

The Q-matrix predictions were an improvement to the previous E-field attempts, particularly with respect to the magnitude component. The network consistently outperformed the motion-induced baseline in over half of the pTx channels, and in worst-case slices, the discrepancy between network estimation and ground-truth was minimal. This success can be attributed to the inherent smoothness and lower spatial complexity of Q-matrix magnitude maps, which likely made them more amenable to image-to-image translation by a cGAN. The reduction in high-frequency content and the relative uniformity of these images mitigated the difficulties previously encountered with sharp spatial gradients and fine structural detail in $E_x$-fields.

In contrast, the phase component of the Q-matrices continued to pose an issue for the cGAN. Despite preprocessing strategies like phase jittering to mitigate overfitting and account for phase wrapping, the network estimation error remained higher than the baseline motion error across all channels. This reiterates a pattern observed in the $E_x$-field phase predictions and suggests that standard generative models struggle with accurately modeling discontinuities and singularities in phase data. It could simply be the case that cGANs perform better on

images with smooth variations and fail when it comes to more abrupt data, like sharp phase transitions that arise from wrap boundaries. Despite modifications to the network architecture (e.g., increased filters, adjusted channels), the issue appears to be fundamental to the data representation rather than just network design.

Notably, phase unwrapping was not applied, as correspondence with the first author of Ref. [8] confirmed that such techniques were not useful when designing the $B_1^+$ map prediction pipeline [143].

The continued difficulty with phase estimations motivated the next investigations, though retrospectively it is arguable whether 5 mm of head motion was sufficient to conclude whether the phase networks are successful, since the small translation does not seem to affect phase distributions substantially as seen in Figures 4.4.6 and 4.4.7. Additionally, the phase wrap boundaries which were exaggerated by the cGAN are artificial and would most likely not affect the ultimate pTx-based evaluation described in Chapter 6.

The complex valued data showed promise in that the errors caused by the phase wrap boundaries were likely no longer exaggerated. The promising results here show that future researchers working with phase data of Q-matrices need not be discouraged by mediocre network performance on phase data, since the estimated complex data is still improved by the proposed method.

Of note, the initial iterations of this project (with alternative data preparation methods described in the foreword) yielded network-estimated magnitude and phase images with far greater error than for the motion-induced magnitude and phase images. The exact values are unavailable. Complex matrices were not calculated at the time.

### 4.4.5 Methods - Imaginary and Real

Because the original magnitude, phase, and phase-difference pipelines were unsuccessful, a real-imaginary split of the complex SAR matrices was also attempted. A real-imaginary approach was successfully implemented in [39], though it was unsuccessful in the trial-and-error period which led to the completed project which used a magnitude-phase split described in [8] [143]. The results are calculated in the same way as the magnitude Q-matrix nRMSEs.

The real and imaginary datasets were created with the `real` and `imag` functions in Matlab. The pre- and post-processing pipelines were nearly identical to the Q-matrix magnitude pipeline, except that the channelwise normalizations were calculated by

$$\tilde{Z}_c = \frac{Z_c - Z_{c_{min}}}{Z_{c_{max}} - Z_{c_{min}}} \tag{4.4.2}$$

where $\tilde{Z}$ is either normalized real or normalized imaginary data, $Z$ is either real or imaginary data, $c$ is a specified $m, n$ channel combination, $min$ is the channel-specific minimum value, and $max$ is the channel-specific maximum value.

### 4.4.6 Results - Imaginary and Real

The networks trained for approximately 14 hours.

#### Imaginary

Figure 4.4.10 shows that network estimation error of the imaginary Q-matrices is greater (nRMSE = 41.5%) than motion error (nRMSE = 24.7%). The relative dimness of the estimation image and its subsequent error profile clearly show that the network-prediction accuracy was low.

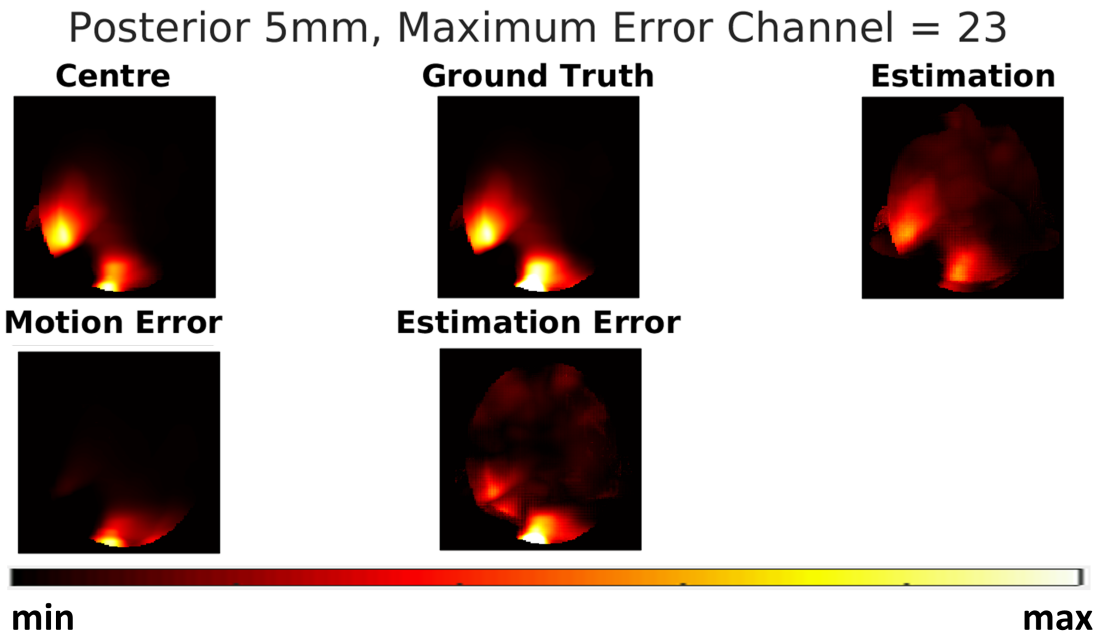


Figure 4.4.10: Error of imaginary component of Q-matrices. This figure shows a maximum intensity projection along the z axis for channel 23 ($Q_{4,5}$) which yielded the worst error. In the final panel, it is clear that the network estimation error is greater than motion-induced error. The same channel yields the greatest error between the ground-truth and centered images and ground-truth and network estimation image.

Figure 4.4.11 shows that 21.4% the channels see similar or lower network estimation error than motion error.

Notably, diagonal Q-matrix terms are intrinsically real valued, therefore any imaginary component is because of numerical error or stored power, and is completely irrelevant to SAR.

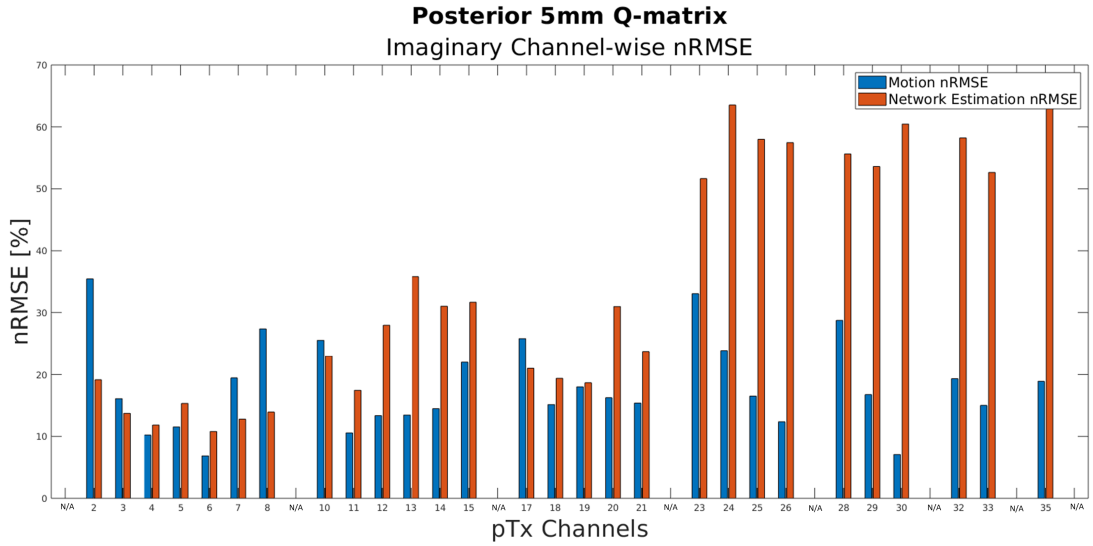


Figure 4.4.11: Network estimation and motion error of imaginary Q-matrices per pTx channel. Where channel $m$ and channel $n$ are identical, the error values are not included. They are real and pertain to stored power or numerical error, and are therefore irrelevant to SAR. The blue bars are default-motion error, while the orange bars are network-estimation error.

### Real

The network estimation nRMSE is 32.4% while the motion nRMSE is 19%, meaning that the cGAN estimations introduced a 70.53% error increase. The proximity of nRMSE values is reflected in Figure 4.4.12, which shows worst case motion error. A gridded artifact manifested in the top right-hand corner of the network estimation error image. Figure 4.4.13, which is generated from worst case network estimation error, shows flawed cGAN estimation ability.

Figures 4.4.12 and 4.4.13 both display a checkered artifact in the top right-hand corner and localized high intensity error on the left-hand edge of the network estimation images.

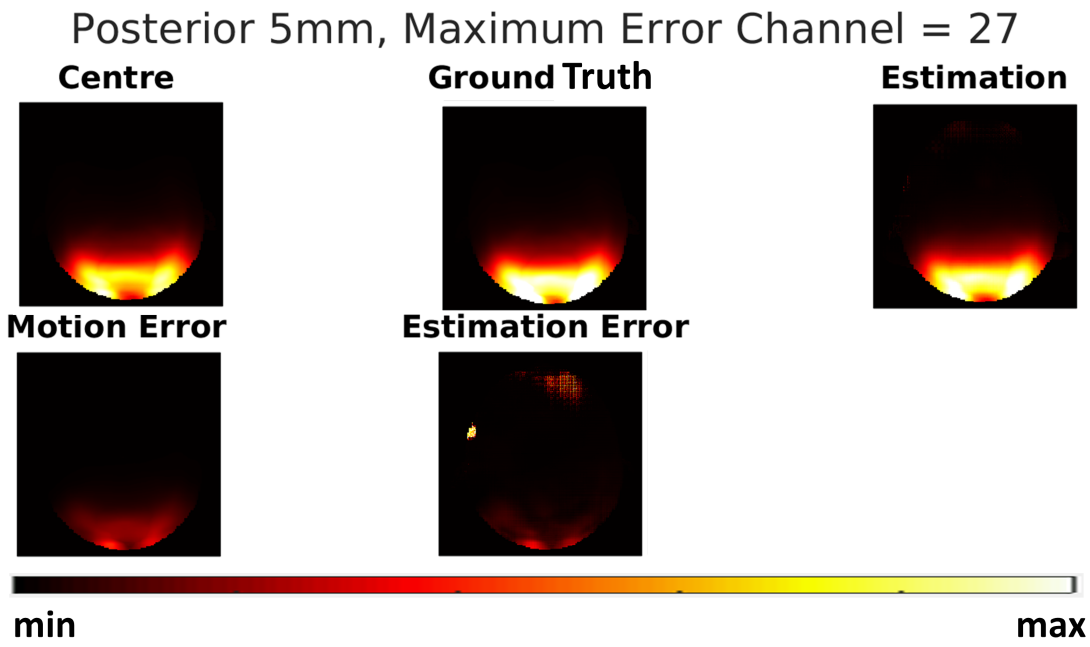


Figure 4.4.12: Error of real component of Q-matrices. This figure shows a maximum intensity projection along the z axis for channel 27 ($Q_{5,5}$) which yielded the worst error between the ground-truth and centered images.

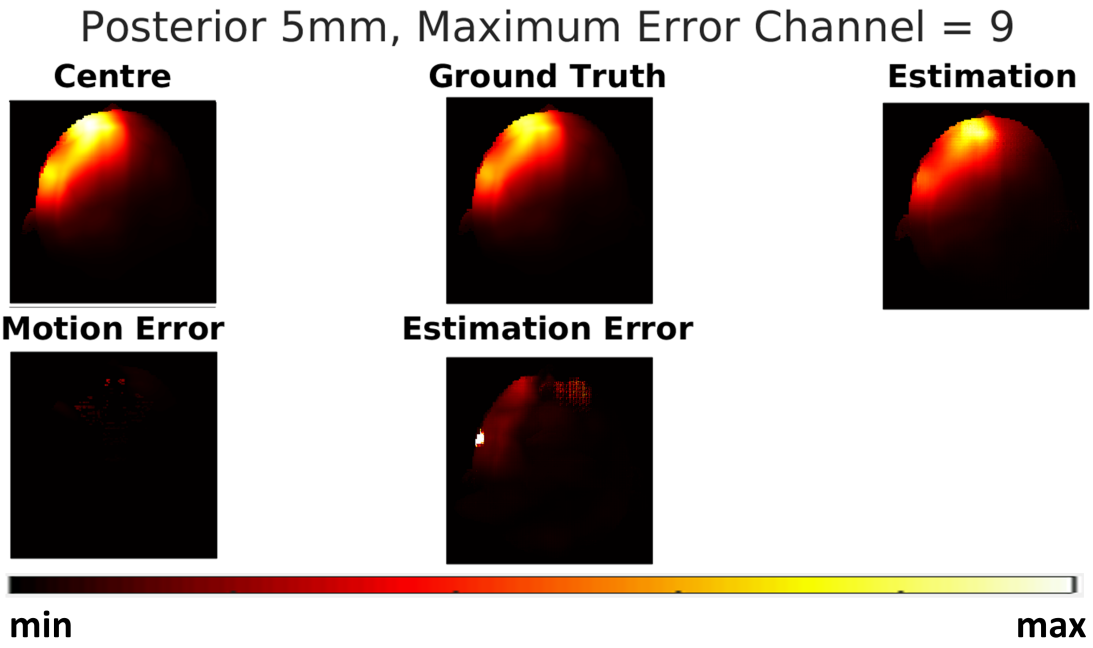


Figure 4.4.13: Error of real component of Q-matrices. This figure shows a maximum intensity projection along the z axis for channel 9 ($Q_{2,2}$) which yielded the worst error between the ground-truth and network estimated images.

Figure 4.4.14 shows that network estimation error of real Q-matrices far exceeds

motion error in most cases, with the exception of $Q_{1,1}$ and $Q_{5,5}$.

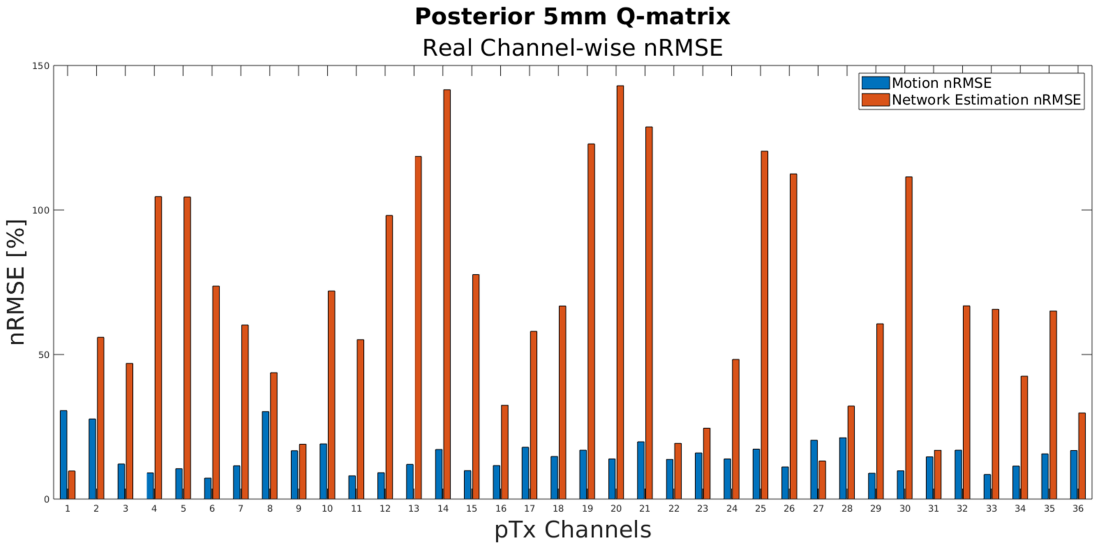


Figure 4.4.14: Network estimation and motion error of real Q-matrices per pTx channel. The blue bars are default-motion error, while the orange bars are network-estimation error.

Of note, and in relation to the previous chapter describing Python's erratic unreliability, the real Q-matrix network failed to run over many attempts. The spyder GUI command window printed the error:

```
DataLossError: truncated record at 2623547994'
failed with Read less bytes than requested
[Op:IteratorGetNext]
```

This was solved by rewriting the preprocessed training pairs from MATLAB into TFRecord format, and failing that, restarting the pipeline from the position-pairing preprocessing step. Repeated three times, these time consuming remedies were unsuccessful. After a one-month pause in implementing the real Q-matrix networks and making no further alternations, the network code was run again, this time successfully. Since no changes were made and only a pause between implementations was taken, it is unclear why the error initially prevented the networks from running. It is not unreasonable to suspect environment-related issues or inherent cGAN volatility.

### Complex Real-Imaginary Q-matrices

The motion-affected nRMSE of complex Q-matrices composed of re-combined real and imaginary parts was 19.4%, while network estimation was 33.2%. Figure 4.4.15 shows the channel-wise nRMSE for this case.

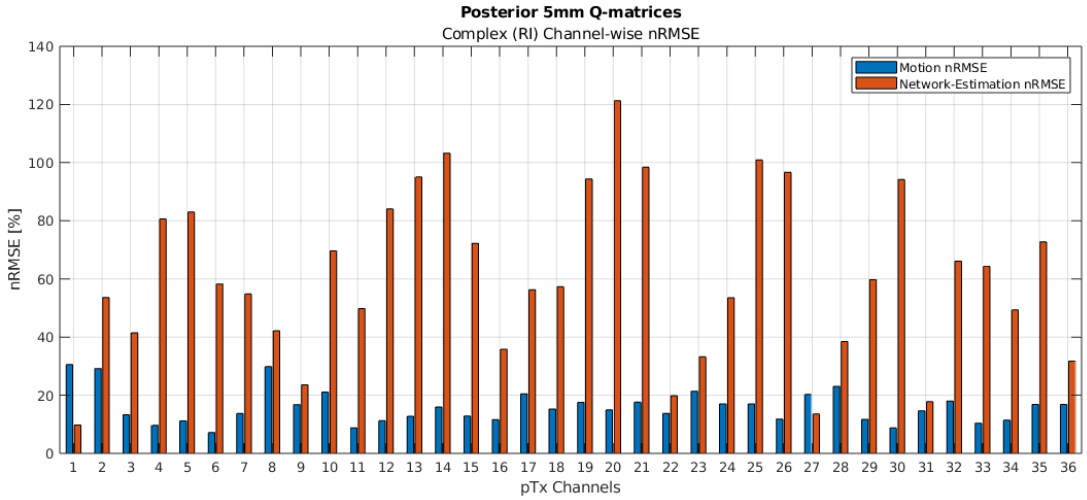


Figure 4.4.15: Network estimation and motion error of complex (real-imaginary) Q-matrices per pTx channel. The blue bars are default-motion error, while the orange bars are network-estimation error.

## 4.4.7 Section Discussion - Imaginary and Real

The real and imaginary Q-matrix results followed a similar pattern to the earlier phase and magnitude findings. Overall, neither the real nor imaginary data yielded network estimation results that were clearly better than motion error across the full channel range, with few exceptions.

The imaginary component proved difficult to estimate, though some channel errors were improved by networks. The network estimation nRMSE was substantially higher than the default motion error. Moreover, the error image in Figure 4.4.10 suggests that the network struggled with the dead space in the top right-hand corner of the images. This resembles the same issues seen in the Q-matrix phase estimation images (Figures 4.4.6 and 4.4.7), where the phase wrap boundaries also posed issues for prediction accuracy, again suggesting that the cGAN is better suited to predicting images containing smoother transitions.

The real component had similar results. Although the overall network estimation error still exceeded the motion error, a few diagonal terms, namely $Q_{1,1}$ and

$Q_{5,5}$, did show better network performance. This suggests that in some limited cases, the cGAN was able to capture useful patterns. However, there were clear issues across the dataset. The top right-hand corner of the prediction images regularly displayed a grid-like artifact (which will be discussed in detail in Chapter 5), which was reflected in the error maps. Additionally, some slices showed a consistent localized spike on the left-hand edge, near the ear region, which could indicate an artifact in the simulation data itself or a feature that the network mislearned. These artifacts may stem from poor neural network hyperparameter choices, including filter size or regularization strength. The localized high intensity error on the right-hand edge may be something ingrained in the simulated images used for training or validation, since it is located near the ear, which contains air. This could likely be reduced with more tuning or by eliminating training slices containing air in the ear region. These are likely responsible for the unfavorable quantitative network estimation metrics.

A separate but noteworthy issue was the unexpected TensorFlow error that repeatedly prevented the real Q-matrix network from training. Despite multiple attempts to regenerate TFRecord files, rewrite the dataset from MATLAB, and restart the full data pipeline, the issue persisted for weeks. After a one-month pause with no changes made to the pipeline, the network eventually ran successfully. Since no code or system updates were made during that time, the cause of the error remains unknown. This kind of random behavior impedes projects requiring long-term reproducibility.

Overall, the real and imaginary Q-matrix methods did not significantly outperform earlier approaches, even when calculating the results based on more complete complex data where real and imaginary Q-matrices were re-united. Real data remains a poor candidate for the safety pipeline. Imaginary data showed slightly better results and may be more suitable for future model improvements if network tuning is prioritized and potential artifacts in the training data are resolved. Like with the phase-based methods, these findings show that cGANs are better suited to learning smooth, structured data distributions.

## 4.5 Intermediary Local SAR

### 4.5.1 Rationale

At the time it was clear that the successes of using the magnitude networks were being dampened by the failures in implementing the phase networks. A different approach was proposed by Ref. [144], which, in short, consisted of applying an algorithm from [132] to the 10g averaged Q-matrices to calculate the power absorption consequence (termed here as intermediary local SAR or ilSAR) of a given RF design. ilSAR is real (non-complex) data, which feature-wise resembles Q-matrix magnitude data. The benefit of this is that only one network needs to be trained, and the overall promise of a higher success rate as suggested by the Q-matrix magnitude networks. To reiterate for clarity, the ilSAR is derived from specifically 10g-averaged Q-matrices. All mentions of ilSAR in the entirety of this thesis are of those derived from exclusively 10g-averaged Q-matrices.

### 4.5.2 Methods

The local SAR distribution in voxel $r$ was calculated from Q-matrices using:

$$SAR[r] = Re\{\mathbf{w}^T \cdot \ \mathbf{Q}[r] \cdot \mathbf{w}\} \tag{4.5.1}$$

where $w$ denotes channel coefficients, the superscript $T$ denotes complex-conjugate transpose, $\mathbf{Q}[r]$ denotes the complex-valued NxN Q-matrix that characterizes the interaction of the coil elements in voxel $r, N$ is the number of coil elements, and $Re\{\cdot\}$ extracts the real-valued component of its input.

To avoid estimating the complex-valued entries of the Q-matrices, these matrices can be characterized by a series of $N^2$ real-valued projections, ie. intermediate local SAR (ilSAR) distributions. This method was proposed by Zhu et al. [132] to enable estimation of Q-matrices from (real-valued) empirical power measurements. For an $N$-channel coil, the first $N$ terms are single-coil distributions given by:

$$ilSAR_m^{c_m}[r] = Re\{\mathbf{w}^T \cdot \ \mathbf{Q}[r] \cdot \mathbf{w}\} \tag{4.5.2}$$

where

$$\mathbf{w}[k] = \begin{cases} c_m, & k = m \\ 0, & else \end{cases} \tag{4.5.3}$$

The other $N(N-1)$ ilSAR distributions are when only two coils are on and the rest are off:

$$ilSAR_{m,n}^{c_m,c_n}[r] = Re\left\{\mathbf{w}^T \cdot \mathbf{Q}[r] \cdot \mathbf{w}\right\} \tag{4.5.4}$$

where

$$\mathbf{w}[k] = \begin{cases} c_m, & k = m \\ c_n \text{ or } \widetilde{c_n}, & k = n \\ 0, & else \end{cases} \tag{4.5.5}$$

and $c_m$ and $c_n$ are coefficients of channels $m$ and $n$. $c_m = \sqrt{2}$ was used for single channel ilSAR while two pairs of channel coefficients $(c_m, c_n)$=(1,1) and $(c_m, \widetilde{c_n})$=(1,$i$) were used for two-coil interactions.

All of the ilSAR matrices were processed the same way as the Q-matrix magnitudes in the previous sections, except that there were 64 normalization factors, since the real ilSAR maps contained power absorption consequences from all 64 unique channel interactions. The exact same cGAN used for the magnitude $E_x$-fields was applied here, with the exception of the number of channels in the data that had to be accounted for while training and validating (64 for ilSAR vs 36 for Q-matrices).

For clarity in all future references to ilSAR in this thesis, Figure 4.5.1 illustrates how each single-value channel index corresponds to a specific channel combination $Q_{m,n}$.

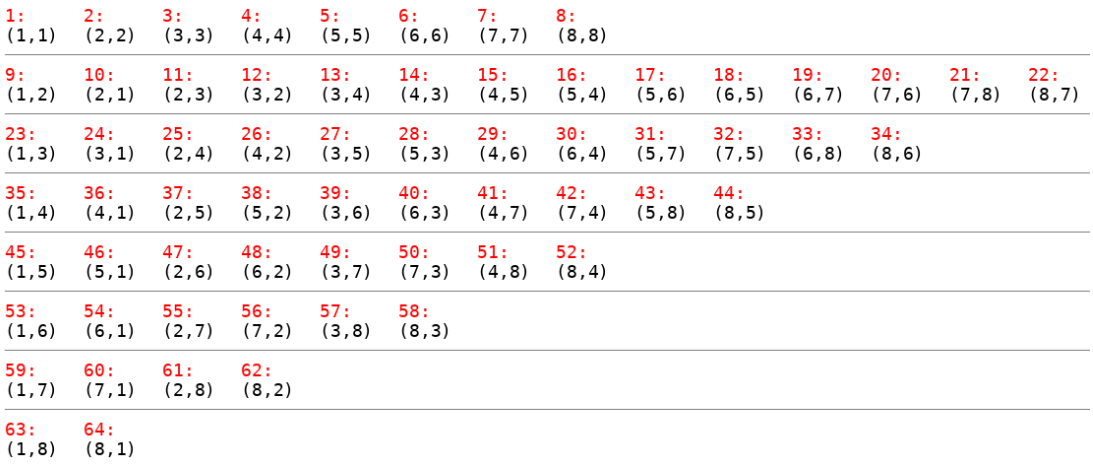


Figure 4.5.1: Diagram of ilSAR channel indices (in red), organized by transmit channel combinations (in parentheses).

### 4.5.3 Results

The cGAN trained for approximately 20 hours.

Figures 4.5.2 and 4.5.3 show that the initial attempt using ilSAR was unsuccessful. Both error maps reveal a distinct checkerboard pattern (as also observed in the real Q-matrix estimations). This grid-like structure appears consistently in several channels and aligns with the "gridding artifact" noted later in Figure 4.5.5. While the nRMSE between the centered and ground-truth image was 17.8%, the network estimation error was 55.7%.

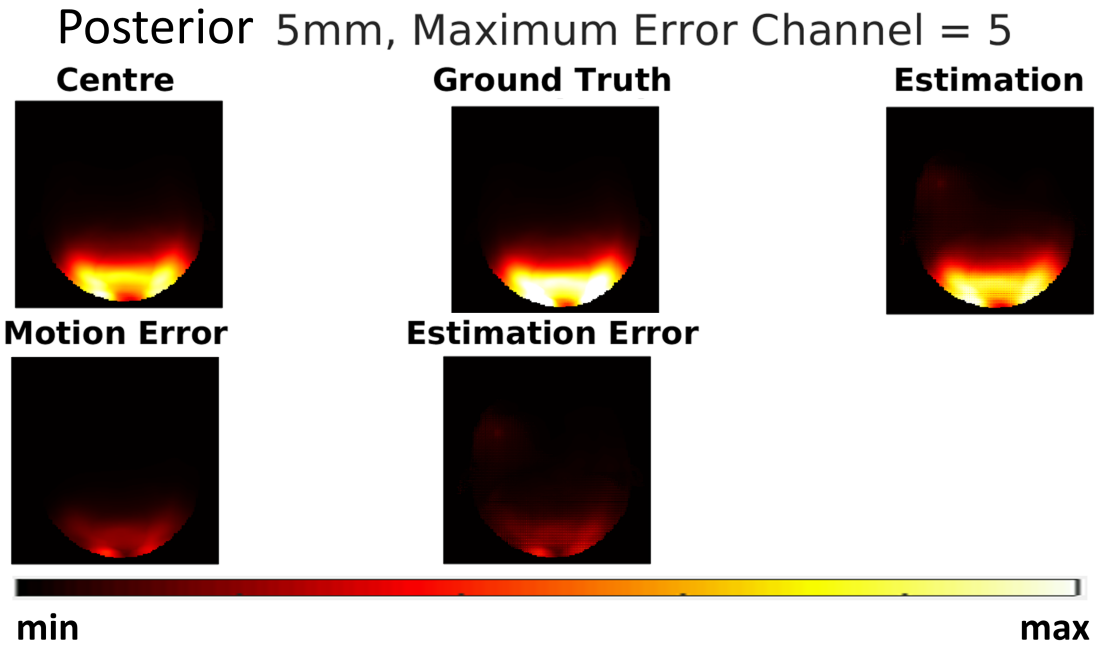


Figure 4.5.2: ilSAR error. This figure shows a central slice for channel 5 ($ilSAR_{5,5}$) which yielded the worst motion error. In the top row, it is evident that the network-predictions more resembled the centered images as opposed to the ground-truth.

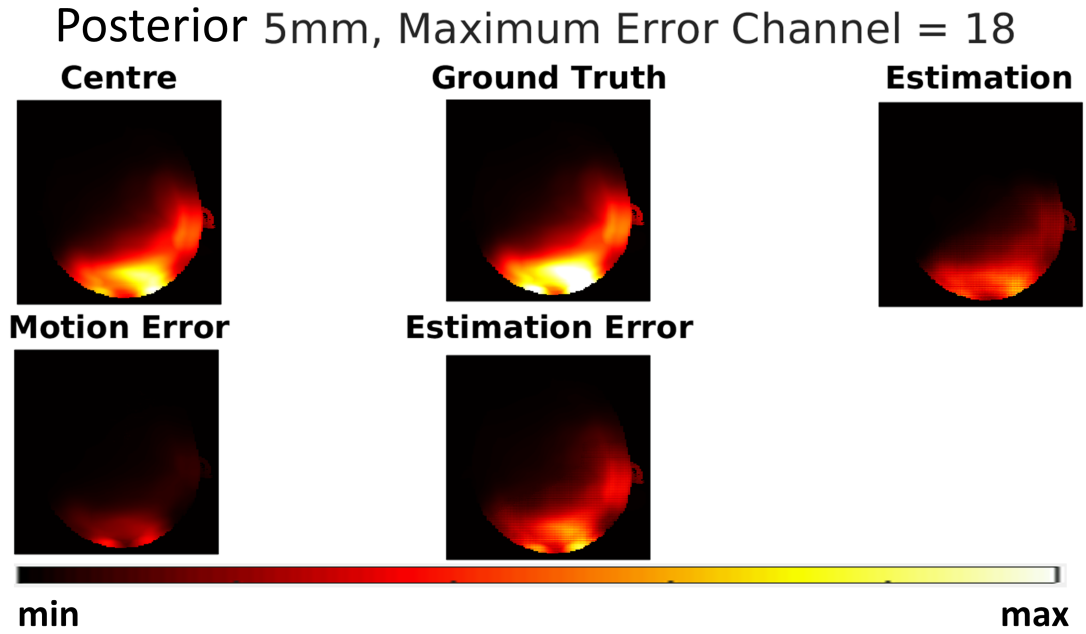


Figure 4.5.3: ilSAR error. This figure shows a central slice for channel 18 ($ilSAR_{6,5}$) which yielded the worst network estimation error. The final image on the top row is a highly inaccurate network estimated image.

Only 6% of the channels (channels 5 ($ilSAR_{5,5}$), 31 ($ilSAR_{5,7}$), 43 ($ilSAR_{5,8}$),

and 54 ($ilSAR_{6,1}$)) experienced lower network estimation nRMSE than default motion nRMSE. Motion nRMSE shows modest variation across the transmit channels and remains mostly below 30%.

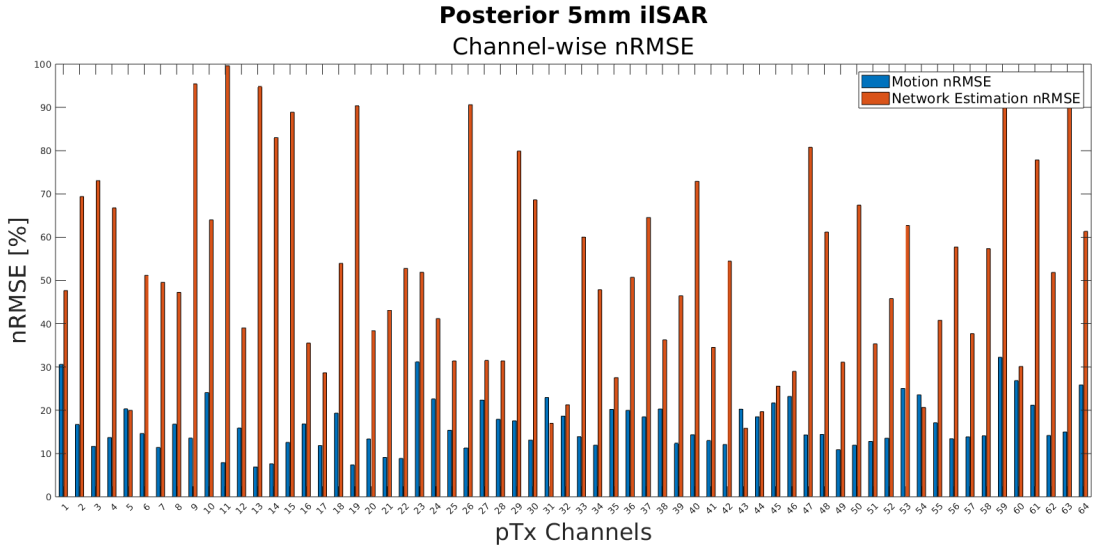


Figure 4.5.4: Network estimation and motion error of ilSAR distributions per pTx channel. The blue bars are default-motion error, while the orange bars are network-estimation error.

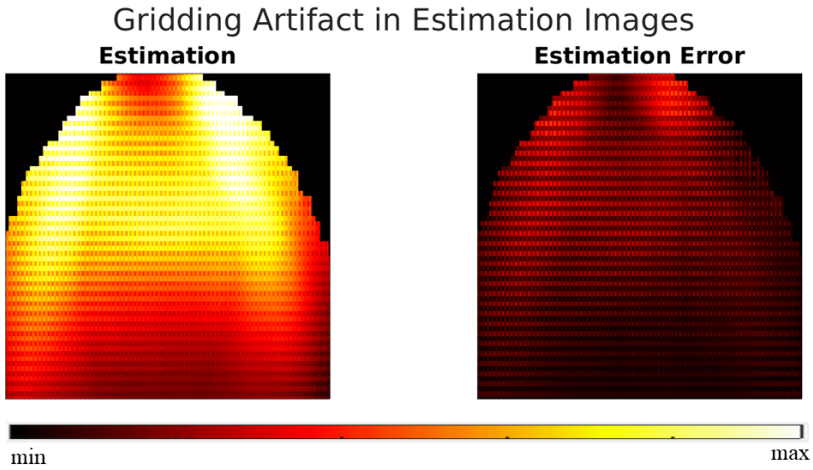


Figure 4.5.5: Network estimated image and associated network estimation error from 4.5.3 magnified to show the gridding artifact.

### 4.5.4 Section Discussion

The use of ilSAR was motivated by the relative success of the Q-matrix magnitude estimations and the hope that a single, real-valued map incorporating power absorption would simplify the training problem. However, it seems to be the case that simply removing complex components does not guarantee improved network performance. The ilSAR predictions performed poorly in most cases, and the appearance of prominent checkerboard artifacts suggested a deeper architectural issue in the network design. The artifact is not only visually evident but potentially contributes significantly to the inflated error values observed throughout the channel maps.

While the network was able to learn some aspects of the ilSAR structure, it ultimately failed to represent the spatial continuity of the distributions, instead defaulting to a patterned output that fit the structure of the upsampling kernel rather than the target data.

The peculiarity of the gridding artifact inspired the following chapter (Chapter 5), since it makes a recurring appearance in computer vision tasks which employ convolutions [145]. The motivation was to learn whether the gridding artifact has an effect on network estimation accuracy, which measures can be taken to eliminate it, and whether eliminating it improves network estimation accuracy.

Channelwise, Figure 4.5.4 suggests that motion-related effects are relatively consistent and do not vary strongly from one channel to another. In contrast, a few channels, for example channel 64 ($Q_{8,1}$), show large differences between the motion and network estimation errors. It is unclear whether this results from the network's tendency towards insufficient generalization or local overfitting. A potential pattern could be discerned from the repeated participation of channel 5, but beyond that there is no immediately obvious trend.

## 4.6 Cumulative Discussion

This chapter describes the process of trialing datatypes intended for a pipeline which uses machine learning to predict motion-induced variations in E-field distributions and related quantities. While initial efforts to predict E-fields were hindered by high estimation errors, the transition to Q-matrix prediction showed promise, markedly for magnitude data. However, phase-related predictions re-

mained problematic, likely due to the singularities and wrap discontinuities inherent to phase representations.

The increased error in $E_x$-field magnitude and phase predictions is a limitation in the chosen data representation. E-field distributions contain high-frequency spatial variations that may not be well captured by conventional convolutional architectures. The need to train separate networks for each field component has the potential to introduce compounding errors. Direct $E_x$-field prediction is likely impractical for real-time MRI safety applications, though very recent studies have been successful in manipulating E-fields to be more suitable as training data for machine learning tasks [146]. Additionally, it is feasible to create a method which combines the individual E-field components ($E_x$, $E_y$, $E_z$) into one data unit in a fourth dimension, yielding a size of 256 x 256 x 8 x 3. This method would prevent the compounding error problem arising from running multiple separate networks for each of the three E-field components. It would also give the network more contextual information about the datatype, thereby improving overall prediction accuracy. This could be an topic for further investigations once the prediction pipeline's results are satisfactory.

The transition to Q-matrix predictions mitigated some of these issues. By encoding RF power deposition in a more compact form, Q-matrices provided a structured representation that was easier for neural networks to learn. The improvement in magnitude predictions suggests that deep learning is well-suited for learning smooth, low-detail distributions. However, the persistent difficulties in phase prediction indicate the need for alternative approaches to manipulating the data to be more palatable for the neural network. Though, for the reasons mentioned previously, the unfavorable phase results are inconclusive. In the future, the effects of larger displacements should be tested.

The real-imaginary approach was explored as an alternative to the phase-magnitude representation due to persistent inaccuracies in phase predictions. This method has been successfully applied in previous deep learning studies for SAR estimation [39], but its implementation here was not ideal. Neither real nor imaginary networks were successful. In the future, it would be worthwhile to test whether the networks might benefit from real and imaginary parts as separate channels within one network rather than treating them independently. In other words, it could be the case that networks fed a 256x256x36x2 input, with the final dimension containing separated real and imaginary parts, would perform better

since they would train on the whole context of the data. In this case, like with the E-fields, later compounding error would be reduced since ultimately the real and imaginary parts would have to be recombined. Providing the network with more context and layering datatypes into a separate dimension was successfully implemented in the literature [39] [85].

Beyond network estimation accuracy concerns, the implementation of real-imaginary networks was further hindered by persistent TFRecord file corruption issues. Multiple attempts to rewrite the preprocessed training pairs into a valid TFRecord format were unsuccessful, requiring a complete reprocessing of the dataset on three separate occasions. Surprisingly, after a month-long pause in network implementation with no modifications to the preprocessing pipeline, the previously failing network code executed successfully. This unexpected resolution suggests that the issue may have been linked to underlying hardware memory constraints, software inadequacies, or filesystem inconsistencies rather than errors in dataset generation itself. The instability of TFRecord-based input pipelines once again calls into question the reproducibility and reliability of open-source, deep learning workflows.

The ilSAR investigation, although unsuccessful in this implementation, still has utility. Its promise lies in consolidating the pipeline into a single network and bypassing the challenges of phase modeling. The high estimation error may stem from architectural or data representation issues that can be optimized in future iterations. For this reason, its continued exploration is justified.

A notable limitation of the approach described in this chapter is its reliance on insufficiently anatomically varied training datasets constrained by the number of available body models. The anatomical variability across models may not fully reflect the range seen in clinical applications, potentially reducing generalizability. Previous work has emphasized the importance of subject-specific modeling for accurate SAR estimation [86]. Future work should consider expanding the dataset to include a larger and more diverse set of anatomical models, which could improve network generalization and robustness. This expansion is unlikely without significant funding, considering the high cost of model acquisition.

Another limitation is that the network-generated results in this chapter were not subsequently applied to the more definitive pTx evaluation method described in Chapter 6. It could be the case that even though the visual error and empirical calculations reported in this chapter are unfavorable and suggest poor network

estimation performance, the combined halves of the complex data types could nonetheless perform well relative to initial position and initial position when used to calculate predicted local SAR distributions for realistic pTx pulses.

Given the cGAN's successful estimation of the Q-matrix magnitudes, it was surprising that the ilSAR distributions failed. Moreover, the observed gridding artifacts in network-predictions cast doubt on the suitability of cGANs for the given task. The recurrence of such artifacts in computer vision applications indicates that improvements in convolutional architecture may enhance predictive accuracy [145]. Addressing these architectural limitations could improve network performance and enable more accurate field estimations. This final point is further explored in the next chapter.

## 4.7 Conclusion

This chapter explored the use of GANs for predicting motion-induced variations in datatypes relevant to MRI safety. While the initial approach using $E_x$-fields proved infeasible due to high error rates and computational inefficiencies, transitioning to Q-matrix predictions offered a more structured and tractable solution. The success of magnitude predictions suggests that neural networks can effectively learn smooth RF deposition patterns, but prediction failures persist in phase estimation due to singularities and wrap discontinuities.

The findings support the importance of selecting appropriate data representations for deep learning applications in MRI safety investigations. Additionally, the work illustrates the need for larger, more diverse training datasets and more suitable network architectures.

Despite its poor initial performance, ilSAR remained a promising direction, offering a simplified, more elegant, single-network approach that bypasses phase modeling. Future chapters show how its refinement significantly improved the SAR estimation pipeline.

# Chapter 5

# Grad Student Descent: The Effects of Simulated SAR Data Processing Methods and Network Parameter Tuning on Gridding Artifacts and Network Estimation Accuracy

## 5.1 Foreword

Alix Plumley wrote the original pipeline that was adapted for the following investigations. Plumley's pipeline included the preprocessing, training, evaluation, and postprocessing procedures. The scripts were altered as deemed appropriate by the author. Error metrics were altered.

The data used in this chapter was simulated and the codes written to process the simulation data were written by Emre Kopanoglu. The scripts were used in their original, unaltered form.

The suggestion to forward map Q-matrices to local SAR matrices was put forward by Shaihan Malik. The algorithm used to do so was published in Ref. [132].

## 5.2 Introduction: Gridding Artifacts Overview

During the development of the ilSAR prediction pipeline, a gridding artifact was discovered in the network-predicted images (Chapter 4). This artifact persisted through post-processing. An example of the artifact is presented in Figure 5.2.1.

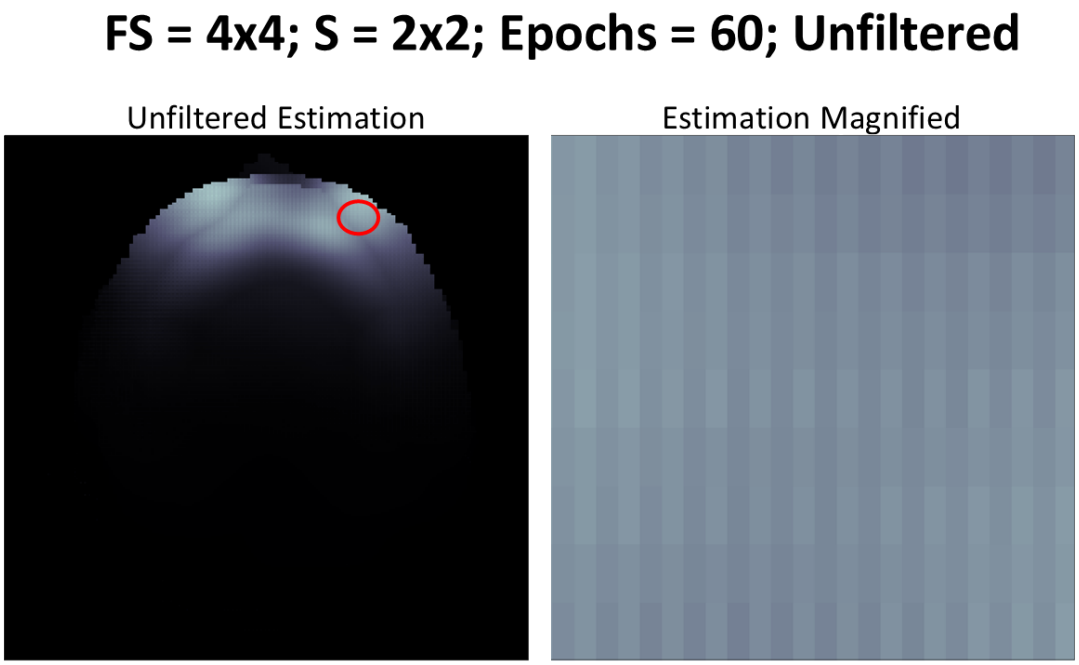


Figure 5.2.1: Network-estimated ilSAR distribution from a centred to P5 mm translation containing a gridding artifact after postprocessing. **FS** denotes filter size, while **S** denotes stride. The number of epochs are listed. The image is unfiltered (with neither Gaussian nor Hanning filters). This is a slice taken from the center of the head in the axial plane with a SAR distribution arising when only the first channel is on.

According to Ref. [145], the gridding or "checkerboard" artifact could occur during the encoding portion of the U-Net which uses transposed convolution, or deconvolution. As described in Chapter 2, convolution occurs in the first portion of the U-Net (decoding) and is the process of using a sliding window of a predefined size with weighted kernels over an image to detect features. This process reduces the spatial dimensions of the input volume. The second part of the U-Net uses deconvolution which uses a transposed convolutional layer to increase the spatial resolution of the feature maps. It is the opposite of a regular convolution operation. This convolutional layer has learnable parameters that allow it to learn how to upsample the feature maps effectively.

To investigate the effect of an emerging gridding artifact on network prediction accuracy, a variety of scenarios are tested. These range from dataset manipulation (altering which body models are used for network training, validation, and testing), to network parameter and hyperparameter tuning, to applying filters in postprocessing.

For computational speed, the results reported in this section arise from only seven mid-head slices from the crown to the cerebellum. The chosen area contains a large amount of tissue data. Error between the ground-truth and network-estimation images is reported in terms of L1 error. This is because gridding artifacts typically appear as structured, high-frequency errors that can be spatially localized but have high-intensity deviations. L1 error (mean absolute error) treats all errors equally and is less influenced by large outliers, making it a robust metric for general error magnitude. In contrast, metrics like nRMSE emphasize large, isolated errors because they square the differences. As a result, nRMSE can sometimes underrepresent consistent and structured error patterns, such as those caused by gridding, if those errors are relatively small in magnitude.

## 5.3 Methodological Overview

Identical to previous chapters, the cGANs tested in this study are variations of the one used in [8], which is a marginally altered version of pix2pix [83]. Ref. [8] decreased the number of epochs from 200 to 60 and removed batch normalization in the generator network during downsampling (convolution). Like in Chapter 4, the networks were trained on a posterior 5 mm (P5 mm) displacement.

Briefly, the following methods to test the prevalence of a gridding artifact and its effect on network estimation accuracy were implemented:

1. Implemented Hanning filter (HF) in postprocessing to mitigate the gridding artifact (configuration FBD-G-E, see bullet point 2), FSs: 5x5, 4x4, 3x3, 2x2 and 1x1.

2. Body model configurations investigated using the cGAN parameters from [8] with a leaky rectified linear unit (LReLu) instead of a ReLu during convolution in the generator:

   (a) Train: Fats, Ella, Glenn; validate: Billie; test: Duke

(b) Train: Fats, Billie, Duke; validate: Glenn; test: Ella (configuration FBD-G-E)

(c) Train: Fats, Ella, Glenn; validate: Duke; test: Billie

3. Neural network (NN) parameters evaluated on FBD-G-E:

(a) Changing the leaky rectified linear unit (LReLu) to a ReLu

(b) FS=2

(c) FS=3

(d) FS=3; stride=1

(e) FS=1; stride=1

(f) FS=1; strides=1; change LReLu to a ReLu during convolution in generator

(g) Epochs=160

The detailed justification, methodology and associated observations are presented in depth in the next sections.

## 5.4 Methods

### 5.4.1 Error Metrics

Error metrics in this chapter are given in terms of L1 error (%) and percent L1 error reduction. This L1 error reduction is calculated by

$$\Delta_e = \frac{x - \hat{y}}{x} \times 100$$

where $x$ denotes the motion-induced error and $\hat{y}$ denotes the network-estimated error (NNE).

### 5.4.2 Hanning Filter

The first tested mitigative method was implemented by means of filtering the network-estimated ilSAR distribution.

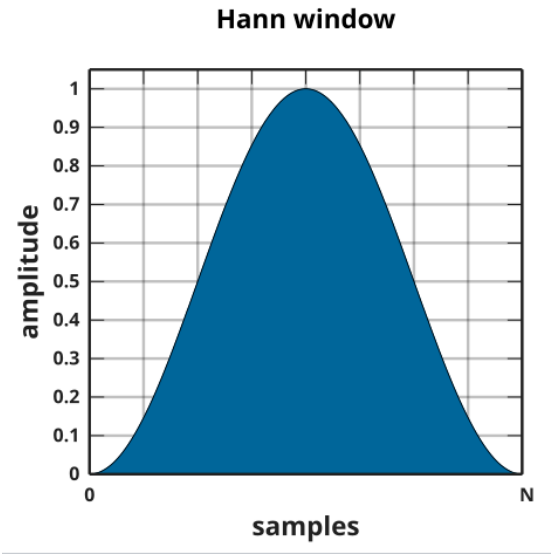


Figure 5.4.1: Hann function. Image retrieved and adapted from Ref. [10].

The original postprocessing pipeline from Ref. [8] applied a Gaussian filter to the images. This did not fully eliminate of the gridding artifact, so an additional Hanning filter was applied (filter size 5x5 pixel coverage) in MATLAB.

Figure 5.4.1 shows a Hanning filter, also known as a "Hann window" or "cosine window". It is a type of windowing function used in signal processing and image processing.

It is defined mathematically as

$$w(n) = 0.5\left(1 - \cos\left(2\pi\frac{n}{N}\right)\right), \quad 0 \leq n \leq N.$$

where $w[n]$ is the window function at sample $n$, $N$ is the total number of samples (window length), $n$ is the sample index, ranging from 0 to $N$, inclusive. When applied to an image, a Hanning filter can smooth and enhance certain characteristics. It acts as a low-pass filter, attenuating high-frequency noise and sharp transitions. Hanning filters can be useful for reducing artifacts in images in the presence of aliasing or other high-frequency noise. This type of filter has a tapering effect, which means that it gradually reduces the amplitude of pixel values with increasing distance from the center of the filter. This can reduce ringing artifacts that may occur when sharp edges are subjected to traditional filtering [147–149].

To eliminate the high intensity pixels in the network estimation error image, a

range of Hanning filter sizes were tested, from 5x5 down to 2x2 (1x1 is excluded because it is effectively not a filter). The Hanning filters of varying sizes were applied to all of the slices and channels of P5 mm network-estimated ilSAR distributions.

### 5.4.3 Sim4Life body model configurations

A total of five body models from Sim4Life were available for the whole data set, which was split model-wise into training, validation, and testing. The training data set consisted of three models, and validation and testing each consisted of one model. This setup reflects a common practice where a larger proportion of the data is used for training to allow the model to learn a diverse set of features, while smaller validation and testing sets help evaluate the model's generalization capacity [150].

The first body model configuration used Ella, Fats, and Glenn for training, Billie for validation, and Duke for testing. Previous experience showed that this configuration caused errors theoretically resulting from Duke's large nose being outside of the anatomical bounds of the rest of the body models. Therefore, additional configurations were tested.

Next, Duke, Fats, and Billie were used for training, Glenn for validation, and Ella for testing. This configuration was designed to ensure a more balanced representation of the models in the training set, which may have allowed the network to generalize better to a new, unseen, and more anatomically-typical body model during testing.

Finally, a body model configuration using Ella, Fats, and Glenn for training, Duke for validation, and Billie for testing was trialed using the original network and postprocessing pipeline parameters. This configuration was more in line with the one used for [8], where Duke was used for training, Dizzy (no longer available) and alternating slices of Billie for validation, and the remaining Billie slices for testing.

### 5.4.4 Hyperparameter Tuning

The gridding artifact could relate to the stride and the filter size used during encoding. The filter size is the number of pixels covered by the sliding window, and the stride is the number of pixels the window moves by. The combination of

these parameters could cause a gridding artifact when the filter size is not divisible by the stride, leading to uneven overlap. This can result in noticeable grid-like patterns in the output, which can detract from the natural appearance of the generated images [145]. On the other hand, perfectly even overlap causes artifacts as well. In such cases, the regularity of the overlap can lead to repetitive patterns, which may be perceived as artificial or unnatural in the generated image. Having no overlap at all (filter size = stride), otherwise known as sub-pixel resolution [151], has been proposed as a potential solution, though it still has the capacity to create checkered artifacts. This approach can work well in certain contexts where high-resolution detail is prioritized, but it may struggle to maintain smooth transitions between pixels, leading to visible discontinuities or jagged edges.

Another potential solution is to increase the training by the number of epochs in order to yield more effective generator weights for testing. Extending the training process might allow the model to better generalize to unseen data and produce more accurate results. Though the present network implementation does not use the generator weights from the final epoch but rather from the epoch yielding the lowest loss, more training epochs may give the network a higher chance of securing a generator weight with lower loss. However, this is not guaranteed, as diminishing returns may be encountered after a certain point, and overfitting could become a concern.

**Halving the filter size from 4 to 2 [1]**

This method is equivalent to a sub-pixel resolution implementation, where the filter size is equally divisible by the stride [151]. This method provides a high level of detail while maintaining low computational cost. Smaller filter sizes allow the network to map finer details in the generated images. They have a smaller receptive field, meaning they extract less context from the input image [152]. Halving the filter size reduces the number of parameters in the model, which can lead to faster training and inference times and lower memory requirements [153]. As a result, smaller filter sizes are computationally more efficient. They can increase the capacity of the network's ability to learn more complex patterns and representations, though they can make the model more prone to overfitting, especially when the dataset is small or noisy [133]. In image generation tasks, smaller filter sizes could help generate more detailed and intricate images, but they may also lead to over-emphasis on fine details at the expense of global structure [55]. Moreover, the limited context captured by smaller filters could

hinder the model's ability to learn larger, more coherent patterns in the image.

**Changing the filter size from 4 to 3 [2]**

Reducing the filter size from 4×4 to 3×3 while maintaining a stride of 2 is a commonly used configuration in convolutional neural networks [55]. This configuration aims to preserve details at an optimal computational cost. Smaller filters, such as 3×3, have a reduced receptive field compared to 4×4 filters, which can be beneficial for detecting granular spatial features. The downside is that this setup may limit the model's ability to learn broader contextual information from the input data, which may be essential for maintaining structural coherence.

The use of a stride that is not proportionally smaller than the filter size can cause uneven overlap resulting in misalignment that can manifest in the gridding artifact. This filter size and stride combination is not likely to eliminate the gridding artifact, but the way the artifact is altered and its effect on empirically assessed network estimation accuracy is informative.

**Changing the filter size from 4 to 3 and the stride from 2 to 1 [3]**

This combination is also commonly used for standard convolution operations because it preserves spatial resolution and is also effective for detecting fine details. A smaller filter size, such as 3, reduces the receptive field, allowing the network to focus on more localized patterns, which is advantageous in applications requiring intricate details, like high-resolution image synthesis [153]. It is therefore more suitable for tasks where detailed information is important. A stride of 1x1 means that the filter moves one step at a time along both the horizontal and vertical dimensions of the input data. This smaller stride results in denser sampling of the input space, which is useful for resolving small-scale variations in images [152]. This method can be used to preserve the spatial dimensions of the input as much as possible. The choice of stride in convolutional layers affects the output size of the layer. With a 1x1 stride, the output size may be the same or very close to the input size, depending on the size of the convolutional filter and any padding used. For example, with zero padding, the spatial dimensions of the output remain identical to the input, whereas padding reduces the output size based on the filter dimensions [55]. This can be useful for maintaining subtle spatial information in the feature maps. Convolutional layers with a 1x1 stride can be used for feature extraction or generating high-resolution images while preserving spatial

details. The 1x1 stride also allows for deeper networks with fewer parameters, which helps improve computational efficiency [153].

**Changing the filter size from 4 to 1 and the stride from 2 to 1 [4]**

This method also reduces computational cost by applying pointwise convolution. Pointwise convolution (1x1 convolution) reduces the depth of feature maps while preserving detailed information [153]. Pointwise convolution in a cGAN can enhance the model's expressive power, control the model's capacity, enable channel-wise interactions, and improve computational efficiency.

In the context of neural networks (as opposed to the pTx context), "channels" refers to the different information-carrying components of an image or feature map. Each channel represents a particular aspect or feature of the data. For example, in color images, each channel might correspond to a color (red, green, blue), and in feature maps, each channel might represent different learned features, such as edges, textures, or high-level semantic information. Point-wise convolutions operate on each element (pixel) independently in each channel of the input tensor. This means that they can extract relationships and dependencies between different channels effectively.

**Changing the filter size from 4 to 1, the stride from 2 to 1, and replacing the LReLu with a ReLu [5]**

ReLU (Rectified Linear Unit) and Leaky ReLU are both types of activation functions commonly used in deep learning models. As explained in Chapter, 2, an activation function in deep learning is a mathematical function applied to the output of a neural network layer. It introduces non-linearity into the network, allowing the model to learn patterns in the data. Without it, a neural network would be akin to a linear model, regardless of how many layers it has, limiting its intended ability to learn patterns in the data.

A ReLU transforms its input by maintaining positive values while nullifying negative values. It is defined as:

$$f(x) = \max(0, x)$$

where $x$ is the input to the activation function. If $x \geq 0$, then $f(x) = x$. If $x < 0$, then $f(x) = 0$. A ReLu is computationally efficient and accelerates the training

of deep networks. One major drawback is that since it nullifies negative values, it has the potential to kill neurons and disable elements of the network architecture (dubbed the "Dying ReLu" problem). For this reason, Ref. [83] used a Leaky ReLu which avoids the dying ReLu problem by incorporating a slight negative slope for negative values, instead of nullifying them. It is defined as:

$$f(x) = \begin{cases} x, & x \geq 0 \\ \alpha x, & x < 0 \end{cases}$$

where $\alpha$ is a constant smaller than 1 [134, 154].

The pointwise convolution cGAN was tested with a ReLu instead of a LReLu in the generator. The LReLu in the original network was also replaced by a plain ReLu.

**Training across 160 epochs [6]**

One of the primary benefits of increasing the number of epochs is that the model has more opportunities to learn and refine its representations. More epochs allow the model to iterate over the dataset more times, progressively refining its ability to learn patterns [155]. In this instance, the number of epochs was increased by 100, from 60 (as initially implemented in [8]) to 160. Increasing the number of epochs may enable the cGAN to generate more diverse samples because it has more time to explore different attributes in the data distribution [55]. With more training time, the generated images may become more accurate. As the model sees more data and refines its weights, the generator becomes better at producing realistic outputs, and the discriminator improves at distinguishing between real and generated data [156]. Training for longer allows the generator and discriminator networks to refine their parameters, reducing the adversarial gap and making it more difficult for the discriminator to distinguish between real and generated images [55].

There is a risk of overfitting if the cGAN is trained for too many epochs, especially if the dataset is small or unvaried. Overfitting occurs when the model starts to memorize the training data rather than generalizing from it, leading to poor performance on unseen data [157]. Overfitting can lead to the generator producing images that closely match the training data but lack diversity or generalization to new, unseen data. This is why early stopping is employed to prevent the model

from training beyond the point where it starts to overfit [158]. Accordingly, the current cGAN implementation mitigates overfitting by means of early stopping, where the weight with the best L1 loss is chosen for evaluation, as opposed to the weight arising from the final epoch. Generally, early stopping works by monitoring the performance of the model on a validation set, and halting training once the model's performance starts to degrade [159].

## 5.5 Observations

### 5.5.1 Hanning Filter

Figure 5.5.1 shows the effects of increasing Hanning filter sizes on the gridding artifact. After filtering the network estimation with a Hanning filter, the gridding artifact disappeared, but the estimation error (the difference between the simulated ground-truth image and the network-estimated image) yielded regions with high intensity pixels, shown in Figure 5.5.1. Hanning filter size is proportional to the emergence of the high intensity pixels observed in the error images. For this reason, the Hanning filter is a poor solution to eliminating the gridding artifact, since high intensity pixels, even at the highest error reduction, are undesirable. Figure 5.5.6 shows that the best performing error reduction with the application of the Hanning filter is mean 45.32%, maximum 69.46% (3x3 pixels).

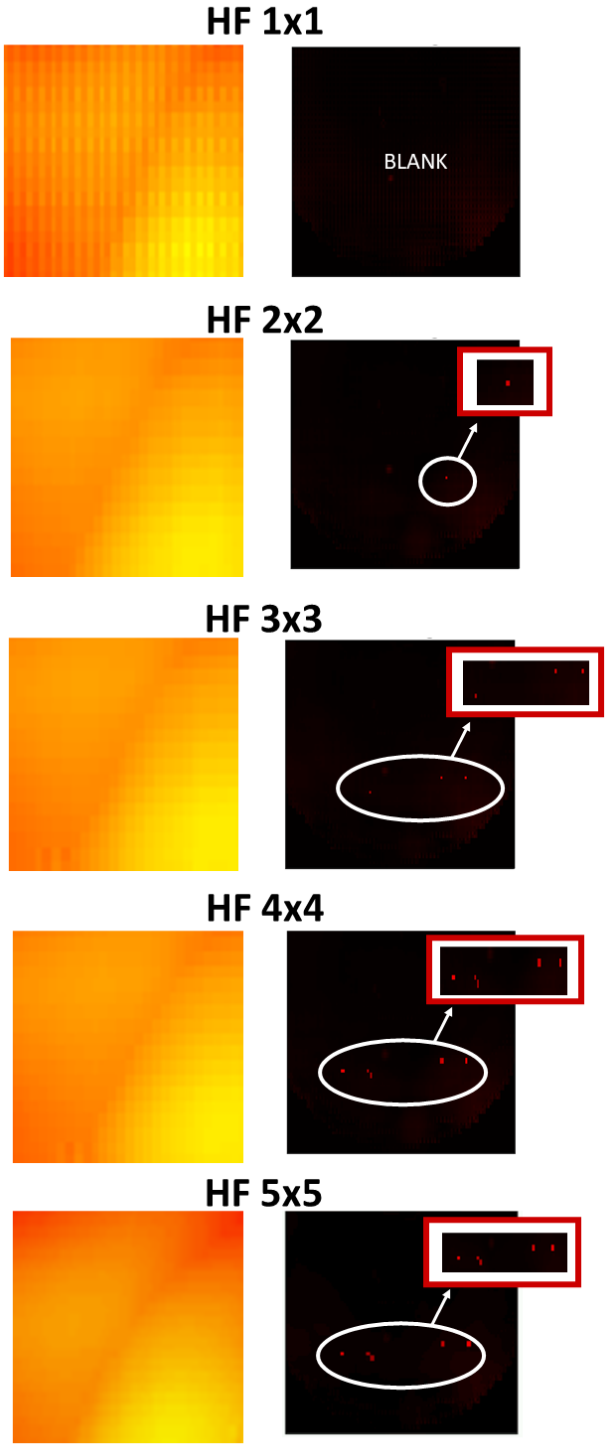


Figure 5.5.1: Effects of applying Hanning filters (**HF**) of varying sizes on P5 mm ilSAR estimations. From top to bottom: the rows increase in size of applied Hanning filter from 1x1 pixels to 5x5 pixels. The ilSAR distributions are maximum intensity projections on the z-axis. The left column is a magnification of the network-estimated image. The right column shows the high intensity pixels which are visible when visualizing the difference between the network-estimated and simulated ground-truth ilSAR distribution. The high intensity pixels are magnified in the top right corner for improved visibility.

### 5.5.2 Sim4Life body model configurations

The results from the configuration where Ella, Fats, and Glenn were used for training, Billie for validation, and Duke for testing were poor (L1 error reduction: mean 11.80%, maximum 49.61%) (Figures 5.5.2 and 5.5.6), presumably because Duke's nose extended past the spatial confounds occupied by the models used for training. This misalignment could have led to a mismatch in the spatial features learned by the network, affecting its performance on unseen data with anatomical differences. Not only was the gridding artifact present, but after passing the estimations through the Hanning filter, the high error values were concentrated in the nasal area of the predicted images (Figure 5.5.2). The presence of high error values in specific regions suggests that the model might not have been able to handle the anatomical variances in the data, possibly due to insufficient training on similar spatial elements.

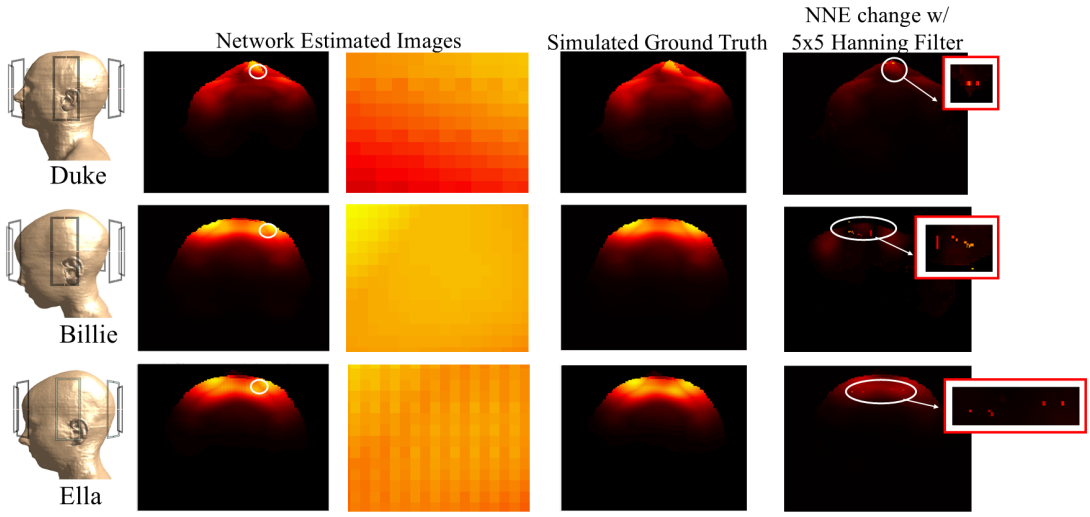


Figure 5.5.2: Gridding artifact expression arising from a P5 mm displacement across three body models using network parameters with filter size $4 \times 4$ and stride $2 \times 2$. Each row corresponds to a different model. The ilSAR distributions are maximum intensity projections along the z-axis. Column 1 shows the body model positioned within the pTx coil as simulated in Sim4Life. Column 2 presents the ilSAR distribution predicted by the network. Column 3 highlights a magnified region (indicated by a white ellipse) where artifacts are most visible. Column 4 displays the ground-truth ilSAR distribution. Column 5 visualizes the Hanning-filtered difference between the neural network-estimated and initial position images, with high-intensity regions further magnified in the red inset. The NNE (Neural Network Estimated) change (i.e., the difference between the NNE image and the initial position image) is shown so that error images contain anatomical features and better illustrate the location of high-intensity error pixels. These pixels are equally apparent in the NNE error images (ground-truth - NNE). This illustrative method is repeated in Figures 5.5.5 and 5.5.4.

The configuration where Duke, Fats, and Billie were used for training, Glenn for validation, and Ella for testing yielded more favorable results. The overall error for this configuration is significantly lower (L1 error reduction: mean 38.38%, maximum 68.56%) than when using Duke for testing, even given the original network and postprocessing pipeline parameters (Figure 5.5.6). This improvement in performance could be attributed to the better alignment of body models during training, leading to more resilient network learning. It could also be attributed to Ella's anatomy lacking the more extreme peculiarities seen in Duke. Meanwhile, applying the Hanning filter still caused high intensity pixels to appear (Figure 5.5.2.)

The body model configuration using Ella, Fats, and Glenn for training, Duke for validation, and Billie for testing yielded mean and maximum L1 error reduction values lower than when testing on Ella (mean 33.25%, maximum 67.94%; Figure 5.5.6). In other words, the network-estimated ilSAR distributions are less effective in the former than in the latter case. Notably, the gridding artifact was also not present in Figure5.5.2, second row. This finding supports the notion that the specific body model configuration can influence the presence of artifacts, and choosing an appropriate model for training and testing can help mitigate it presence. It also indicates that decreasing the emergence of a gridding artifact does not necessarily yield the relative network-caused L1 error improvement.

Figure 5.5.2 also shows that applying a 5x5 Hanning filter caused high intensity pixels to appear in the Billie model, even though the gridding artifact disappeared. This indicates that the gridding artifact and the high intensity pixels arising from applying a Hanning filter may not be connected.

### 5.5.3 Hyperparameter Tuning

#### Halving the filter size from 4 to 2 [1]

Halving the filter size but maintaining the same stride reduced the network estimation error (mean 34.84%, maximum 69.95%; Figure 5.5.6) and altered the gridding artifact without eliminating it (Figure 5.5.3). Therefore, further tuning of filter sizes and strides was necessary to fully mitigate the artifacts and reduce network estimation error. The maximum error occurred during the ilSAR[1,1] channel interaction.

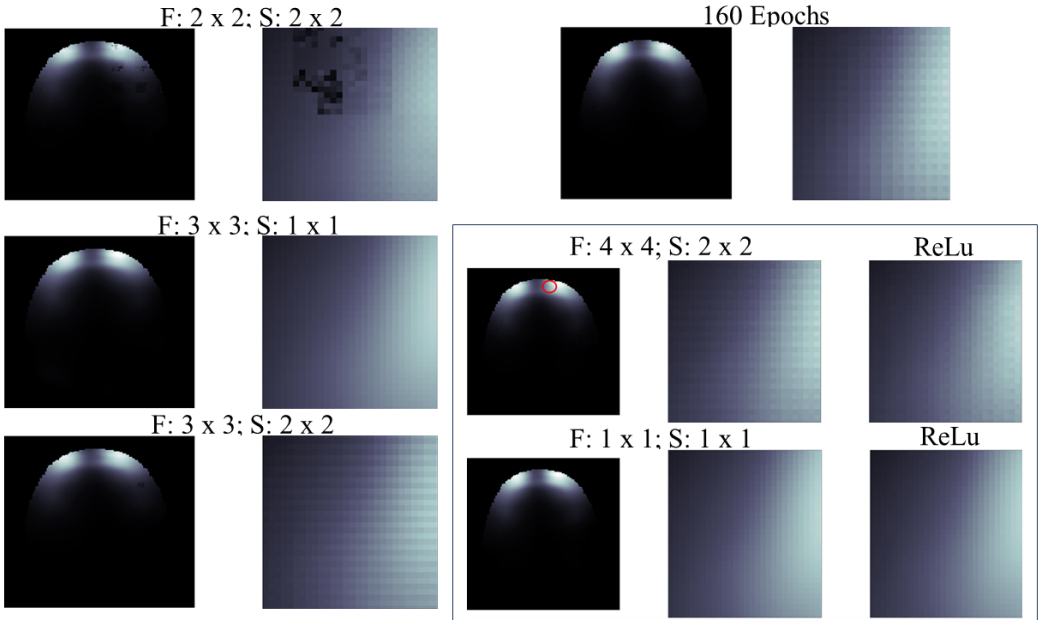


Figure 5.5.3: ilSAR distributions and corresponding magnified gridding artifacts resulting from different network parameter configurations. All ilSAR distributions are displayed as maximum intensity projections along the z-axis. Here, **F** denotes filter size and **S** denotes stride. The label **ReLU** refers to the network variant where leaky ReLU activations were replaced with standard ReLU. All magnified insets correspond to the same anatomical region, marked by a red ellipse in the baseline configuration (**F = 4×4, S = 2×2**).

### Changing the filter size from 4 to 3 [2]

Decreasing the filter size but maintaining the same stride altered the gridding artifact by increasing the artifact's contrast (Figure 5.5.3). Reducing the filter size from 4 to 3 decreased the receptive field of the convolutional kernel, potentially leading to less context being encoded from the input image. This may have exacerbated the artifact, as the smaller filter created more pronounced boundary effects between regions. The stride of 2, while reducing the computational cost, contributed to the spatial resolution of the generated image and led to the misalignment of pixel activations during the forward pass [160]. As a result, when the filter size is decreased from 4 to 3, the reduced overlap coupled with the fixed stride made the checkerboard artifact more noticeable. The mean and maximum L1 error reduction was similar at 35.90% and 70.11%, respectively (Figure 5.5.6). This mean error reduction is worse than the original model from [8] when testing on Ella (mean 38.38%, maximum 68.56%). Again, the maximum error occurred during the $ilSAR_{1,1}$ channel interaction.

**Changing the filter size from 4 to 3 and the stride from 2 to 1 [3]**

The gridding artifact was not present with the application of a 3x3 filter size and 1x1 stride (Figure 5.5.3), but the L1 error reduction between the network-estimated and ground-truth image was not improved (mean 38.85%, maximum 53.46%; Figure 5.5.6). This lack of improvement in the L1 error might indicate that the filter size and stride combination, while preserving spatial resolution, may not have mapped features adequately, limiting the model's performance in the current setup [55]. It is possible that the combination of a small filter size and a 1x1 stride was too constrained for detecting larger-scale patterns or features within the dataset, suggesting a trade-off between preserving spatial resolution and capturing more global features.

On the other hand, it's possible that the combination of minimal stride and large filter size produced such extensive filter overlap that the artifact was effectively suppressed, but at the cost of overrepresenting contextual information, which ultimately hindered model learning and degraded performance.

Interestingly, with these parameters, the channel interaction producing the maximum amount of error changed from $ilSAR_{1,1}$ to $ilSAR_{5,5}$. This shift could suggest that the network, due to the change in filter size and stride, began focusing on different aspects of the image, such as other structural features or patterns, leading to an altered error distribution.

Notably, the Hanning filter caused high intensity pixels in the error images while the gridding artifact was not present (Figure 5.5.4). In Figure 5.5.4, the high intensity pixels are not visible in the NNE Hanning filtered image in the left panel because they are likely camouflaged by surrounding pixels. The high intensity pixels are only visible in the NNE change image (righthand panel: NNE - Initial Position) and the NNE error image (not shown; NNE - Ground Truth). The NNE change image is displayed here because it shows more spatial context than the NNE error image. The appearance of the high intensity pixels despite the gridding artifact not being present further solidifies the prospect that the manifestation of extreme, localized error pixels are unconnected to the gridding artifact. The Hanning filter may have introduced smoothing effects in the error computation that inadvertently emphasized the high-intensity error areas.

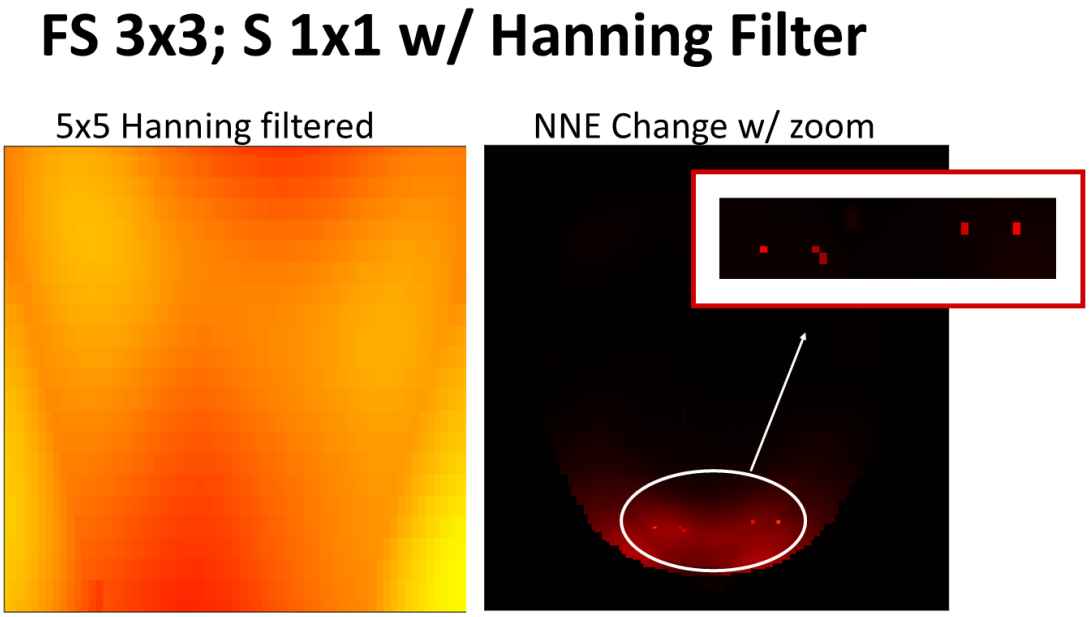


Figure 5.5.4: The ilSAR distributions are maximum intensity projections on the z-axis. **FS** is filter size, **S** is stride, and **NNE** is neural network-estimated. The panel on the left is the magnified network-estimated image. The area of magnification is from the same region shown in the right panel. The right panel shows the difference between the network-estimated image and the initial position image.

### Changing the filter size from 4 to 1 and the stride from 2 to 1 [4]

Changing the filter size from 4 to 1 and the stride from 2 to 1, like in [3], eliminated the gridding artifact (Figure 5.5.3), but the Hanning filter reintroduced the high-intensity pixels in the network estimation error.

Without the Hanning filter, the error reduction improved (mean 49.26%, maximum 73.73%). The reduction in L1 error with the 1x1 filter and stride of 1 implies that the finer-grained processing enabled by the pointwise convolution improved the overall accuracy of the generated images.

As in the previous subsection, the Hanning filter caused identical high intensity pixels in the same location as when a 3x3 filter and 1x1 stride were selected for the neural network architecture (Figure 5.5.5). This allows speculation that the high intensity pixels could be related to the body models rather than the gridding artifact.

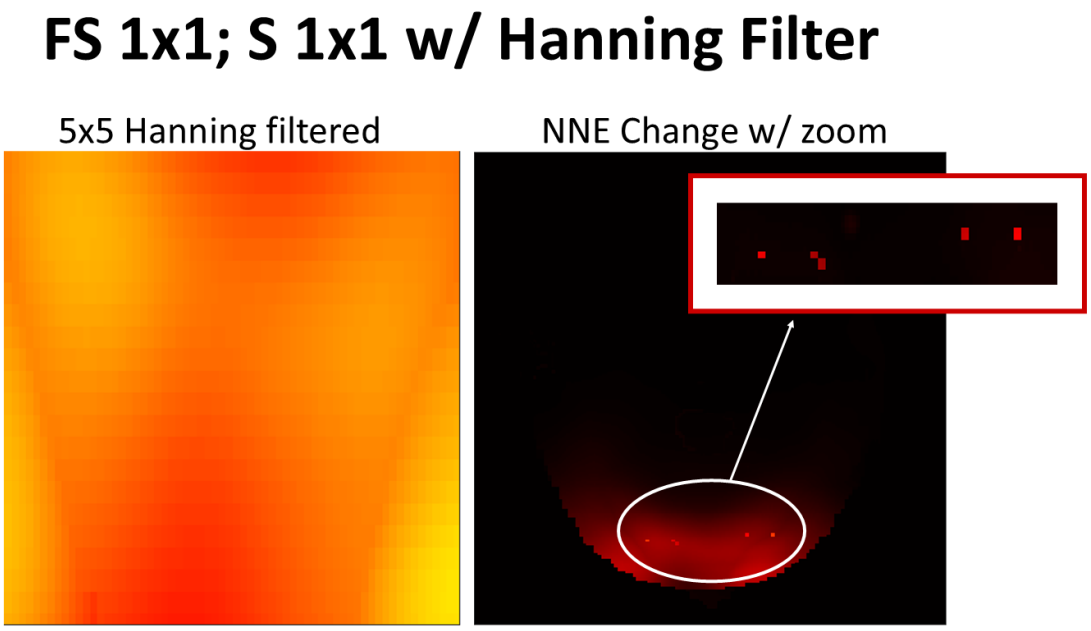


Figure 5.5.5: The ilSAR distributions are maximum intensity projections on the z-axis. **FS** is filter size, **S** is stride, and **NNE** is neural network-estimated. The panel on the left is the magnified network-estimated image. The right panel shows the difference between the network-estimated image and the initial position image.

**Changing the filter size from 4 to 1, the stride from 2 to 1, and replacing the LReLu with a ReLu [5]**

The pointwise convolution cGAN with a ReLu instead of a LReLu in the generator yielded a mean and maximum L1 error reduction of just 22.95% and 34.78%, respectively (Figure 5.5.6). While the gridding artifact remained eliminated (Figure 5.5.3), the decreased mean error reduction indicates that a LReLu is a more effective activation function for this network architecture and estimation task.

When the LReLu in the original network (filter size = 4x4, stride = 2x2) was replaced by a plain ReLu, the mean L1 error reduction was also 22.72%, but the maximum error reduction rose to 58.46% (Figure 5.5.6), and the gridding artifact persisted (Figure 5.5.3), but its presentation was altered. These results suggest that 1) the activation function does not affect the emergence of a gridding artifact, but has the potential to change the network estimation accuracy, 2) while not eliminating the emergence of a gridding artifact, a change in activation function can alter how it manifests, and 3) a change in the L1 error may not be related to the emergence of the gridding artifact.

### Training across 160 epochs [6]

Increasing the number of epochs by 100 did not eliminate the gridding artifact and maintained the mean L1 error reduction at 42.44% (maximum 64.90%), compared to the original network implementation with 60 epochs (Figures 5.5.6 and 5.5.3). While all of the above networks were run using 1 GPU, the computational load of 160 epochs required the use of 3 GPUs. Therefore, it can be concluded that increasing the number of epochs is not effective at eliminating the gridding artifact or improving the error reduction, and is severely computationally inefficient.

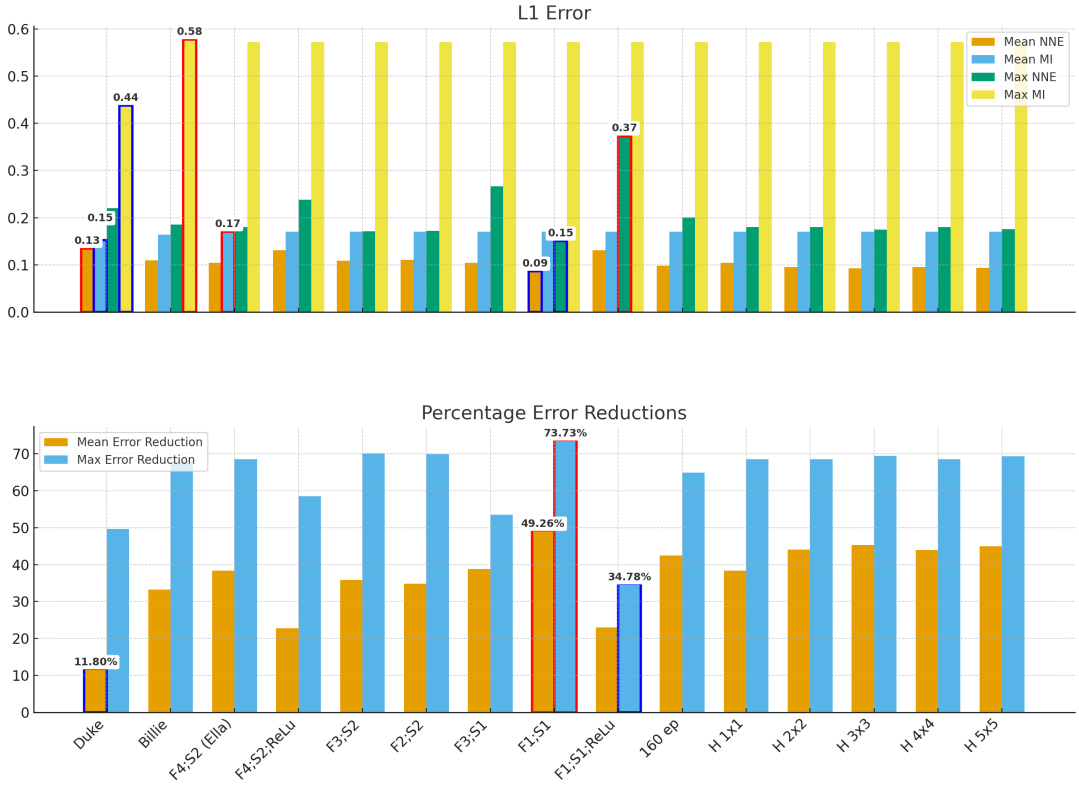


Figure 5.5.6: Graphs summarizing the errors for each investigation. Top graph: overview of motion induced (**MI**) and network estimation error (**NNE**). The units for L1 error is %. x-axis labels on bottom graph correspond to top graph. Bottom graph: A visualization of the error reductions for the tested parameters. L1 error values from networks run with various parameter changes, where Billie and Duke are the alternative BM configurations, **F** is filter size, **S** is strides, **ep** is epochs, and **H** is Hanning filter size in postprocessing. Bars outlined in blue are the lowest mean and maximum values per graph. Bars outlined in red are the highest mean and maximum values per graph. The numbers above highlighted bars reflect the precise value (either L1 error or % error reduction.

## 5.6 Discussion

The filter size and stride had the most effect on the gridding artifact and network estimation accuracy, with a 1x1 filter size and stride with LReLu yielding the best outcome. These hyperparameters were used in a cGAN trained on other motion types to further develop one branch of the processing pipeline, contributing toward the overarching objective of this thesis. For this reason, an abstract accepted for a power pitch presentation at the 2024 International Society for Magnetic Resonance in Medicine annual meeting [161] is attached in the appendix (Appendix: A). This method improves prediction accuracy and reduces computational cost through so-called pointwise convolution [162].

The increase in gridding artifact contrast observed when reducing the filter size from 4 to 3 can be attributed to the reduction in the effective receptive field per convolutional layer. While smaller kernels preserve local detail, their limited spatial coverage prevents the network from adequately smoothing or contextualizing features across larger regions. When paired with strided operations (stride = 2), this results in downsampling with minimal overlap, thereby introducing sharp transitions between activation regions (ie. gridding artifacts). These transitions manifest in ilSAR maps. The increased artifact contrast, despite a modest reduction in L1 error suggests that numerical error alone does not fully illustrate the quality of the output.

Smaller filter sizes (e.g., 3x3, 2x2, or 1x1) reduce computational complexity by decreasing the number of parameters, but they also reduce the receptive field, meaning they learn less contextual information from the input image. While this can be beneficial for learning intricate details, it can also make the network more sensitive to artifacts, including gridding. In the experiment described, halving the filter size to 2x2 while maintaining the stride at 2x2 did alter the appearance of the gridding artifact but did not eliminate it, suggesting that the network still struggled with capturing global structures or contextual information. Moreover, the network-estimated images contained dramatic artifacts and distortions. The presence of gridding artifacts in the generated images may reflect the network's inability to fully understand or preserve the larger-scale spatial relationships in the data, even when computationally efficient parameters were used. It could be inferred that the gridding artifact is a sign of limitations in the network's ability to generalize or model global structures, which may relate to the choice of smaller filter sizes and strides.

A stride of 1x1 can preserve spatial resolution. The lack of a gridding artifact with a 1x1 stride shows that this stride helped to avoid issues related to misalignment or uneven overlap, which can exacerbate gridding. However, despite this improvement in artifact elimination, the network's L1 error between predicted and ground-truth images hardly improved when combined with a 3x3 filter size, if at all. This indicates that while the gridding artifact was addressed, the overall accuracy of the model did not necessarily improve, and the network might still struggle to generate more accurate outputs. The gridding artifact may be a reflection of spatial misalignment or insufficient feature learning in the network, but its absence does not always correlate with improved overall performance, evaluated empirically. Network estimation accuracy might even be worse for the 3x3 filter size, because if the overall error is the same and the high intensity pixels are less prominent, it is an indication that the background error is likely higher. Since it is evident that the high intensity pixels are numerical error but the background error is the estimation mismatch, perhaps these hyperparameter choices are particularly unfavorable.

The Hanning filter was not an effective solution to eliminating the gridding artifact while maintaining high quality network estimations. The high intensity pixels observed in the error images are problematic for the current task of predicting the effects of motion on ilSAR distributions, since the success of the overall research pipeline relies on the network-estimated ilSAR distributions to be subsequently passed through a pTx protocol. The high intensity pixels may mislead the pTx script to perceive those areas to contain extremely high peak localized SAR values, even though they are artificial and unrelated to the actual computed local SAR values.

The misalignment between training and testing body models led to higher error values and the presence of gridding artifacts, particularly when testing on Duke's model, whose nose extended beyond the spatial limits of the training data. When body models were better aligned (e.g., using Duke, Fats, and Billie for training, Glenn for validation, and Ella for testing), the overall error was lower, but the artifact persisted. The alternative combination with Billie used for testing eliminated the artifact. The absence of gridding artifacts in this configuration reflects the network's improved ability to handle spatial alignment and anatomical variation across different models. The gridding artifact reflects issues with spatial misalignment between training and testing data; proper alignment significantly reduces these artifacts. This also points to a limitation of the network's ability

to generalize across anatomies, which is highly problematic for the overall goal of the thesis objective. The network's generalizability limitation can likely be alleviated by the introduction of more body models with more varied anatomies, thereby indicating an overall limitation present throughout the thesis: the availability of an insufficient number of body models and therefore varied anatomies for the most effective solution to SAR-motion problem. Since more body models do exist in the Virtual Population available for Sim4Life applications, generalizability can be improved by means of obtaining more research funding to gain access to the remaining body models: Eddie, Eartha, Charlie, Nina, Roberta, Theloneous, Louis and Jeduk. Unfortunately more limitations arise from introducing more body models over time, as the Sim4Life software undergoes yearly updates and simulation-related irregularities can occur and subsequently impact cross-model consistency. Concerns regarding inconsistencies already affected the pipeline and resulted in a conscious decision to not apply the Dizzy and Yoon-sun body models.

The LReLU outperformed the ReLU in this particular network architecture for improving estimation accuracy. While the choice of activation function may influence model accuracy, it does not directly impact the emergence of the gridding artifact, though it does affect the manner in which it manifests. The results imply that the lower L1 error reduction and the presence of gridding artifacts may be influenced by many different factors.

Despite the increase in training epochs, the gridding artifact remained, though the L1 error between the predicted and ground-truth images was reduced. Additional training time provided more opportunities for the network to refine its parameters, but it did not fully eliminate the artifact, indicating that model architecture, data representation, or the specific body model configurations used may have contributed to the persistence of the gridding artifact. The gridding artifact is not solved by additional epochs alone, and may suggest the need for more nuanced model adjustments or training strategies.

A notable limitation during this investigation was that networks configured with $4\times4$ filters and $1\times1$ strides, as well as filters of size $5\times5$ and larger with $2\times2$ or $1\times1$ strides, could not be executed successfully. These configurations were extremely computationally inefficient. As a result, the cGAN code halted and crashed due to memory overload, despite the availability of high-performance GPUs (NVIDIA DGX-100) for processing. For this reason, computational effi-

ciency is emphasized throughout the text, and the effects of larger filter sizes and more than 160 epochs are not reported.

Another limitation is that not all investigations were combined comprehensively. In other words, all of the hyperparameter tuning was conducted on the body model configuration where Ella was used for the testing data. A more comprehensive study would have applied all possible hyperparameter alterations to all possible body model configurations.

## 5.7 Conclusion

A variety of parameters which contribute to and eliminate cGAN-created gridding artifacts in local SAR matrices were tested. The results of this work can further inform researchers working with computer vision whose results are affected by gridding artifacts.

The gridding artifact may be a useful indicator of the network's performance, especially in terms of its ability to model spatial relationships, generalize across different data configurations, and learn effective representations of the input data. Throughout the experiments, the gridding artifact was influenced by factors such as filter size, stride, postprocess filtering, and body model configurations. Its presence was often accompanied with poor model performance, indicating that the network struggled with spatial misalignments or failed to generalize effectively. In contrast, the elimination or reduction of the gridding artifact corresponded with better generalization, lower error rates, and more accurate image generation. However, the absence of the artifact did not always equate to better empirically-defined model performance, indicating that other factors are relevant to improving overall network results.

Importantly, this investigation's contribution is that architectural modifications aimed at reducing computational load, such as smaller filters or strided convolutions, may inadvertently worsen perceptual quality. While error metrics offer convenient proxies for model performance, visual and structural fidelity should also be considered.

The gridding artifact is a pertinent diagnostic tool for evaluating the performance of a cGAN. Reducing or eliminating this artifact through model and data configuration might improve the network's accuracy and robustness.

The most prominent discovery from this investigation is that the cGAN is not the optimal tool to predict the effect of motion on ilSAR distributions. For this reason, the following chapters describe the transition to a more simple U-Net network architecture.

# Chapter 6

# Using U-Nets to predict the effects of head motion on simulated specific absorption rate during ultra-high field magnetic resonance imaging with parallel transmission

## 6.1 Foreword

This chapter is a copy of a manuscript that has been conditionally accepted by Magnetic Resonance in Medicine (MRM) for publication (at the time of submission, the manuscript is undergoing its first round of revisions). The only changes to the originally submitted manuscript are that it has been reformatted for LaTeX, the reference, figure and equation numbers are changed because they are adapted for the entire thesis, and the MRM submission format has not been adhered to. The appendix submitted to MRM alongside the manuscript is included in this chapter directly, rather than in the Appendix section of the whole thesis. This manuscript was co-authored with Alix Plumley, Alper Gungor, Shaihan Malik, and Emre Kopanoglu. Katherine Blanter is the first author.

Alix Plumley wrote the original pipeline that was adapted for the following in-

vestigations. Plumley's pipeline included the preprocessing, training, evaluation, and postprocessing procedures. The scripts were altered as deemed appropriate by the author. Error metrics were altered.

Alper Gungor advised on the network architecture and data preprocessing methods.

The data used in this chapter was simulated and the codes written to process the simulation data were written by Emre Kopanoglu. The scripts were used in their original, unaltered form.

The suggestion to forward map Q-matrices to local SAR matrices was put forward by Shaihan Malik. The algorithm used to do so was published in Ref. [132].

All authors proofread, edited, and signed off on the manuscript before submission to MRM.

## 6.2 Abstract

Purpose: Ultrahigh-field MRI requires careful management of the specific absorption rate (SAR), which is subject-position dependent. Within-scan subject-motion may exacerbate local SAR exposure, necessitating large safety margins to prevent SAR-underestimation, which hampers imaging performance. This study proposes a U-Net architecture to adapt safety calculations to motion as it happens, to facilitate high-performance scanning without compromising safety.

Methods: Electromagnetic simulations were performed for five body models at multiple positions with an 8-channel parallel-transmit coil. Q-matrices were transformed into real-valued SAR distributions—to train U-Nets to estimate motion-related effects on local SAR—which were then mapped back to Q-matrices. Separate U-Nets were trained for different types of body motion (rightward/ leftward/ anterior/ posterior/ yaw), which were then cascaded to predict the effect of composite (off-axis) and larger displacements on SAR. Finally, network-estimated local SAR distributions were compared with ground-truth after-motion local SAR for realistic parallel-transmit pulses.

Results: Subject motion had a statistically-significant effect on local SAR, but network-estimated safety models recovered a faithful representation of the ground-truth after-motion local SAR. For the investigated parallel-transmit pulses, the

proposed approach reduced the safety margin from 2.14-fold to 1.3-fold, and ensured more than 68% of the imaging performance could be realized while a safety model that includes all simulated subject positions would have limited scanning performance to as low as 21% of the maximum.

Conclusions: The proposed position-aware SAR calculation approach allows smaller safety margins, which has the potential to enable higher-performance UHF MRI scanning without compromising safety for subjects who are unable to remain still.

### 6.2.1 Key words

Parallel transmit (pTx), specific absorption rate (SAR), Deep Learning, MRI Safety, Ultrahigh field MRI

## 6.3 Introduction

Advancements in MRI applications have led researchers to ultra high field (UHF) strengths since they offer improved SNR and contrast to noise ratio (CNR). Improved SNR and CNR allow images to reveal finer anatomical details and subtler physiological effects [163], thereby improving MRI's overall utility. However, the shorter RF wavelength at UHF MRI makes it comparable to or shorter than the body part being imaged [164]. This results in transmit field (B1) inhomogeneity, leading to images containing tissue-unrelated intensity variations that may overshadow anatomical information, or complete signal drop outs [165]. Although transmission field inhomogeneity can be improved with adiabatic pulses, these pulses are much longer and come with a penalty of increased specific absorption rate (SAR) [166]. SAR is an intermediary parameter that quantifies the power absorbed in the tissue during MRI which can be used as a guide to subsequent tissue heating. It is used as one of the main safety parameters in MRI to ensure subject safety [25] [167] [168] [169]. B1 inhomogeneities can be addressed with parallel-transmit arrays (pTx-array) that provide simultaneous control of multiple transmit coils that are driven by tailored radiofrequency pulses [170] [171]. The benefits of pTx-arrays are complicated by the possibility of inadvertently creating increased localized SAR in unpredictable locations due to constructive interference of electric fields (E-fields) from the independently controlled pTx coils [172]. This has led to safety concerns, as localized SAR limits can be reached with less power input than global SAR [173].

E-fields generated within tissues during MRI cannot be measured *in vivo*, and therefore, numerical simulations based on generic body models are used to evaluate the E-field distributions responsible for local SAR [174]. The interaction of E-fields from independent channels can be represented in Q-matrices [41, 175]. Since Q-matrices contain tens of thousands to millions of data points, in clinical and research practice, they are reduced to a representative subset, called virtual observation points (VOPs) [176], which never underestimate SAR and only overestimate SAR by a limited factor. VOPs that represent multiple body models simulated at multiple positions and orientations in a coil can also be developed.

Concerns regarding increased local SAR in unpredictable locations are exacerbated with the prospect of unavoidable and unplanned subject motion and change in position within the coil after scan registration and calibration [177–180]. For example, when evaluating all six degrees of motion at 7 T using pTx excitation, Kopanoglu et al. reported that peak spatial local SAR (psSAR) can change location as well as increase by up to 210% because of subject motion [177]. Since sedation is ethically complicated, invasive, and limiting to scantime task protocols during fMRI, it is not a preferable option.

In practice, safety models used on the MRI systems for real-time safety calculations are unaware of the actual subject position. In this case, safety models that are not sufficiently representative of possible subject positions may substantially underestimate SAR and lead to safety hazards in the presence of motion, and alternatively, safety models that contain all possible subject positions may be unnecessarily cautious and impede the full potential of UHF MRI in the absence of motion. To evidence this concern, Kopanoglu showed a mismatch between the position of the safety model being used and the actual subject position also influences SAR, with up to 5.2-fold SAR underestimation shown for both single-channel and parallel-transmit pulses, and for different slice orientations [178]. Therefore, it can be concluded that safety models that can adapt to the subject in case of motion are desirable to ensure safety while maximising scan performance.

Recent studies have utilised the computational efficiency offered by artificial intelligence towards realising tailored safety models; Brink et al. [86] used convolutional neural networks to map T1-weighted data to a local SAR distribution whereas Meliado et al. [39] used subject-specific $B_1^+$ maps in a similar attempt. However, neither study considered the effect of motion on local SAR.

Subject motion also affects transmit coil sensitivities and therefore flip-angle distributions ($B_1^+$ maps) [47], and although $B_1^+$ maps can be measured using MRI, in contrast to E-fields which cannot, remeasuring $B_1^+$ maps as subject motion happens is not practical. Therefore, Plumley et al. [8] proposed to use deep learning (DL) to predict the effect of motion on $B_1^+$ maps. The study demonstrated that by training separate networks for a subset of motion types and amounts, and cascading these networks to approximate the actual motion, changes in $B_1^+$ maps can be successfully estimated.

In this work, we use a similar approach to [8], but instead use DL to predict the effect of motion on simulated local SAR distributions. For this purpose, multiple realistic body models were simulated at multiple positions within a generic 8-channel loop array. Paired and labelled images of local SAR distributions preceding and following motion were used to train U-Nets [2]. DL-predicted local SAR distributions were compared to ground-truth counterparts to evaluate prediction quality. Finally, predicted local SAR distributions were converted back to Q-matrices to calculate predicted local SAR distributions for realistic pTx pulses and compared to ground-truth local SAR.

## 6.4 Methods

### 6.4.1 Simulations

Simulations were carried out similarly to published work [177] [178] [8]. Briefly, electromagnetic (EM) simulations were conducted on five Virtual Population models—Billie, Duke, Ella, Fats, and Glenn (IT'IS Foundation, Zurich, Switzerland)—at one central and 14 off-center positions within a generic 8-channel pTx coil (coil element height, 110 mm; width, 40 mm; microstrip width, 3 mm) tuned to 7 T (295 MHz) in Sim4Life (ZMT, Zurich, Switzerland). The off-center positions included rightward (R) shifts of 5, 10, and 20 mm and posterior (P) shifts of 5 and 10 mm, along with all possible combinations thereof, as well as 5°, 10°, 15° rotations around the Yaw axis. As recommended in the literature, the neck and shoulders were included within the computational domain [53]. To ensure consistent tissue voxelization, the coil array was moved with respect to the body, similar to Refs. [177] [178] [8]. This meant that all field distributions were co-registered. Three-dimensional E-field distributions and current densities were mapped onto a predefined grid (size: 180 × 215 × 250 mm, resolution: 1.5 × 1.5 × 1.8 mm),

and exported to MATLAB (The MathWorks, Natick, MA imported to MATLAB (The MathWorks, Natick, MA), where 10-gram averaged Q-matrices were calculated using cubical volumes [25].

### 6.4.2 SAR Calculations

#### Forward-mapping

The local SAR distribution in voxel $r$ was calculated from Q-matrices using:

$$SAR[r] = Re\{\mathbf{w}^T \cdot \ \mathbf{Q}[r] \cdot \mathbf{w}\} \tag{6.4.1}$$

where $w$ denotes channel coefficients, the superscript $T$ denotes complex-conjugate transpose, $\mathbf{Q}[r]$ denotes the complex-valued NxN Q-matrix that characterizes the interaction of the coil elements in voxel $r, N$ is the number of coil elements, and $Re\{\cdot\}$ extracts the real-valued component of its input.

To avoid estimating the complex-valued entries of the Q-matrices, we note that these matrices can be characterized by a series of $N^2$ real-valued projections, which we refer to as intermediate local SAR (ilSAR) distributions. This method was proposed by Zhu et al. [132] to enable estimation of Q-matrices from (real-valued) empirical power measurements. For an $N$-channel coil, the first $N$ terms are single-coil distributions given by:

$$ilSAR_m^{c_m}[r] = Re\{\mathbf{w}^T \cdot \ \mathbf{Q}[r] \cdot \mathbf{w}\} \tag{6.4.2}$$

where

$$\mathbf{w}[k] = \begin{cases} c_m, & k = m \\ 0, & else \end{cases} \tag{6.4.3}$$

The other $N(N-1)$ ilSAR distributions are when only two coils are on and the rest are off:

$$ilSAR_{m,n}^{c_m,c_n}[r] = Re\left\{\mathbf{w}^T \cdot \mathbf{Q}[r] \cdot \mathbf{w}\right\} \tag{6.4.4}$$

where

$$\mathbf{w}[k] = \begin{cases} c_m, & k = m \\ c_n \text{ or } \widetilde{c_n}, & k = n \\ 0, & else \end{cases} \tag{6.4.5}$$

and $c_m$ and $c_n$ are coefficients of channels $m$ and $n$. In this paper, we used $c_m = \sqrt{2}$ for single channel ilSAR and two pairs of channel coefficients $(c_m, c_n)$=(1,1) and $(c_m, \widetilde{c_n})$=(1,$i$) for two-coil interactions.

**Reverse-mapping**

The estimated ilSAR distributions were converted to estimated Q-matrices, denoted by ${}_{\text{estim}}Q_{m,n}$ and calculated via:

$${}_{\text{estim}}Q_{m,n} = \begin{cases} \dfrac{{}_{\text{estim}}\text{ilSAR}_m^{c_m}}{2}, & m = n \\ \dfrac{1}{2}\,{}_{\text{estim}}\text{ilSAR}_{m,n}^{c_m,c_n} + \dfrac{-i}{2}\,{}_{\text{estim}}\text{ilSAR}_{m,n}^{c_m,\widetilde{c_n}} + \dfrac{-1+i}{4}\,{}_{\text{estim}}\text{ilSAR}_m^{c_m} + \dfrac{-1+i}{4}\,\text{ilSAR}_n^{c_n}, & m < n \\ \dfrac{1}{2}\,{}_{\text{estim}}\text{ilSAR}_{m,n}^{c_m,c_n} + \dfrac{i}{2}\,{}_{\text{estim}}\text{ilSAR}_{m,n}^{c_m,\widetilde{c_n}} + \dfrac{-1-i}{4}\,{}_{\text{estim}}\text{ilSAR}_m^{c_m} + \dfrac{-1-i}{4}\,\text{ilSAR}_n^{c_n}, & m > n \end{cases} \tag{6.4.6}$$

where $m$ and $n$ are channel indices, the superscripts denote the channel coefficients $(c_m, c_n) = (1, 1)$ and $(c_m, \widetilde{c_n}) = (1, i)$ for two coil interactions, which yields ${}_{\text{estim}}\text{ilSAR}_n^{c_n} = {}_{\text{estim}}\text{ ilSAR}_n^{\widetilde{c_n}}$, and $c_m = c_n = \sqrt{(2)}$ for single channel ilSAR cases. Eq. [A1] in the Manuscript Appendix provides the general solution, which reduces to Eq. [6.4.6] for the coefficients used in this study.

### 6.4.3 Dataset Preparation

To prepare the data for processing via neural networks, ilSAR data were normalized by the overall maximum value across all body models and positions and interpolated to 256x256 resolution. Data were processed slice-by-slice, with all 64 combinations of the 8-channel pTx coil concatenated in the third dimension, yielding data size of 256x256x64 for training neural networks per slice. The number of axial slices which contained tissue data varied between 126 and 136 between body models due to different body model sizes.

Because SAR / Q-matrix / ilSAR distributions show similar variations in adjacent slices, splitting data into training, validation and testing datasets at a slice level

can introduce crosstalk between the datasets and produce misleading results. Therefore, datasets were split at the body model level, with Ella, Fats and Glenn forming the training, Duke the validation, and Billie the testing datasets. This approach ensures that the testing dataset is completely unseen by the networks during training. Moreover, this scenario is realistic, since the neural networks would be expected to be used on unseen subjects in practice.

To create the training datasets, data from before and after motion were paired for all combinations of positions that yielded the same relative displacement; for example, for rightward 5 mm motion, both pairs of positions (R0, R5) and (R5P5, R10P5) were included in the training dataset ("R5" denotes 5 mm rightward displacement while "P5" denotes 5 mm posterior displacement). During the initial stages of development, we discovered that due to the smooth and low-frequency spatial variations of ilSAR distributions, U-Nets were susceptible to over-fitting. To reduce over-fitting, we only included every third slice in the training dataset (and discarded the rest), yielding 1,560 cases for posterior, 1,300 cases for rightward motion, and 402 cases for yaw rotation. All networks were validated with 135 and tested on 136 pairs of ilSAR distributions.

Training a separate network for each type and amount of motion is unrealistic (discussed below). To test network performance outside the amount of displacement networks were trained for (e.g. R5), we paired all simulated positions with the centred position as reference (e.g. R10-vs-R0), instead of pairing positions that were 5 mm away from each other like in the training dataset (e.g.R10-vs-R5). This enabled testing the networks with larger displacements (e.g. R20 mm) by cascading the networks as necessary.

### 6.4.4 Investigations

**Rightward and posterior:** Two separate but architecturally identical neural networks were trained on R5 mm and P5 mm displacements.

**Off-axis displacements and displacements beyond 5 mm:** It is not practical to train separate networks for all possible motion types and amounts. Therefore, we cascaded networks as necessary to investigate evaluation performance for larger displacements and off-axis displacements, similar to Ref. [8] (Figure A.0.1, panel A). To cascade, the output of one evaluation network provided the input

for a subsequent network until the target evaluation was reached. For example, four copies of an R5 mm model needed to be cascaded to create an evaluated R20 mm output (R5R5R5R5), and copies of both networks were cascaded to create an estimated R20, P10 mm output (R5R5R5R5P5P5). We investigated prediction performance at all simulated positions previously listed in Dataset Preparation.

**Leftward and Anterior:** We also investigated translations paired in the opposite directions: leftward and anterior. For this purpose, we used the same dataset but with reversed pairing, ie. R5 mm labeled as the reference position for R0 mm, emulating leftward L5 mm movement (similarly for posterior and anterior).

**Yaw:** Yaw 5° networks, which had the same network architecture, were investigated with identical cascading mechanism. We did not include pitch or roll because previous work [177] determined that local SAR varies most with translation. Yaw is included only to show that while yaw-induced error was less significant, our method is suitable for rotational motion.

**Evaluating the effect of cascading on network prediction error:** We investigated the effects of cascading by evaluating the error produced by

A) *evaluating estimation error floor:* By cascading different combinations of networks that summed up to zero displacement, such as R5L5, we evaluated the estimation error floor.

B) *changing the order of the networks being cascaded:* By comparing different cascading orders that sum to the same total displacement, such as R5R5P5P5, R5P5R5P5 to estimate the changes following R10, P10mm displacement.

C) *increasing the number of cascades:* Comparing nRMSE from cascading P5mm twice to reach P10mm to nRMSE from a network directly trained on P10mm which did not require any cascading.

D) *off-grid displacement:* Evaluating whether having a 1mm difference in actual motion affects network prediction accuracy. For this we compared the R5 network predictions with R4 ground truth.

**4-fold cross-validation of body models:**To investigate the robustness of net-

work performance against variations in body mass index (BMI) and other potential confounds, we included a 4-fold cross validation investigation similar to[21,22]. For this purpose, we shuffled the order of the body models used to create the training, validation, and testing datasets. In addition to the previous testing performed on a female pre-adolescent body model (Billie), these investigations tested the networks on obese adult male (Fats), elderly adult male (Glenn), and adult female (Ella). Body model BMIs ranged between 15.4 and 36, and are listed in Figure 4A. This investigation was conducted on P5 and P10 motion. From here onward, the configurations will be named with the first letter of the body model name listed in order of training (three models), validation (one model), and testing (one model); e.g., EFG-D-B for training: Fats/Ella/Glenn, validation: Duke, testing: Billie. The investigations here were conducted for EBG – D – F, FED – B – G, FDB – G – E, whereas the other investigations in this manuscript are for EFG-D-B.

**Comparing dataset preparation methods:** We chose to separate our training, validation and testing datasets by body model (from here referred to as the leave-one-out approach) to ensure that the testing dataset was unseen by the evaluation network. We investigated splitting the datasets slice-wise like Ref.[28] by splitting the whole dataset as 2/3 training, 1/6 validation and 1/6 testing, and compared P5 and P10 results obtained with the two methods.

### 6.4.5 Network Structure

The neural network architecture was a U-Net (Figure 6.4.1), implemented in TensorFlow [181] version 2.4.1. The U-Net contained five downstack layers joined by skip connections to four upstack layers. The activation functions were leaky-ReLu for the convolution layers in the encoder path and ReLu for the deconvolution layers in the decoder path. The first three deconvolution layers were joined by dropout layers (dropout rate, 0.5). Other parameters were selected as: filter size, 3x3; stride, 2x2; learning rate, 2e-4, and $\beta 1$ in the Adam optimizer [135] as 0.9. The number of filters changed between 128 in the first and last layers and 1024 in the deepest layer (Figure 6.4.1).

#### Training

We reached optimal training performance at 40 epochs. The networks took approximately 4 hours to train on a NVIDIA DGX v-100 GPU.

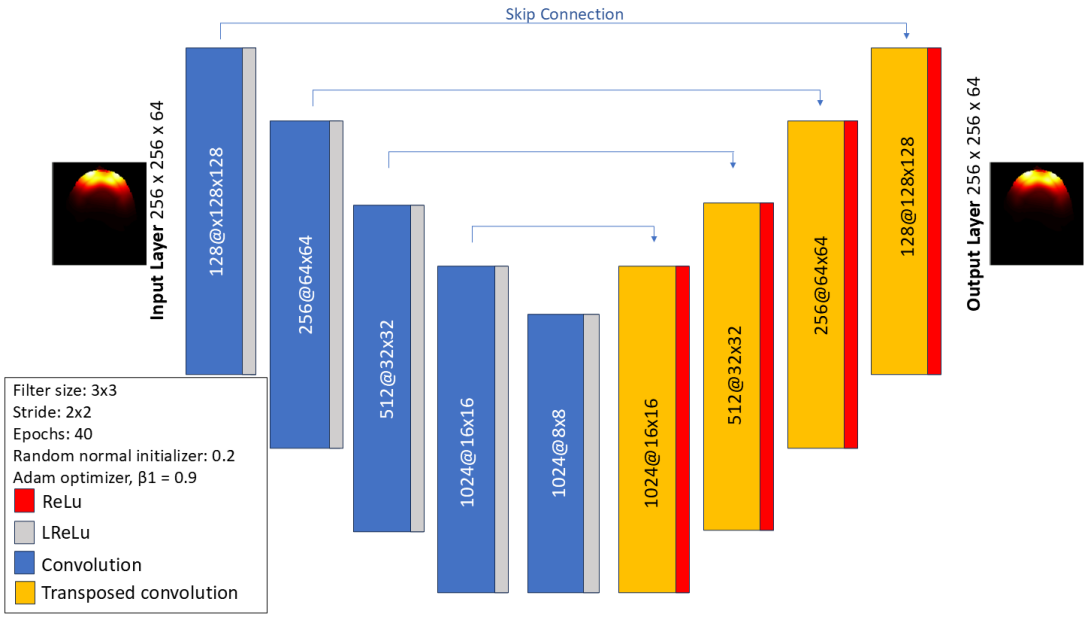


Figure 6.4.1: The neural network architecture. Five downstack layers are joined by skip connections (blue arrows) to four upstack layers. Activation functions were leaky-ReLu for the convolution layers in the encoder path and ReLu for the deconvolution layers in the decoder path.

### 6.4.6 Postprocessing and Evaluation

Once the A5, P5, L5 and R5 mm, and Y5° networks were trained separately, they were evaluated using the testing dataset from the Billie model. Predicted ilSAR distributions were exported to MATLAB and masked to remove the background. To remove the estimation noise in regions where initial field values are minute, distributions ¡1% were also interpolated and smoothed using a Gaussian kernel, after which larger magnitudes were restored to their correct values.

Network prediction quality was quantified at each position using normalized root-mean-squared error (nRMSE), between ground-truth simulations and network estimated ilSAR:

$$\mathrm{nRMSE} = 100 \times \frac{\sqrt{\frac{1}{N_r}\sum_r^{N_r}\left({}_{\mathrm{estim}}\mathrm{ilSAR}_{m,n}^{c_m,c_n} - {}_{\mathrm{GT}}\,\mathrm{ilSAR}_{m,n}^{c_m,c_n}\right)^2}}{\sqrt{\frac{1}{N_r}\sum_r^{N_r}\left({}_{\mathrm{GT}}\mathrm{ilSAR}_{m,n}^{c_m,c_n}\right)^2}} \tag{6.4.7}$$

Where ${}_{\mathrm{estim}}\mathrm{ilSAR}_{m,n}^{c_m,c_n}$ and ${}_{\mathrm{GT}}\mathrm{ilSAR}_{m,n}^{c_m,c_n}$ are the estimated and ground-truth ilSAR distributions for a given channel combination. The effect of motion was similarly quantified by calculating the nRMSE between ground-truth simulations

and initial position local ilSAR.

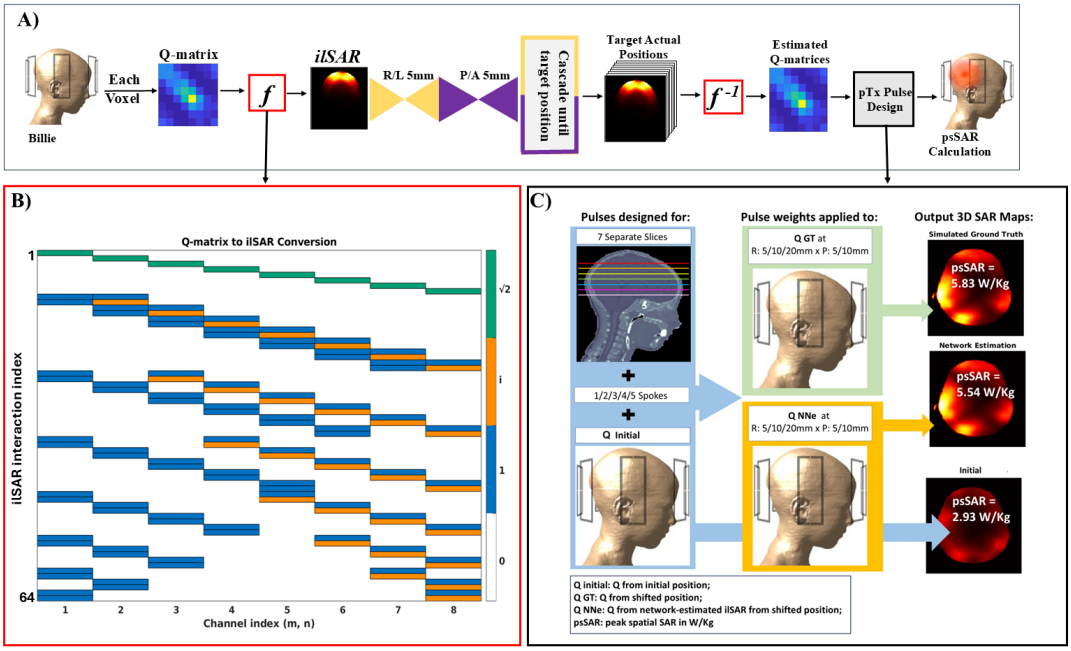


Figure 6.4.2: The pipeline illustrated from body model simulation through local SAR calculation. (A) The overall methodology: Q-matrices are extracted from simulated body models, fed through an algorithm to calculate ilSAR, and passed through a neural network to estimate ilSAR at an indicated shifted position. The estimated ilSAR is remapped to Q-matrices which are fed through a pulse design algorithm which indicates peak spatial SAR values for the network-estimated, ground-truth, and initial positions. (B) A visualization of the algorithm designed to calculate intermediate local SAR (ilSAR) values given Q-matrices. Top eight: single-channel interactions: Bottom section: Two-channel interactions: , starting with adjacent channels followed by 1-apart, 2-apart, and so on. Colors indicate channel combination weights. (C) pTx pulses are designed for the initial position with 1/2/3/4/5 spokes for 7 slices. The SAR distribution at off-center positions are calculated using the same pulse weights and resulting psSAR values are compared. Example case shown for a 2-spokes pulse evaluated after a R:20 mm, P:10 mm displacement.

### 6.4.7 pTx Pulse Design Application

Slice-selective pTx pulses with 1, 2, 3, 4, or 5-spokes were designed to create homogeneous in-slice excitation at the centred position [47, 48]. Pulses were designed for seven different slices from cerebellum to crown, with 1.8 mm slice separation. During optimisation the channel-by-channel RF power was controlled using Tikhonov regularisation. Three-dimensional SAR distributions were calculated

using the centred, ground-truth off-centre and estimated off-centre Q-matrices to investigate the effect of motion on SAR, similar to the literature [177]. In other words, we assume knowledge of the ideal safety model and exclude the mismatch between the safety model and the actual body in order to focus on the effect of motion and the correction our models provide. Network estimated 3D SAR distributions were passed through a 3D binomial filter for smoothing, where the element in position $(i, j, k)$ is given by: $I_{3D}(i, j, k) = I_{1D}(i) \times I_{1D}(j) \times I_{1D}(k)$ where $I_{1D}$=[1/4,1/2,1/4].

### 6.4.8 Statistical testing

We hypothesized that subject motion has a substantial effect on psSAR which can be corrected using neural networks. We tested the hypothesis using a two-sample Kolmogorov-Smirnov test [182, 183] with a significance level p=0.05 on psSAR values from 35 pulses that were designed for the centred position and evaluated at 11 other positions each, yielding 385 samples.

## 6.5 Results

### 6.5.1 Network Estimation Quality

Figure 6.5.1 shows example ilSAR distributions for four different off-centre positions. For each position, two channel combinations are presented, one that yields the worst estimation error and one that yields the worst motion error. In all cases, including those selected to show the worst estimation performance, the networks clearly reduce motion-related SAR calculation errors.

Across the 136 slices of a single body model, the networks took an average of $4^{+}_{-}3$ ms per slice to estimate the ilSAR for all 64-channel combinations.

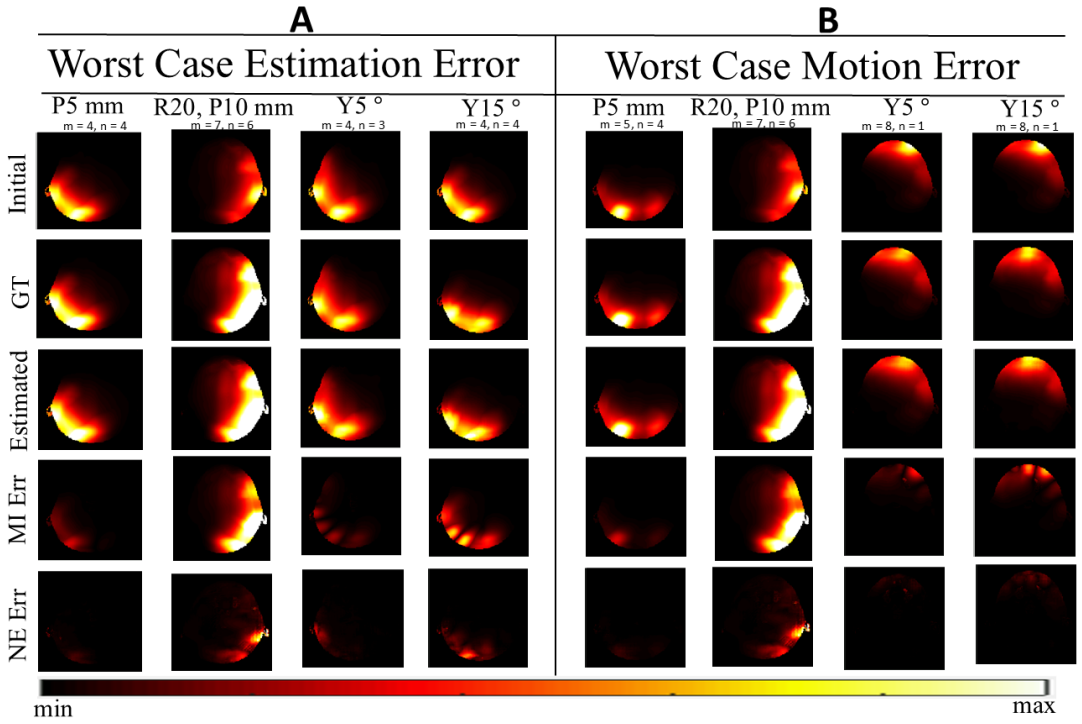


Figure 6.5.1: SAR distributions before motion, after motion, and after estimation are shown for four motion states. Section A of the figure shows the channel combinations that led to the largest network estimation error, while the right half (B) shows combinations that led to the worst-case motion error. Each column contains maximum intensity projections along the z axis. Interacting pTx channel indices and are listed for each column. In order, the rows show the given reference image, followed by the ground-truth and network estimated images. The bottom two rows show motion-induced and network-estimation error. Colour ranges are consistent within each column with corresponding first row. SAR maps are displayed in the body model's coordinate system due to the co-registration procedure described in the methods section detailing the model simulation protocol.

For all cases investigated, networks reduced motion error considerably. Figure 6.5.2 shows nRMSEs averaged across all 136 slices and 64 channels for each motion case for the Billie testing dataset. Network estimations reduced the motion-related ilSAR from a worst-case of 67.2% to 14.5%, and an average of 37.9% across all positions to 7.8% for translations in the axial plane. For rotations in yaw, network estimation reduced ilSAR nRMSE from a worst-case of 44.9% to 13.7%, and an average of 32.0% across all rotations to 9.9%. A paired t-test showed that all reductions were statistically significant ($p = 0.01$).

Figure 6.5.2 shows that the motion-related error in SAR increases monotonically with increasing displacement (translational or rotational) from the initial posi-

tion, with the only exception being the error for A10 being larger than L10A5. Furthermore, it was observed that displacements in anterior and posterior directions yielded more error than rightward and leftward displacements of equivalent distances; i.e., the error for A10L5 was larger than A5L10.

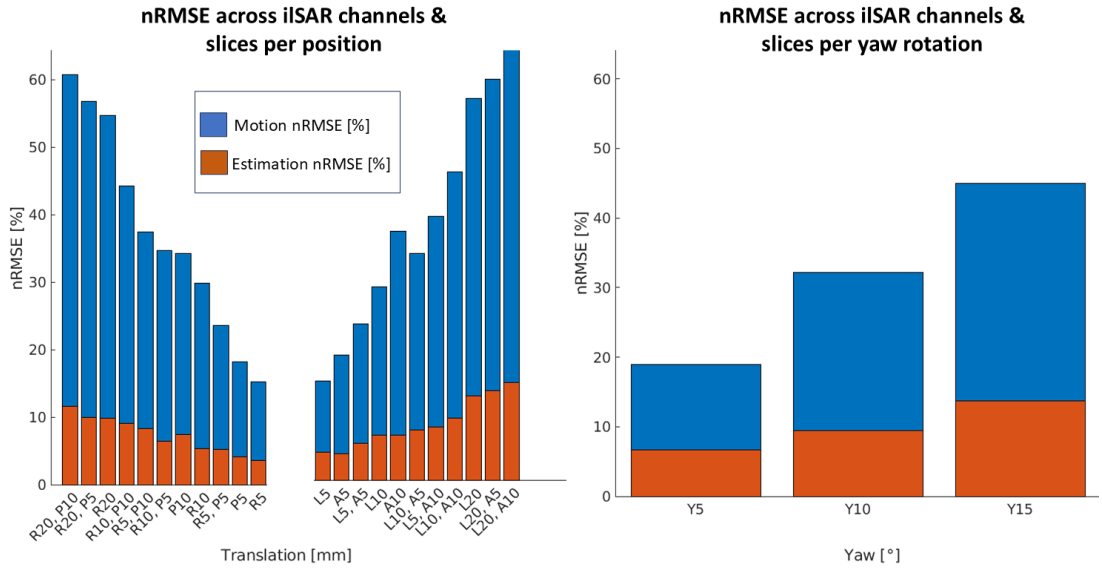


Figure 6.5.2: Motion-related and network predicted nRMSE (%) values, averaged across slices and channels, are shown for each degree of motion. R: rightward, L: leftward, A: anterior, P: posterior, Y: yaw. Values in the left panel are displayed in descending(R) / ascending(L) order of radial displacement from the center of the RF coil. For equivalent radial distances, leftward and rightward increases are positioned closer to the centre than anterior and posterior; i.e., L10A5 closer to centre than L5A10.

### 6.5.2 Alternative cascading strategies

Table 6.5.1 lists all the results from different cascading strategies tested.

#### Cascading networks to go back to initial position

We investigated the inherent error introduced by the networks by investigating a series of cascades that yielded zero displacement in total and compared the resulting distributions with the initial data. The error increased with the number of networks cascaded; the cascades with 2 networks, RL and PA both yielded approximately 3.2% error. The cascades with 4 networks, RPLA, RRLL, RLLR, AAPP yielded 4.95% $\pm$0.43% (mean $\pm$ sample standard deviation) and a standard error of mean of 0.22%. For the cascade with 8 networks, RR-PP-LL-AA, the

mean error was 18.55%. Across all cases, the error introduced by a network was calculated as 1.50% ± 0.41%.

### Evaluating the effect of cascading order and inefficient cascading

We investigated the effect of the order in which networks are cascaded, for the R10-P10 mm case, by cascading the R5 and P5 networks in the following patterns: RRPP, PPRR, RPPR, RPRP. All cases yielded similar nRMSE values that ranged between 9.06% and 9.86%. We observed that cascading the R5 networks earlier in the chain yielded marginally smaller nRMSE, which may be attributed to two reasons. First, the training datasets for rightward and leftward networks were larger than those for anterior and posterior networks. Second, anterior and posterior motion yielded larger motion and estimation error than rightward and leftward displacement (Figure 6.5.2) and introducing this larger estimation error earlier in the chain likely led to larger error at the final output. Nevertheless, the influence of the order of the networks on cascading performance was small, with a sample standard deviation of 0.33% and a standard error of mean of 0.17%.

We also investigated the accumulation of error when 12 networks were cascaded instead of 4 by inserting 8 additional networks that sum up to zero displacement, between the PPRR network, yielding: PP-AAPPRRLL-RR. The additional 8 networks increased the error by 15.1% to 24.96%. This increase is in alignment with the expected 1.50% ± 0.41% increase in error-per-network from the previous investigation. To put the estimation error of 24.96% into context, the motion induced error for R10, P10 mm was still much higher at 44.25%.

### Directly training vs most efficient cascading

Next, the effect of training separate networks for larger movements was compared with cascading networks trained for smaller movements, the example case here being P10 vs P5P5. Because P10 mm data are also used in the training of P5 networks, there was more data available for training the P5 network than those for the P10 network. To reduce the influence of data size on the outcomes, we trained an alternative P5 network with a smaller dataset similar in size to that of the P10 network, dubbed here P5(smaller dataset). The default P5 network trained with more data provided the best performance with an nRMSE of 7.42%, which increased to 9.53% for the P5(smaller dataset) network. This indicates that training datasets in this study are not "over-sized" and therefore, performance of

the method can potentially be increased further with a larger dataset. Interestingly, cascading the P5(smaller dataset) networks yielded a better outcome than the purpose-trained P10 networks, which yielded 10.52% error. This could be attributed to the small difference between the datasets between the P10 (782 data pairs) and P5(smaller dataset) networks (940 data pairs). Alternatively, the 2-cascade solution providing less error than the single-network solution can indicate that networks trained on small displacements yielding small motion-error might be more stable than networks trained on large displacements that cause large motion-error. For comparison with the different cascade scenarios investigated, the motion induced nRMSE here was 34.20%.

### Off-grid Evaluations

Finally, we investigated how the R5 network results compare to R4 ground truth. In other words, we compared 5 mm predicted maps with 4 mm actual displacement maps. The difference between the R5 network estimation and the R4 ground truth was 3.83%., while the original R5 estimation error was 3.61%. The relatively small difference between the R5 network estimated and R4 ground truth images hints at the validity of the method for motion states that do not fit the simulated grid.

### 4-fold cross validation

For both P5 and P10 motion, the networks reliably reduced the error in ilSAR due to motion across all investigated motion types and body model configurations (Figure 46.5.3). For P5 motion (Figure 4B), motion-induced nRMSE averaged across slices and channels varied between 15.4% and 18.2% across the testing models, whereas the network estimations reliably reduced the error to between 2.9% and 4.7%. For P10 displacement (Figure 4C), motion-induced error was between 28.3% and 34.2%. For the three testing models with similar BMI, the error was between 5.6% and 7.4% whereas for the highest-BMI model it was 12.2%.

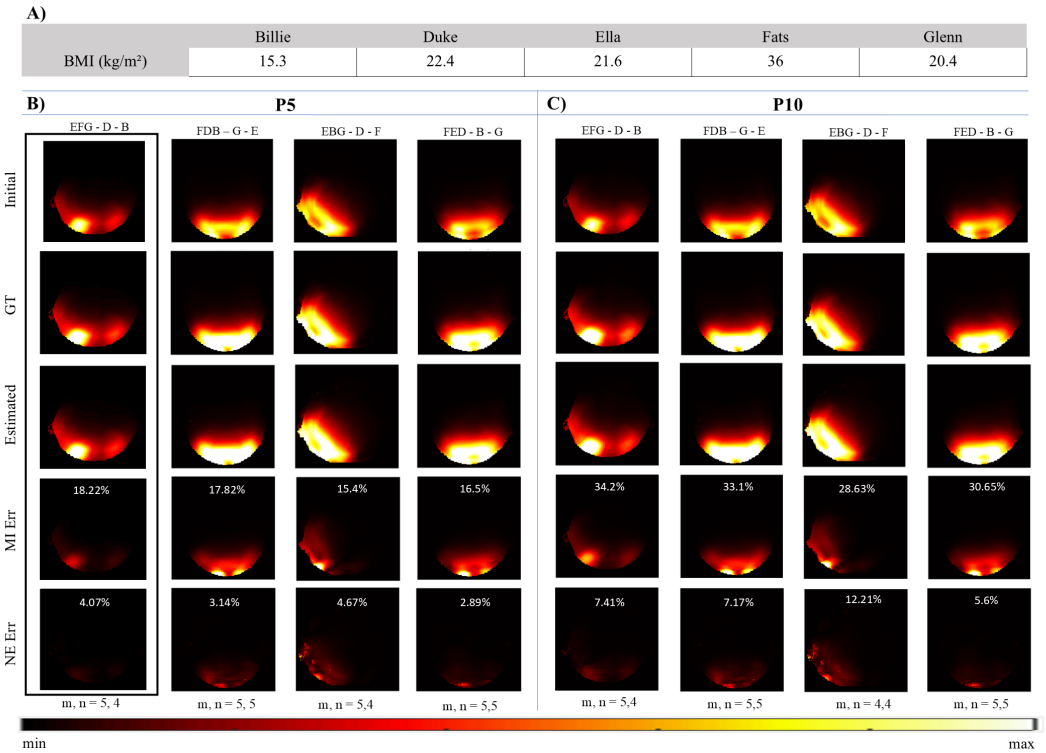


Figure 6.5.3: Cross-validation of proposed method on alternative body models. The letters F, D, G, B, E indicate Fats, Duke, Glenn, Billie, Ella, and the grouping (e.g., EFG-D-B) indicates the three models used for training (e.g., Ella, Fats, Glenn), followed by the model used for validation (e.g., Duke) and finally the testing model (e.g., Billie). A) BMI values in kg/m$^2$ per body model. B) Each column shows P5 network testing results from the body models indicated at the top. Each row, in order, is Initial (before motion), ground truth (GT; after motion), Estimated (network-estimation of ilSAR distribution after motion), motion-induced error (MI Err; —Initial – GT— ), and network estimation error (NE Err; —Estimation – GT— ). Numerical values on the error maps indicate the in-slice nRMSE averaged across slices and channels. The first column (highlighted by the rectangle) is repeated from Figure 2 for easier reference. C) Similar to panel B) but for P10 motion.

| Cascade type | Estimated nRMSE | Motion nRMSE |
|---|---|---|
| Cascade to 0 (no motion) | | |
| R5, L5 | 3.18% | N/A |
| P5, A5 | 3.19% | |
| RPLA5 | 5.56% | |
| RR5, LL5 | 4.78% | |
| R5, LL5, R5 | 4.91% | |
| AA5, PP5 | 4.55% | |
| RR5, PP5, LL5, AA5 | 18.55% | |
| Cascade to R10, P10 mm | | |
| RR5, PP5 | 9.06% | 44.25% |
| PP5, RR5 | 9.86% | |
| R5, PP5, R5 | 9.36% | |
| R5, P5, R5, P5 | 9.36% | |
| PP5, AA5, PP5, RR5, LL5, RR5 | 24.96% | |
| Cascade to P10 mm | | |
| PP5 | 7.42% | 34.2% |
| PP5 (smaller dataset) | 9.53% | |
| P10 trained directly | 10.52% | |
| R4 mm GT | | |
| R5 | 3.83% | 12.20% |

Table 6.5.1: Motion-induced and network-estimation error (nRMSE [%]: normalised root-mean-squared-error) from alternative cascading methods. R: rightward, L: leftward, P: posterior, A: anterior motion. Numerical values indicate displacement value in mm.

### Alternative data preparation method by slice-interleaving

The P5 mm network trained, validated and tested on different body models (LOO: leave-one-out) yielded reduced motion-induced error from 18.22% to 4.07% across slices and channels, resulting in a 77.66% relative reduction in nRMSE (Table 6.5.2). The alternative P5 network trained, validated and tested on interleaved slices across all body models reduced the motion error from 17.05% to 3.06%, yielding a 82.05% relative reduction in nRMSE. The difference in motion-induced error values is due to the different testing datasets in both cases. Similarly, that same network cascaded twice to P10 mm reduced the motion-induced error from 31.79% to 6.1%, indicating a 80.81% relative improvement whereas cascading the

| | P5 mm | P10 mm |
|---|---|---|
| SI Estimated | 3.06% | 6.1% |
| SI Motion | 17.05% | 31.79% |
| LOO Estimated | 4.07% | 7.42% |
| LOO Motion | 18.22% | 34.2% |
| SI Improvement | 82.05% | 80.81% |
| LOO Improvement | 77.66% | 78.3% |

Table 6.5.2: Comparison of (nRMSE [%]: normalised root-mean-squared-error) resulting from the slice interleaving (SI) and leave-one-out (LOO) data preparation methods for P5 mm and P10 mm (2 x P5 mm cascaded) networks. 'Estimated' is network estimation error, while 'Motion' indicates motion-induced error. P: posterior displacement. Numerical values indicate displacement value in mm.

network trained with the leave-one-out method reduced the motion-error from 34.2% to 7.42%, resulting in a 78.3% relative improvement. Overall, the network trained on interleaved slices yielded a better relative nRMSE improvement as expected although the improvements were small.

### 6.5.3 Pulse Evaluation

For each designed pTx pulse, three SAR distributions were calculated using centred Q-matrices (initial), off-centre Q-matrices (ground-truth), and estimated off-centre Q-matrices, and the resulting distributions and psSAR values were compared. Figure 6.5.4 compares 3D SAR distributions for an example case. Calculating the after-motion SAR distribution using the centred Q-matrices highly underestimates SAR, whereas the network-estimated safety model provides a visually nearly-identical estimation of the ground-truth SAR distribution. The psSAR values were 1.02 W/kg for ground-truth off-centre SAR, 0.70 W/kg calculated using the centred Q-matrices and 1.00 W/kg with the network-estimated Q-matrices, yielding only 2% estimation error in this instance.

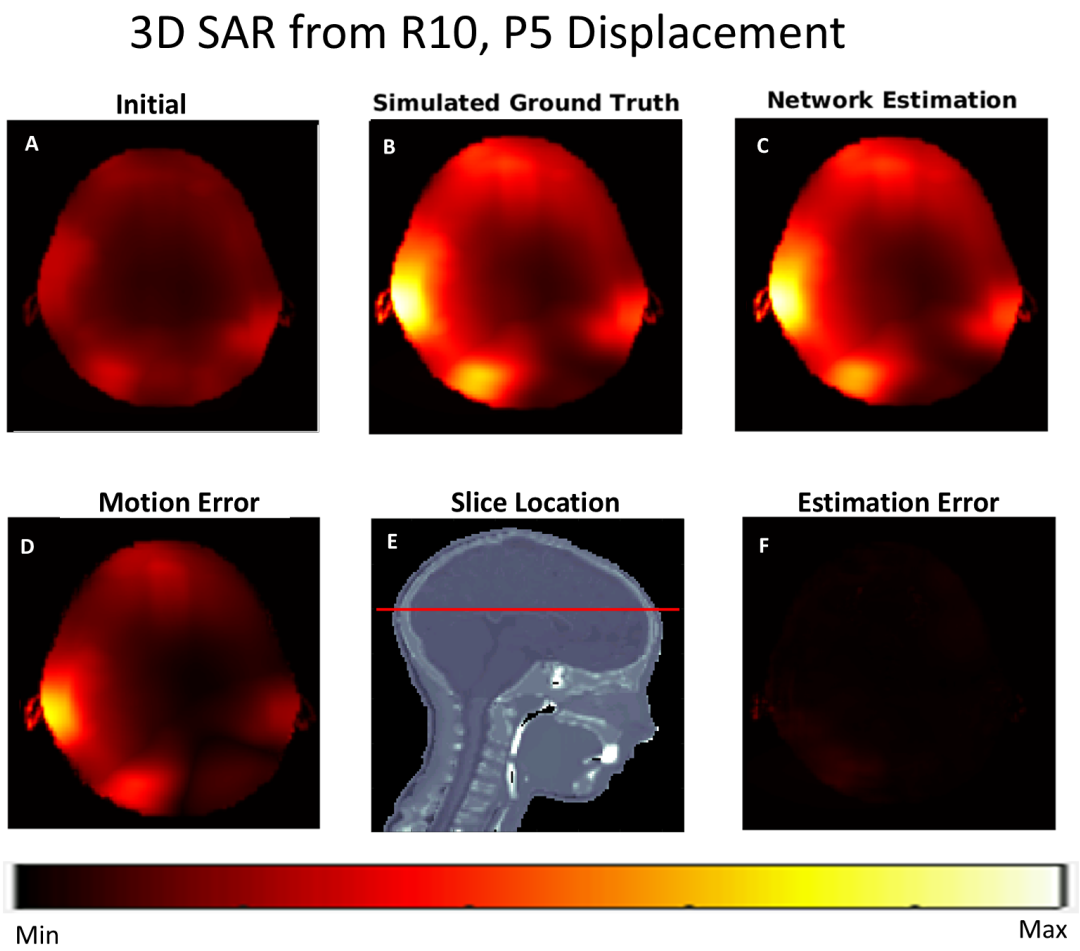


Figure 6.5.4: A maximum intensity projection along the z-axis of the 3D SAR distributions for R:10 mm P:5 mm motion.. A: Initial SAR distribution. B: ground-truth SAR distribution after motion. C: Network estimated SAR distribution. D: Motion error compared to the ground-truth. E: Location of the slice the 3-spoke pTx pulse was designed for. F: Network estimation error compared to the ground truth

The worst motion-related SAR increase was observed for R20 mm, P10 mm motion. The nRMSE between the initial and ground-truth psSAR values was 38.95% compared to 18.39% between ground-truth and network-estimation, yielding a 52.77% relative error reduction. Figure 6.5.5 a shows that the network estimations closely follow the ground-truth peak local SAR for smaller motion. The greatest reduction in error occurred at an inferior slice with 5 spokes, where a 16.90% SAR underestimation was reduced to 3.38% by networks, indicating an 80% relative reduction in SAR estimation error. Figure 6.5.5b demonstrates that the networks mitigate the motion-induced error considerably for larger motion. The maximum motion-induced error was 53.23% SAR underestimation, which

was reduced to 13.50% by the networks, indicating a 74.64% relative reduction in SAR estimation error.

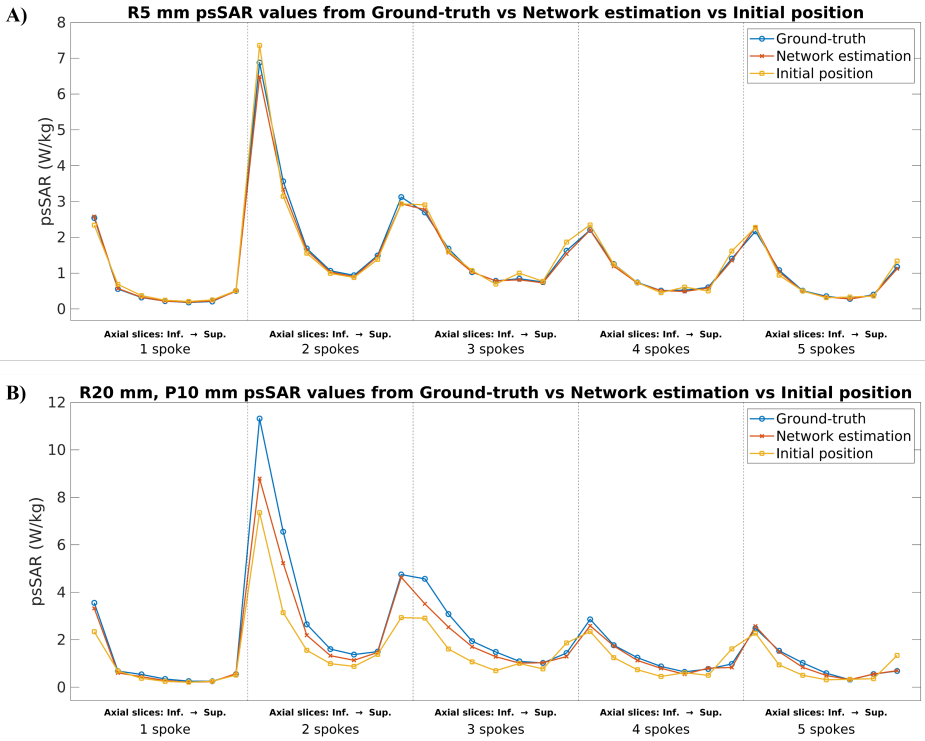


Figure 6.5.5: Comparison of ground-truth, network estimation, and initial position psSAR values for R5 mm and R20 mm, P10 mm motion, displayed for each spoke and slice combination. a) R5 mm motion. b) R20, P10 mm motion. While smaller displacement such as R5 mm showed minimal need for improvement, it is evident that the proposed deep learning method is beneficial for greater displacements such as R20 mm, P10 mm.

Figure 6.5.6 compares the motion-induced and network-estimated error in psSAR for all 35 pulses designed evaluated at all positions. For all pulses, the network estimations represent the actual SAR faithfully, indicated by the narrow range of values that demonstrate reduced underestimation and reduced overestimation.

Figure 6.5.7a, which collates all results into a single comparison, shows that the initial safety model fails to capture the range of SAR values after motion, which are successfully recovered by the network-estimated safety model. Across all cases, psSAR was underestimated by up to 2.14-fold by the initial SAR distribution whereas the worst-case underestimation was only 1.3-fold for the network estimated safety model. In other words, the results suggest that a lower safety factor can be applied when using the proposed method as opposed to the initial method.

Figure 6.5.7b shows scaled calculations that apply worst-case SAR-underestimation values as corrective safety margins to ensure SAR is never underestimated, highlighting how much MRI performance can be gained by the proposed approach. For the initial position-unaware calculations, this scaling emulates a safety model that includes all simulated positions, and limits imaging performance to as low as 21% of the maximum due to SAR overestimation, whereas the proposed position-aware approach ensures more than 68% of the maximum performance can be achieved—indicating a relatively 3.2-fold better imaging performance compared to what is available with the standard position-unaware approach.

Initial safety model fails to capture the variations in after-motion SAR and leads to large underestimations and overestimations in psSAR calculations. Both worst-case SAR underestimation and worst-case overestimation are much lower for network-estimated safety models, which represent after-motion SAR more faithfully. Network estimated Q-matrices also lead to lower imaging performance limitation when compared to using Q-matrices from the initial position.

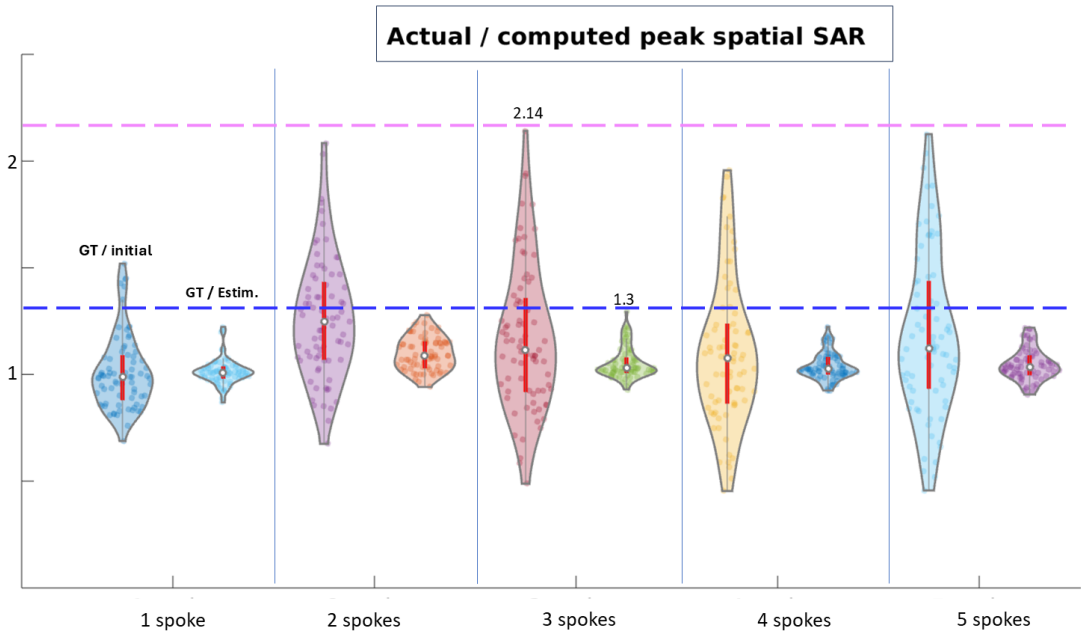


Figure 6.5.6: SAR underestimation due to motion and network-estimations is shown, separately for different pulse types and collated across target slices and evaluated positions. Within each panel, left: ground-truth off-centre psSAR / initial psSAR, and right: ground-truth off-centre psSAR / network-estimated psSAR. Actual psSAR increased by 2.14-fold due to motion (pink dashed line), whereas the networks reduced the estimation error to 1.3-fold (blue dashed line). Red boxplots depict inter-quartile range.

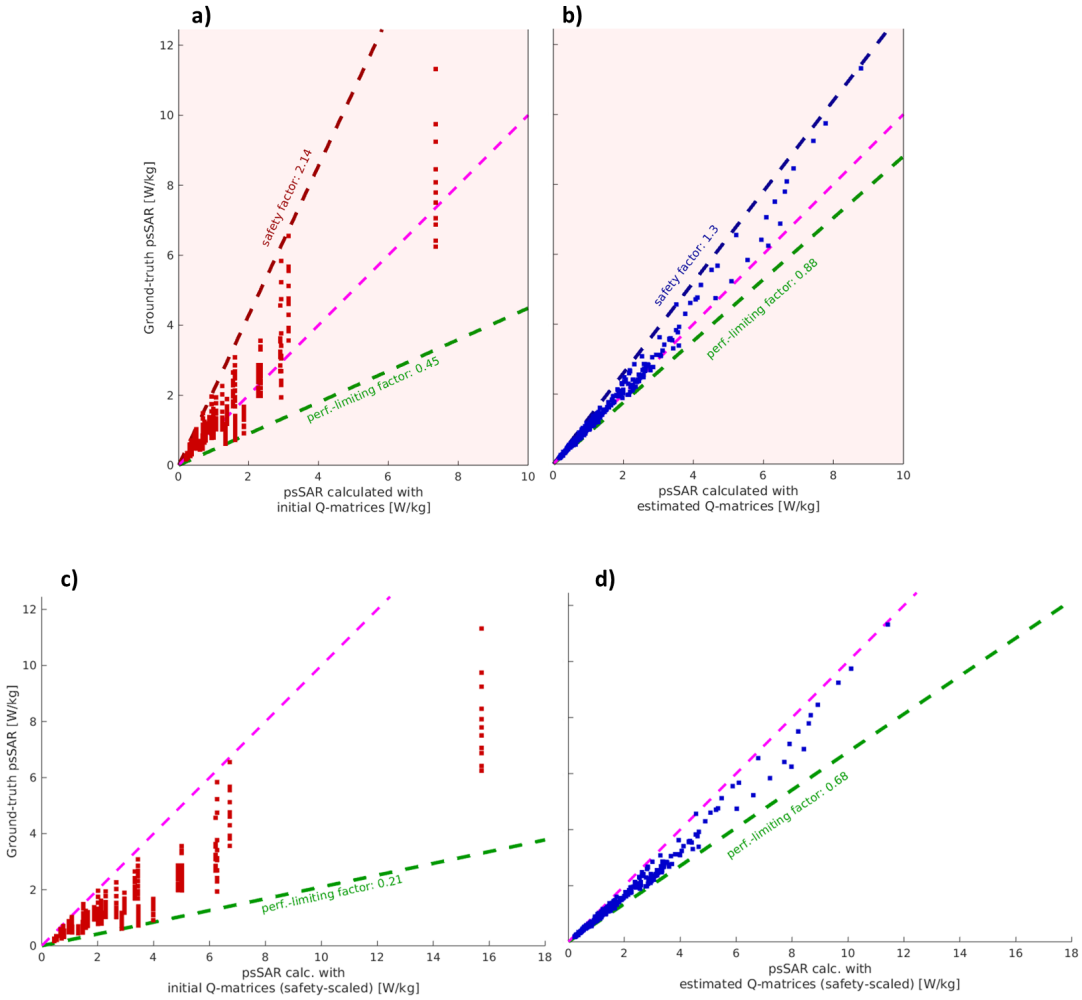


Figure 6.5.7: Initial and network-estimated psSAR values are plotted against ground-truth SAR. (a) initial psSAR vs motion-affected (ground-truth) psSAR. (b) network-estimated psSAR vs motion-affected psSAR. (a-b) show the raw SAR calculations. (c-d) show the scaled calculations when the worst-case underestimation factor is applied as a correction factor to ensure SAR is never underestimated. The corrective factor limits imaging performance to 21% of the maximum when the initial model is used (c), whereas the limitation was 68% when the networks are used. Pink dashed line: identity line, green dashed line: worst-case SAR overestimation, indicating how much imaging performance would be limited due to SAR overestimation.

The Kolmogorov-Smirnov test established that the effect of subject motion on psSAR was significant ($D(385) = 0.10$, $p = 0.035$) while the difference between the ground-truth and network-estimated psSAR values was insignificant ($D(385) = 0.03$, $p = 0.99$), supporting our hypothesis.

## 6.6 Discussion

This study proposes to use DL to predict the effects of subject motion on local SAR distributions. Previous studies in the literature showed that subject motion can have considerable effect on SAR calculations [177]. Our investigations demonstrated that neural networks can estimate these motion-induced changes, yielding local SAR distributions that are in close agreement with actual ground-truth local SAR distributions. This approach can reduce SAR underestimation in the presence of subject motion and thereby decrease the corrective safety margins that are needed to ensure that peak local SAR is not underestimated during imaging. Because corrective safety margins increase SAR overestimation and therefore restrict imaging performance, the proposed approach can lead to higher performance imaging.

We observed that anterior and posterior displacements caused much larger changes in SAR, in alignment with previous literature [177]. This larger increase in anterior-posterior direction is attributed to the head being naturally closer to the coil array in anterior and posterior directions due to head shape. As the head moves, the relative proximity to the coils increases more rapidly—for example, for a coil-tissue distance of 20 mm in anterior and 40 mm in rightward direction; a 10 mm motion would halve the distance if along anterior while only reducing the distance by 25% if along rightward direction. Across positions, we found a monotonous increase in motion-induced SAR error with increasing displacement from the initial position, again similar to previous literature [177], with a single exception that was due to anterior-posterior displacements causing more rapid error increase.

Following a similar approach to previous literature [8], we trained separate networks for different motion types, which can then be cascaded to approximate the actual patient position. This approach provides an efficient training solution, as it allows generating smaller datasets for each network compared to training a single network for all types of motion, which would require a much larger dataset, and training separate networks for each possible subject position, which would not only yield too many networks but also require a larger dataset. In our tests, cascading networks trained for smaller displacements but with more data yielded better estimations of motion-induced changes compared to using networks trained for the exact amount of displacement but with less data.

We investigated various cascading approaches to understand the estimation error floor of the networks and observed that each network adds around 1.5% estimation error. This error was much smaller than the effect of motion on SAR, and therefore, deemed acceptable. To reduce the number of networks that are cascaded, more than one network may be trained for each motion type (e.g., R2 and R5 for rightward) as proposed by Plumley el al. (2022) [8]. This would also improve the accuracy of the approximation of actual subject motion; i.e., having L5, L2, R2 and R5 networks instead of L5 and R5 as implemented in this study would allow cascading to all right-left positions with 1 mm intervals instead of 5 mm. This would enable approximating subject motion with less than 0.5 mm accuracy, which is foreseen to be sufficient. Here we tested an example case where the R5 networks outputs were compared to a position 1 mm further away than the actual displacement. 1 mm did not significantly impact network accuracy. This expands the utility of the proposed approach, as very precise estimation of organic motion is not crucial.

With more than one way to cascade multiple networks to the same end result, we investigated the effect of the order in which networks are cascaded. The effect on estimation accuracy we observed in our investigations was small, indicating the robustness of the cascading approach when estimating changes in SAR.

Previous studies showed that subject motion can have a major impact on SAR estimations [177]—a behavior shown to be consistent across different body models [143]. Here, we observed similar motion-induced SAR increases, while variations in numerical values can be attributed to different body models used. Even in the absence of motion, Kopanoglu [48] showed that initial patient positioning affects SAR considerably, and although safety models that comprise body models at all possible positions prevent underestimation, they are substantially over-conservative, and may therefore hamper imaging performance. Akin to Plumley et al.'s previous work [8] that used $B_1^+$ maps at the initial position to estimate $B_1^+$ maps at other positions, this work can transform Q-matrices at the initial position to Q-matrices at other subject positions. This reduces the need for over-conservative position-wise all-inclusive safety models without risking underestimation of SAR. The Q-matrices at the initial position needed for this method to work in practice can be provided using the recent method by Brink et al. [86] with both methods used in conjunction providing researchers and clinicians with accurate local SAR values during scan time, maintaining high-performance and safe scanning.

The method proposed here introduces networks that can facilitate position-adaptive safety models by taking position tracking information from any method that provides these measurements in near real time and using them to update the safety models. This will in turn enable updating the safety calculations for applied pulses. It can be combined with any real time pulse design method [48] [47], which would also benefit from updated B1 maps to adapt the excitation to motion as it happens [8]. MR scanners use fixed safety models, but this method will facilitate position adaptive safety models. In practice, safety models include many positions, whereas a method like the proposed one can adapt position-aware safety models to patient motion as it happens with recent methods in literature focusing on patient-specific safety models [39] [86], which are inherently position-specific as well. In terms of clinical applicability, we intend our method to take those patient-specific safety models and motion tracking information as input and provide position-adaptive safety models. The envisioned method is sequence-agnostic and agnostic to motion-tracking methods, therefore pertinent tools are not specified here. For the implementation here, total computation time was not a constraint. Therefore, a sequential slice-by-slice estimation approach was adopted, with the estimation for a single slice taking around 16 ms. A further investigation might adopt a three-dimensional estimation approach to estimate motion-related changes across slices together instead of sequentially, to accelerate computation.

The initial in vivo implementation of this pipeline will likely be accompanied by regulatory hurdles in terms of ethics. Ways to validate the final method may include applying a phantom with motion capabilities. These phantoms can be accompanied by multiple local temperature probes. The min issue for safety, and therefore the main interest of regulatory bodies, lies in network underestimation of local SAR. Therefore initial in vivo implementations can be accompanied by conservative upper bounds. There should also be a guard in place to revert to the original overconservative worst case safety margin protocol in case the software solution proposed here fails. Over time, the pipeline can undergo a gradual reduction of safety margin under review board approval. This process will inevutable also require a comparative study, tested against the standard VOP approach.

Q-matrix and SAR distributions have rather smooth spatial variations, which means adjacent slices contain overlapping information. This can positively bias network estimation performance. Therefore, despite there being studies in the

literature that interleaved slices for training, validation and testing datasets [184], we adopted the approach of splitting the datasets at a body model level. This ensures that the test dataset is completely unseen during training. In our limited investigations, interleaving slices indeed yielded better performance than splitting datasets at the body model level, although the differences in this paper were not major. Splitting datasets at the body model is also more realistic, as the neural networks would be expected to perform well on unseen subjects in practice, especially in clinical settings.

Despite the benefit of splitting datasets at the body model level in terms of testing reliability in unseen conditions, it also means that particularities of the anatomy used for testing might not be represented in the training dataset at all. In this study, we have tested the networks in a 4-fold cross validation investigation, testing on female pre-teenager, female adult, male obese adult, and male elderly adult models. For smaller displacements, the networks yielded similar performance for all models, while the improvement provided by the networks was lower for the highest BMI model than the other cases for larger displacements. This is likely due to the large mismatch in BMI values across the training and validation (BMI values between 15.3 and 22.4), and testing (BMI: 36) datasets. For practical implementation, networks trained on more diverse datasets taking into consideration the effect of factors like anatomical variations (head shape and size), BMI, change in tissue characteristics with age, and presence of pathology or inflammation, may yield more reliable and more generalizable performance. Another approach could be to train tailored networks for specific populations (e.g., low-BMI vs high-BMI, paediatric vs adult, healthy vs certain pathology), which could make safety models more tailored to the population and reduce over-estimation margins.

## 6.7 Conclusion

In summary, this proof-of-principle study proposes a way to adapt safety calculations to patient motion as it happens, by estimating the effect of motion on SAR using deep learning. This can reduce the safety margins required to ensure adherence to safety limits. Despite its limitations, the proposed approach demonstrates a feasible alternative to using high-performance but risky safety models that might underestimate SAR and position-wise all-inclusive safety models that hamper imaging performance.

## 6.8 Manuscript Appendix

The mapping scheme used in this paper uses single-coil-only and two-coil-only coil combinations. In the latter, case only two coils are turned on while the others are turned off. Two such combinations exist for each coil pair evaluated, where the coefficients are $(c_m, c_n)$ and $(c_m, \widetilde{c_n})$. The general reverse mapping equation to calculate the Q-matrices is given by:

$$
{}_{\text{estim}}Q_{m,n} \quad = \quad
\begin{cases}
\dfrac{{}_{\text{estim}}\text{ilSAR}_m^{c_m}}{|c_m|^2}, \\
\\
m = n \\
\\
\dfrac{1}{c_m^*(c_n\widetilde{c}_n^* - \widetilde{c}_n c_n^*)}\left[\widetilde{c}_n^{\,*}\,{}_{\text{estim}}\text{ilSAR}_{m,n}^{c_m,c_n} - c_n^*\,{}_{\text{estim}}\text{ilSAR}_{m,n}^{c_m,\widetilde{c}_n} + (c_n^* - \widetilde{c}_n^*)\,{}_{\text{estim}}\text{ilSAR}_m^{c_m} + c_n^*\,{}_{\text{estim}}\text{ilSAR}_n^{\widetilde{c}_n} - \widetilde{c}_n^*\text{ilSAR}_n^{c_n}\right], \\
\\
m < n \\
\\
\dfrac{1}{c_m(c_n^*\widetilde{c}_n - \widetilde{c}_n^* c_n)}\left[\widetilde{c}_n\,{}_{\text{estim}}\text{ilSAR}_{m,n}^{c_m,c_n} - c_n\,{}_{\text{estim}}\text{ilSAR}_{m,n}^{c_m,\widetilde{c}_n} + (c_n - \widetilde{c}_n)\,{}_{\text{estim}}\text{ilSAR}_m^{c_m} + c_n\,{}_{\text{estim}}\text{ilSAR}_n^{\widetilde{c}_n} - \widetilde{c}_n\text{ilSAR}_n^{c_n}\right], \\
\\
m > n
\end{cases}
\tag{6.8.1}
$$

where $m$ and $n$ are channel indices, $(c_m, c_n)$ and $(c_m, \widetilde{c_n})$ denote the two pairs of coefficients used for channel combination $(m, n)$, and the superscript * denotes complex conjugate. For the channel coefficients used in this study, $c_m$=$\sqrt{2}$ for single channel ilSAR and $(c_m, c_n)$=(1,1) and $(c_m, \widetilde{c_n})$=(1,$i$) for two-channel interactions, yielding ${}_{\text{estim}}\text{ilSAR}_n^{c_n} =_{\text{estim}} \text{ilSAR}_n^{\widetilde{c_n}}$, and Equation [A1] reduces to Equation 6.4.6.

# Chapter 7

# Applying the Successful Network Architecture to the Original Tested Data Types

## 7.1 Introduction

Following the success of the U-net on the ilSAR distributions in the previous chapter, we resolved to test the same preprocessing methods and network on the E-fields and Q-matrices presented in Chapter 4. The rationale for experimenting with these datatypes remains the same. The U-Net had the potential to be a reasonable tool for the task because of its simpler and more stable performance, compared to a cGAN. The research objectives for this chapter were to discover whether a simplified network architecture (the U-Net) can be applied to data which overtly appears more nuanced than ilSAR, such as E-field magnitudes and phase data from both E-fields and Q-matrices. Though it is likely that more elaborate physical representations are unsuitable for this architecture, it is still worth investigating whether a cGAN or U-Net is a more effective predictive model. An improved preprocessing pipeline like the one described in Chapter 6 is expected to further optimize the results, since the network will no longer be training on empty data (air above the head of the model within the coil) and be susceptible to overfitting as a result of having repeated anatomies from the three models used in the training dataset. Reconfiguring the body models (ie. testing on the Billie model rather than the Ella model) is additionally likely to improve

the results due to Billie having the lowest body mass index out of all of the models and being overall smaller, meaning that the anatomy fits within the bounds of all of the training models. The objective here is to discover whether both an optimized dataset preparation method and simplified network architecture yields better network prediction results in terms of the established error metrics and qualitative data visualizations. A detailed rationale for the alternative network architecture is discussed in Chapter 8.

## 7.2 Preprocessing methods used for both E-fields and Q-matrices

Overall, the pipeline used to predict the effects of motion on $E_x$-fields and Q-matrices using U-Nets was nearly identical to the pipeline described in Chapter 6, where U-Nets were used to predict the effects of motion on ilSAR distributions.

The body model configuration for both E-field and Q-matrix dataypes was Fats, Ella and Glenn for training, Duke for validation, and Billie for testing. All slices which contained no tissue data were removed. The training datasets only contained every third slice from each training body model due to the U-Net requiring less data and to prevent overfitting (reasoning identical to Chapter 6). The magnitudes of both the E-fields and Q-matrices were normalized to a range between 0 and one, identical to the ilSAR in Chapter 6. Phase distributions were jittered (off-set) and normalized in an identical way to Chapter 4.

## 7.3 E-Fields

The posterior 5 mm (P5 mm) magnitude and phase networks were identical to the one used to predict ilSAR distributions, except that the number of channels was set to eight. The results were processed in an identical way and the error is reported as normalized root mean square error (nRMSE) for magnitude, and $^\circ$ error for phase.

### 7.3.1 Methods

#### Magnitude

The magnitude data was preprocessed with both channelwise and global normalization. Additionally, the results of the magnitude $Ex$-field network estimations were postprocessed with and without a 5x5 pixel Gaussian filter. In total, four magnitude $E_x$-field results are reported here.

#### Phase

The phase network-estimation errors were calculated using an identical method described in Chapter 4.

### 7.3.2 Results

The U-Nets trained for approximately three hours each, reducing the amount of training time by 62.5% compared to the cGAN training time.

#### Magnitude

Processed with channel-wise normalization and without the Gaussian filter, the network estimation nRMSE was 4.81%, while motion error was 10.47%, yielding a 54.05% nRMSE decrease. Compared to the 42.06% error increase from cGAN estimation error for this specific pre and postprocessing combination, the error while using the U-Net reduced considerably. With the Gaussian filter applied to the U-Net estimated image, the nRMSE rose to 7.66%. The global normalization without the Gaussian filter yielded a network estimation nRMSE of 5.45% compared to 10.20% for default-motion error. In this case, the Gaussian filter increased the network estimated nRMSE to 12.93%. Overall, and with the exception of combining a global normalization factor with a Gaussian filter, the U-Net consistently achieved greater reductions in nRMSE than the cGAN. In other words, relative to the baseline motion error, the U-Net produced more substantial prediction improvements than the cGAN.

The $E_x$-field distributions are visualized in Figures 7.3.1, 7.3.2, 7.3.3, 7.3.4,7.3.5, and 7.3.6. Like in Chapter 4 of this thesis, the error images are not nRMSE, but rather reflect cases of maximum point error. This is the case for all of the following sections in this chapter. It is evident that in all of the aforementioned figures except Figure 7.3.5, the network-estimation error is lower. The figures show

the channels which have the highest default-motion error or network-estimation error. In some cases, one channel pertains to both default-motion and network-estimation error. While the maximum intensity projections (Figures 7.3.1, 7.3.2, 7.3.3, 7.3.4,7.3.5, and 7.3.6) display localized points of maximum error within the worst performing channel, they do not reflect the overall channel-wise estimation accuracy.

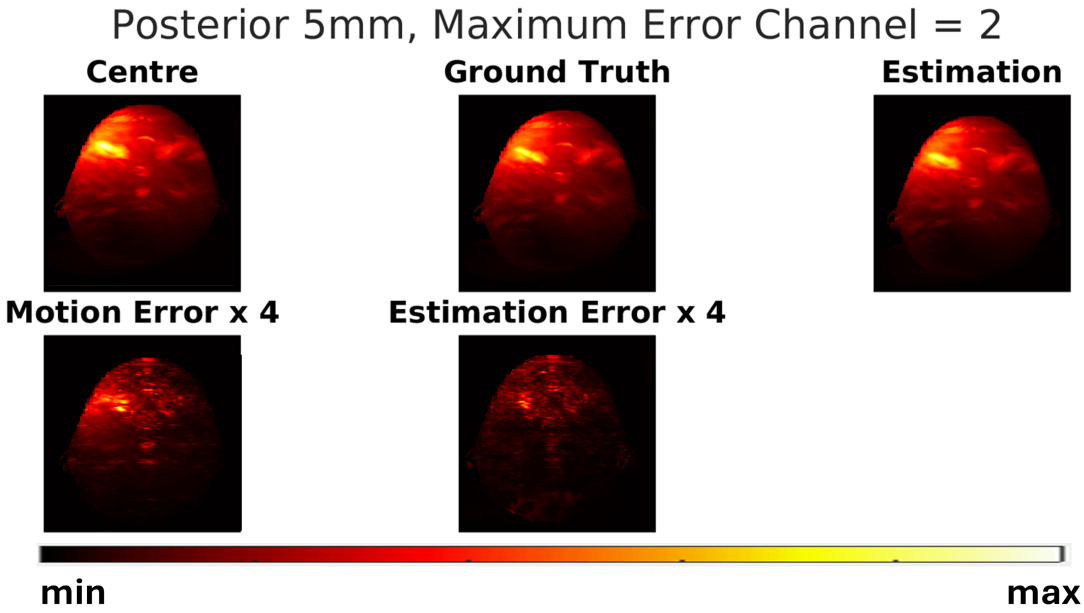


Figure 7.3.1: E-field magnitudes exhibited as maximum intensity projections along the z-axis (all 136 tissue-containing slices) for channel 2 which yielded the worst error in terms of both default motion and network estimation. This data is **normalized channel-wise** and the resulting slices have **not been smoothed** with a 5x5 Gaussian filter. The error images in the bottom row have been magnified 4-fold for clarity.

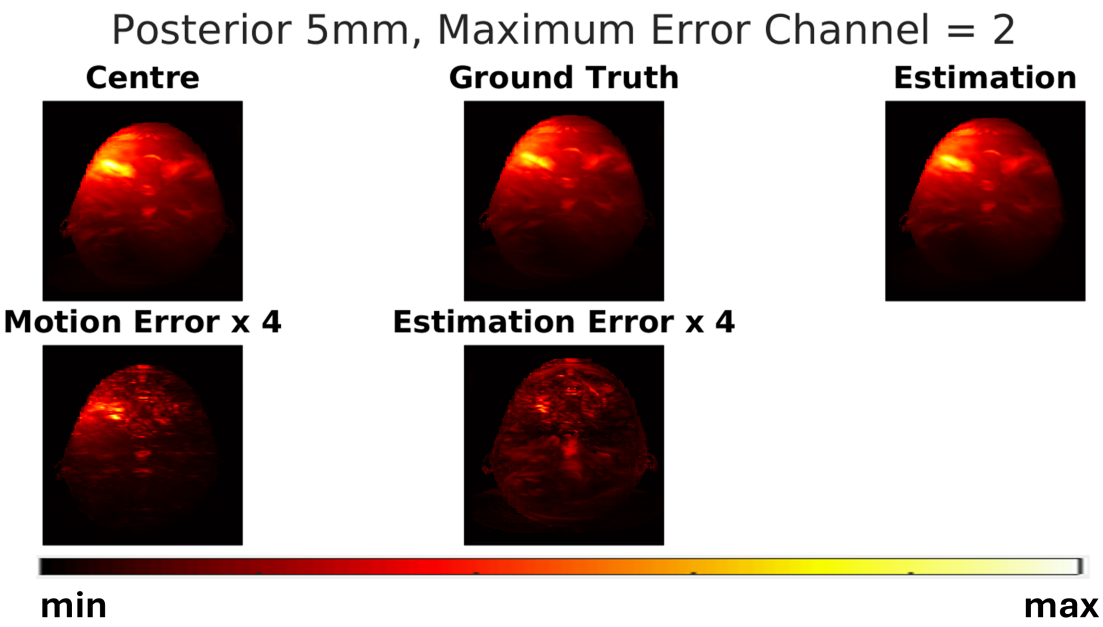


Figure 7.3.2: E-field magnitudes exhibited as a maximum intensity projections along the z-axis (all 136 tissue-containing slices) for channel 2 which yielded the worst error in terms of both default motion and network estimation. This data is **normalized channel-wise** and the resulting slices **have been smoothed** with a 5x5 Gaussian filter. The error images in the bottom row have been magnified 4-fold for clarity.

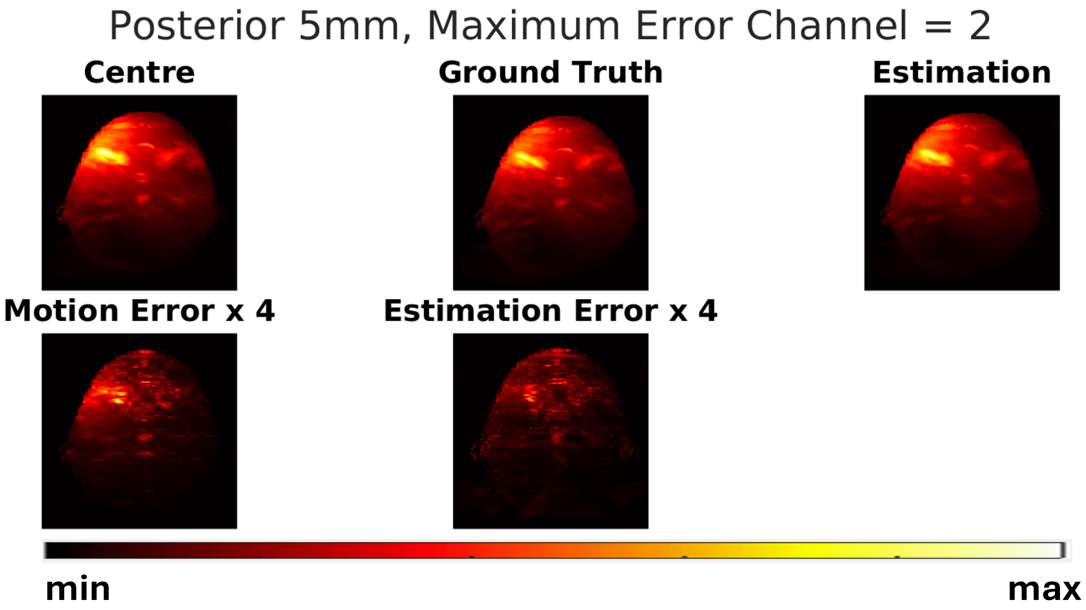


Figure 7.3.3: E-field magnitudes exhibited as a maximum intensity projections along the z-axis (all 136 tissue-containing slices) for channel 2 which yielded the worst default-motion error. The data was normalized with a **global normalization factor**, and the resulting slices **have not been smoothed** with a 5x5 Gaussian filter. The error maps are magnified 4-fold for clarity.

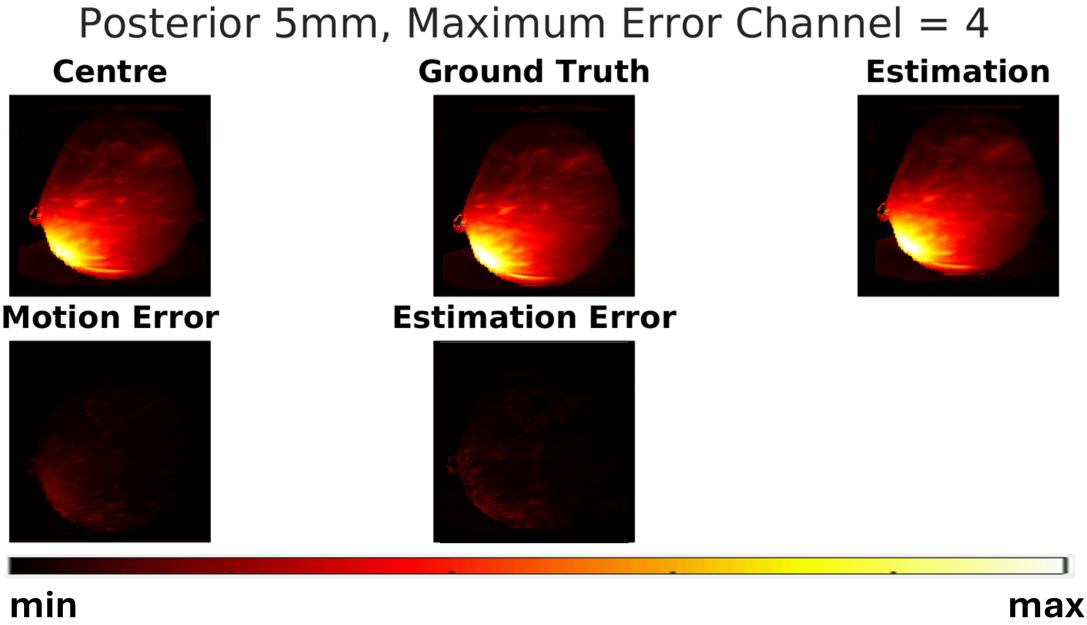


Figure 7.3.4: E-field magnitudes exhibited as a maximum intensity projections along the z-axis (all 136 tissue-containing slices) for channel 4 which yielded the worst network-estimation error. The data was preprocessed with a **global normalization factor** and the resulting slices **have not been smoothed** with a 5x5 Gaussian filter.

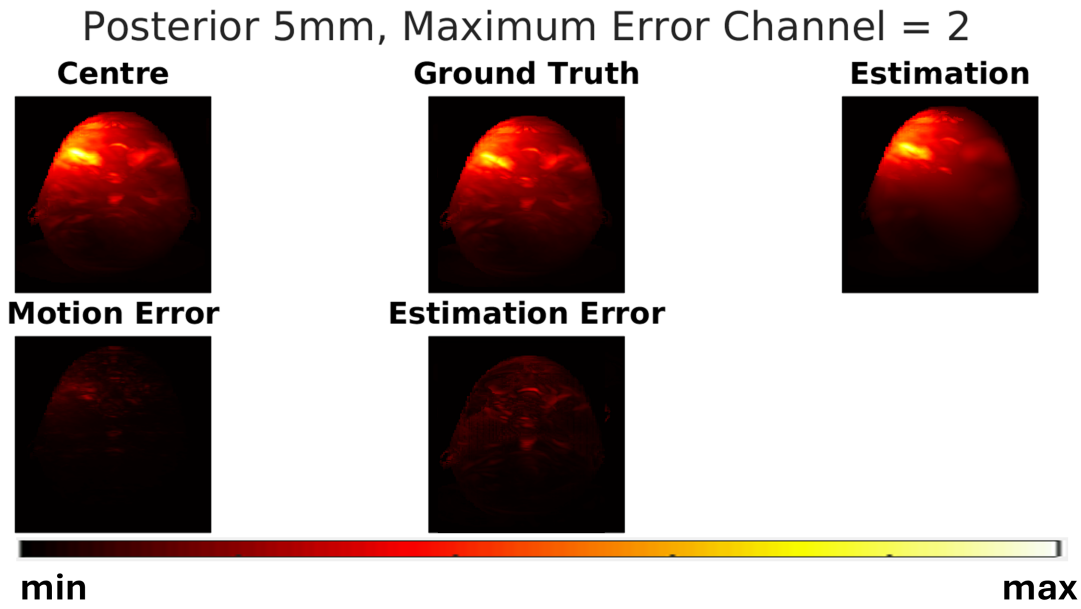


Figure 7.3.5: E-field magnitudes exhibited as a maximum intensity projections along the z-axis (all 136 tissue-containing slices) for channel 2 which yielded the worst default-motion error. This data was normalized with a **global normalization factor,** and the resulting slices **have been smoothed** with a 5x5 Gaussian filter.

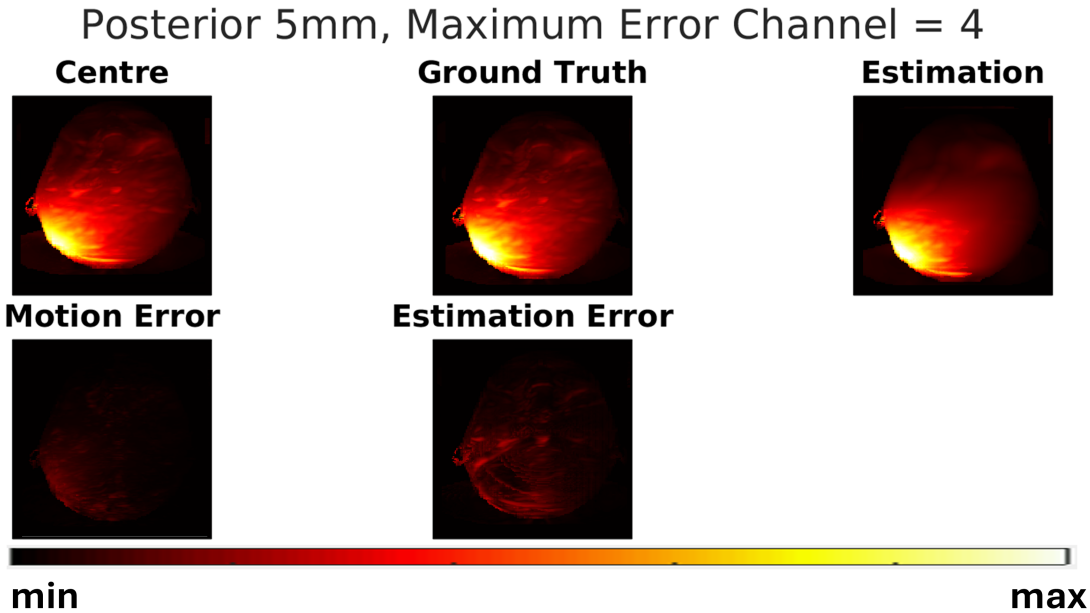


Figure 7.3.6: E-field magnitudes exhibited as a maximum intensity projections along the z-axis (all 136 tissue-containing slices) for channel 4 which yielded the worst network-estimation error. This data has been normalized with a **global normalization factor** and the resulting slices **have been smoothed** with a 5x5 Gaussian filter.

The MIPs do not reflect the nRMSE for the specific channel indicated. For further clarity, Figures 7.3.7, 7.3.8, 7.3.9, and 7.3.10 display the nRMSE per channel for each of the four investigated $E_x$-field magnitude cases. In all cases, the default motion nRMSE was highest at channel 1. For network-estimation error, channel 5 yielded the highest nRMSE in three out of the four configurations. The exception was the global normalization without Gaussian filtering case, where channel 1 also showed the highest network-estimation error.

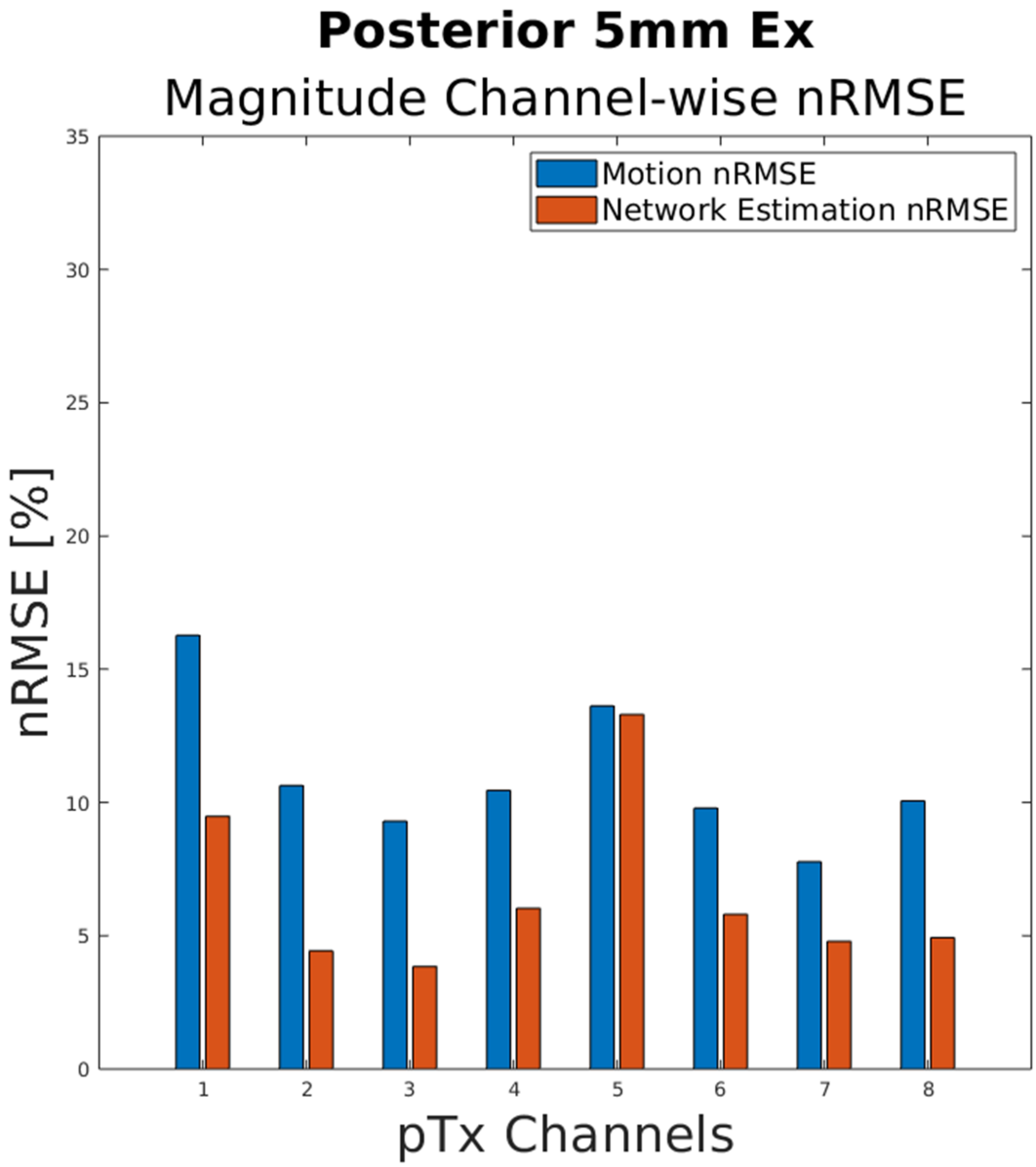


Figure 7.3.7: Bar chart of nRMSE per channel. This data has been **normalized channel-wise.** The network estimations have **not been smoothed** with a Gaussian filter. The blue bars are default-motion error, while the orange bars are network-estimation error.

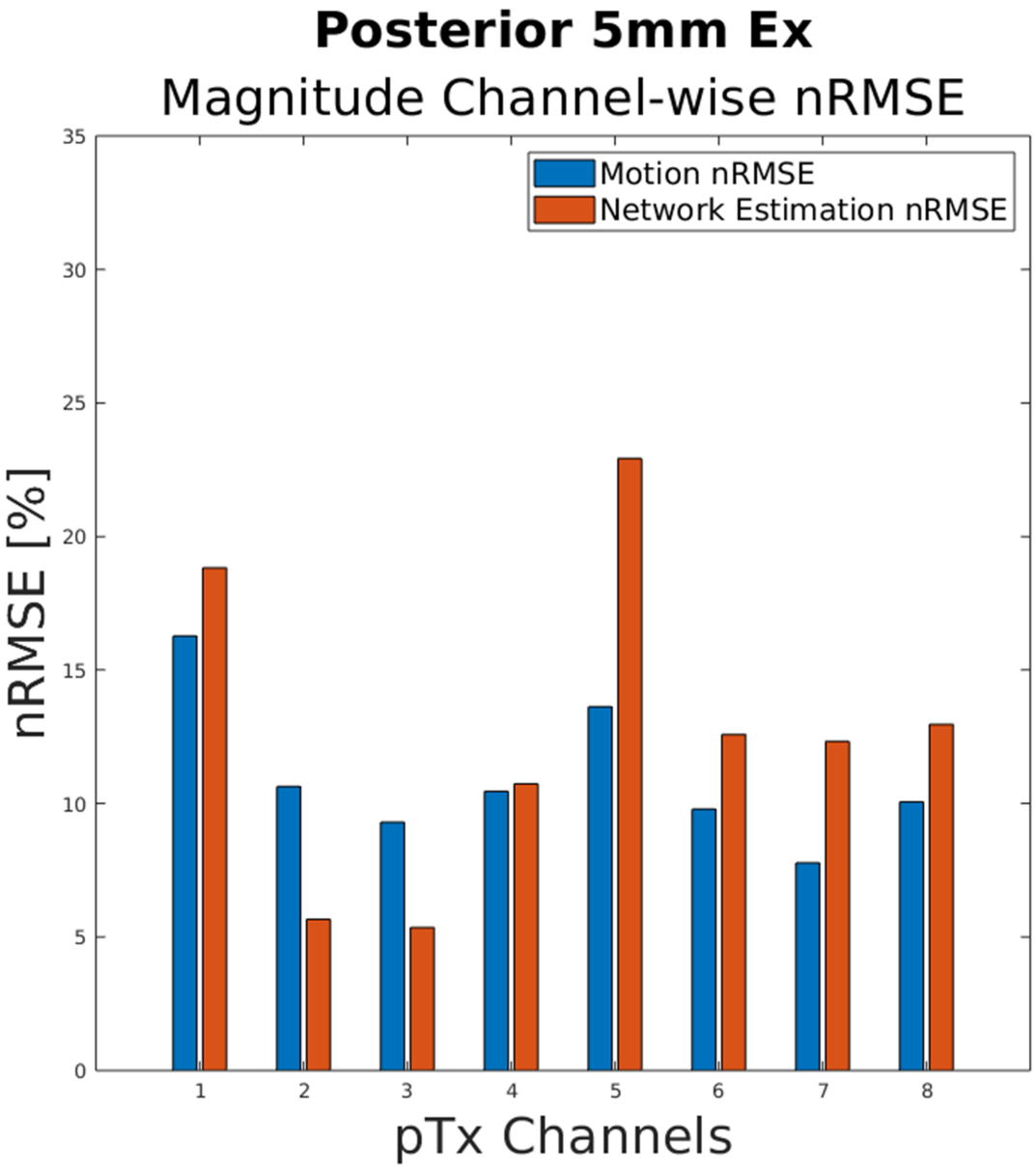


Figure 7.3.8: Bar chart of nRMSE per channel. This data has been **normalized channel-wise**. The network estimations have **been smoothed** with a Gaussian filter. The blue bars are default-motion error, while the orange bars are network-estimation error.

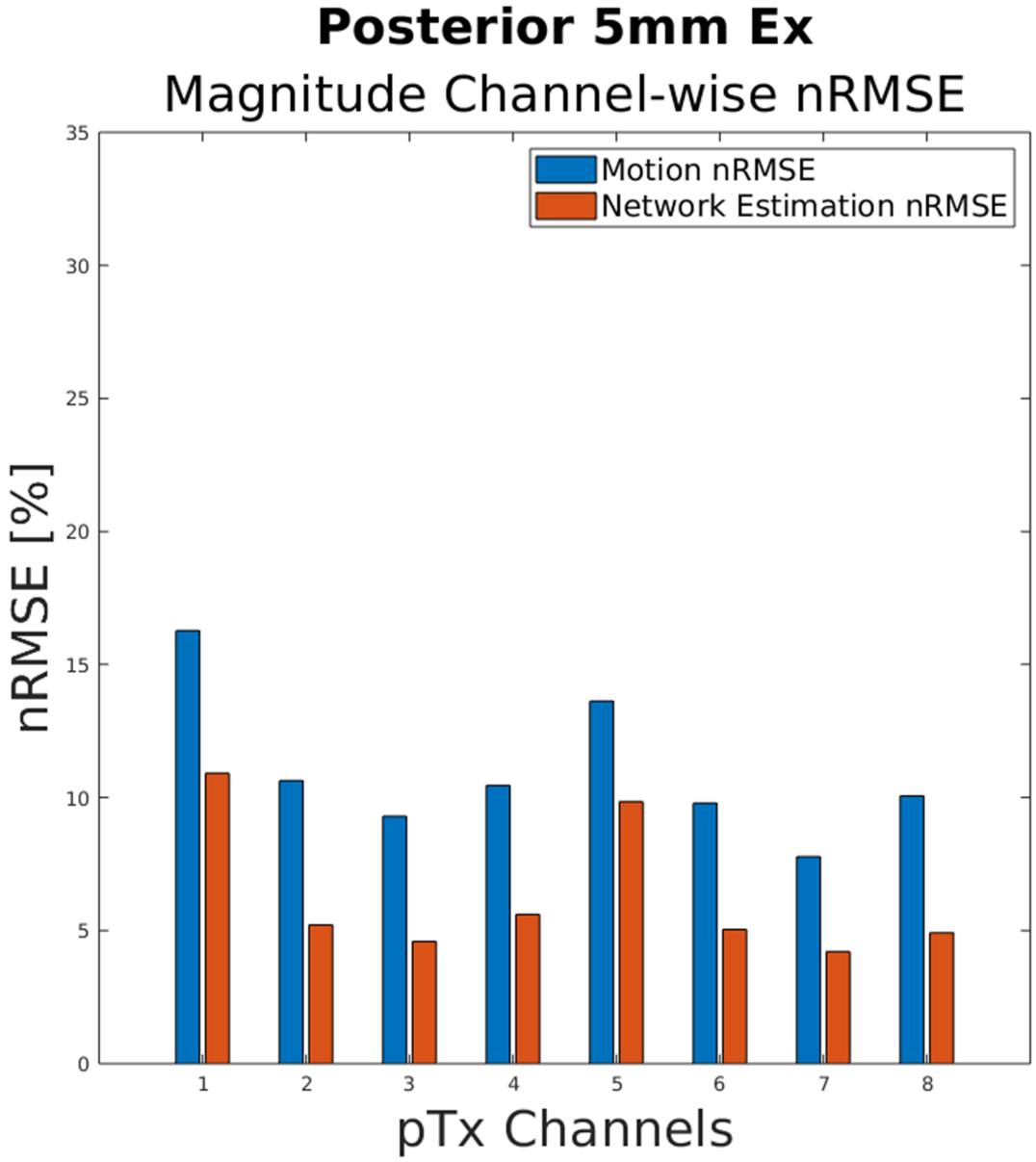


Figure 7.3.9: Bar chart of nRMSE per channel. This data has been normalized with a **global normalization factor**. The network estimations have **not been smoothed** with a Gaussian filter. The blue bars are default-motion error, while the orange bars are network-estimation error.

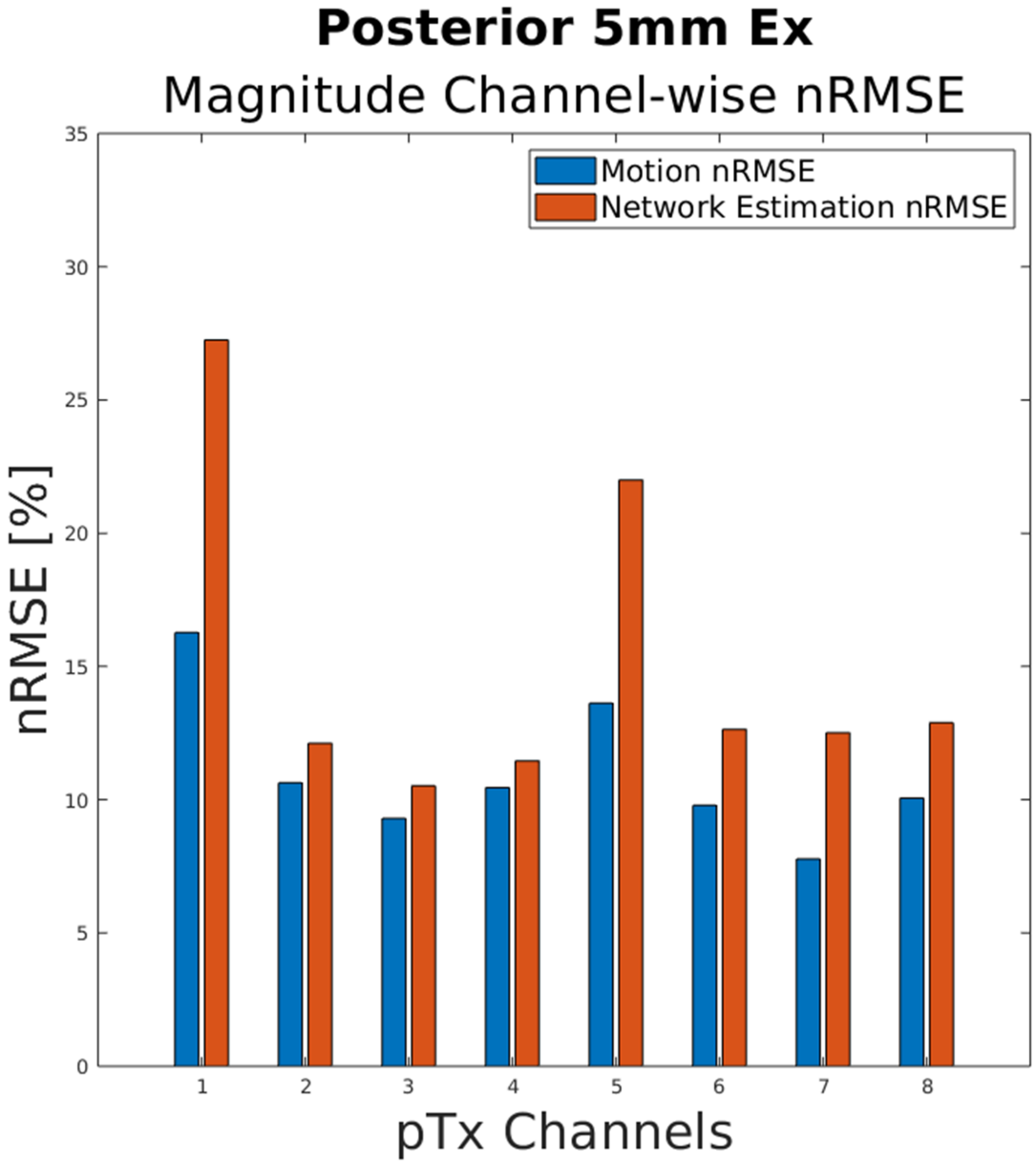


Figure 7.3.10: Bar chart of nRMSE per channel. This data has been normalized with a **global normalization factor**. The network estimations **have been smoothed** with a Gaussian filter. The blue bars are default-motion error, while the orange bars are network-estimation error.

### Phase

The network-estimation error was 1.85°, while the original motion error was 1.25°. This is a 48% error increase compared to the cGAN-predicted $E_x$-field phase results which yielded a 44% error increase, indicating that the U-Net yields worse performance for this datatype scenario than the cGAN. The results are visualized

in Figures 7.3.11 and 7.3.12.

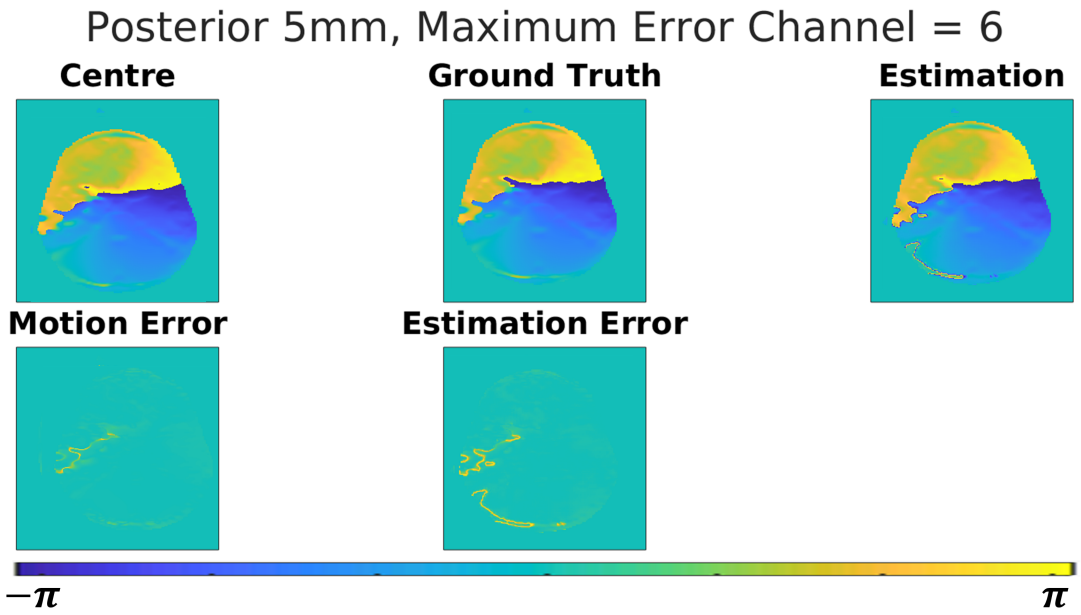


Figure 7.3.11: Error of E-field phases exhibited at a central slice for channel 1 which yielded the worst default-motion error. The scale is between -pi and pi.

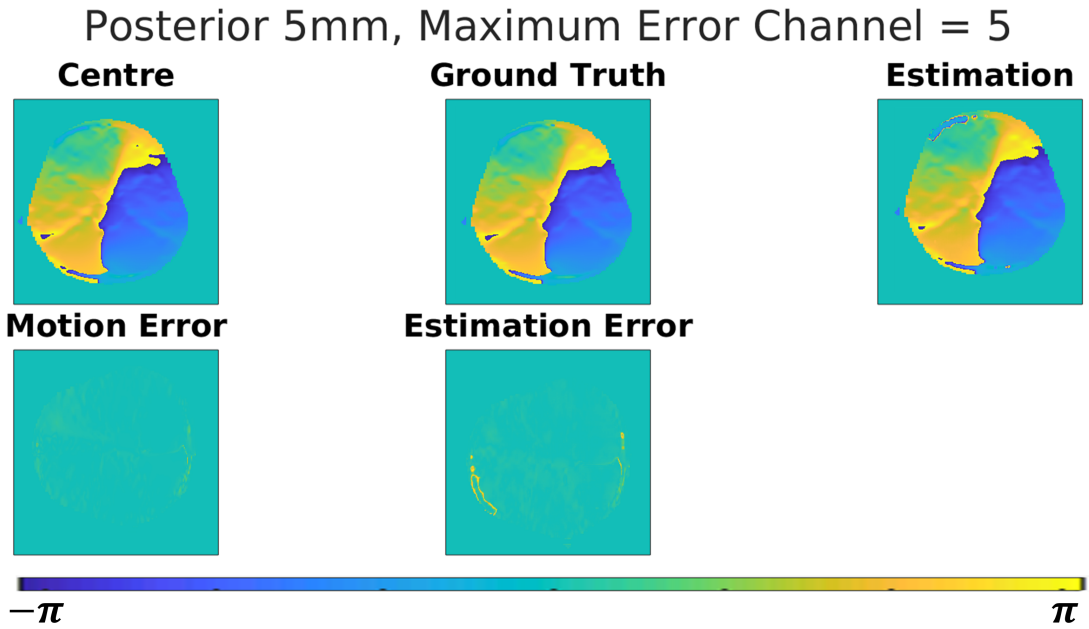


Figure 7.3.12: Error of E-field phases exhibited at a central slice for channel 1 which yielded the worst network-estimation error. The scale is between -pi and pi.

Figure 7.3.13 shows error (unit: °) per channel. None of the network-estimation

errors are lower than the default-motion errors.

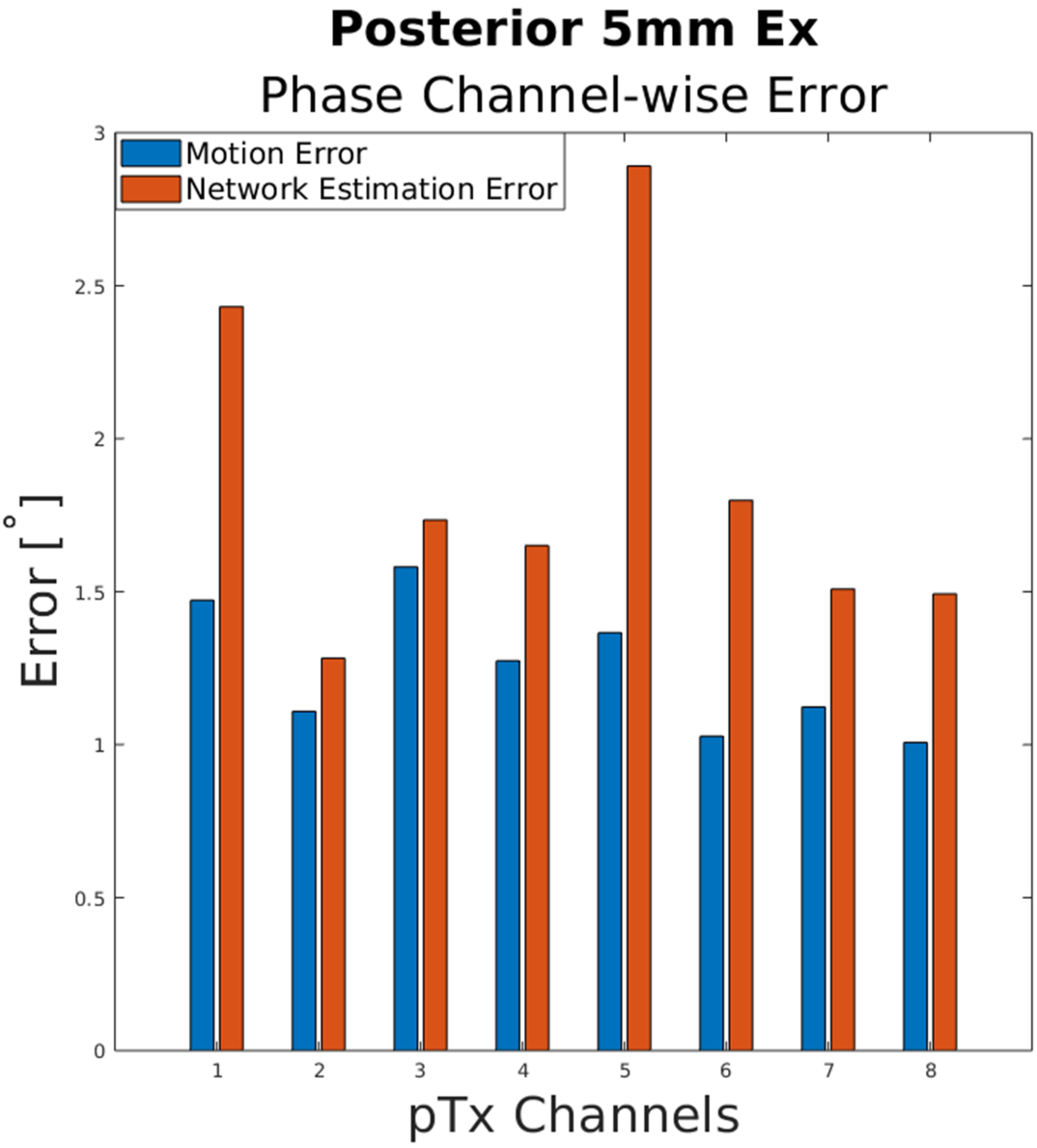


Figure 7.3.13: Bar chart of Motion and Network Estimated ° error per pTx channel for the phases of $E_x$-fields.

### Complex $E_x$-field

The nRMSE was calculated for the best-performing magnitude $E_x$-field data. Motion-affected nRMSE for complex $E_x$-fields where the magnitude part was normalized channel-wise and did not undergo Gaussian smoothing was 11.3%. The network estimation nRMSE for this case was 12.2%. The channel-wise nRMSE values are presented in Figure 7.3.14

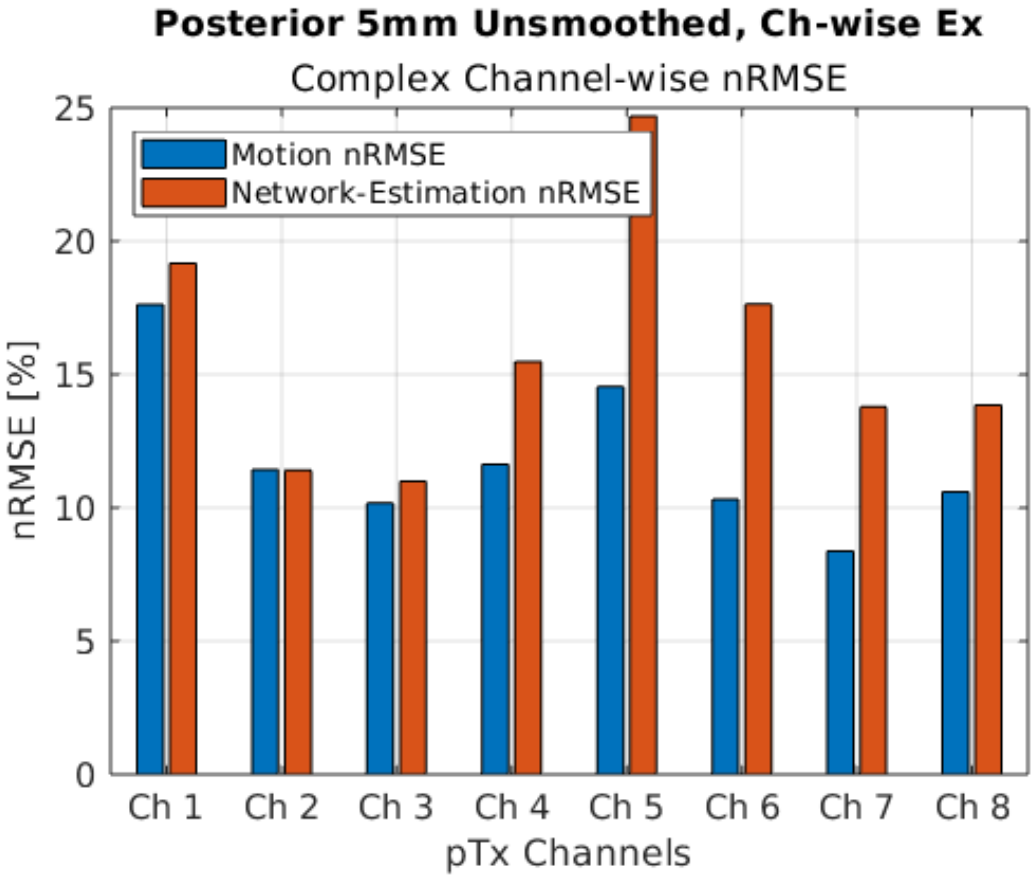


Figure 7.3.14: Bar chart of Motion and Network Estimated nRMSE per pTx channel for complex $E_x$-fields composed of the best-performing magnitude estimations (not Gaussian-smoothed and normalized channel-wise).

### 7.3.3 Section Discussion

The U-Net architecture yielded promising results for $Ex$-field magnitude estimation, with the most accurate predictions obtained under channel-wise normalization without Gaussian filtering. It could be that, under appropriate pre and post-processing conditions, the U-Net is capable of effectively learning the mapping between initial position and motion-affected $E_x$-field magnitude distributions.

The normalization strategy and postprocessing both had a pronounced effect on network performance. Channel-wise normalization consistently outperformed global normalization across all cases. This likely stems from the fact that channel-wise normalization better preserves inter-channel intensity structure, which prompts the network to learn detailed, localized patterns within each $E_x$-field component. In contrast, global normalization may suppress these relative differences, impairing the network's ability to distinguish spatial features that are useful for accurate

estimation.

Postprocessing the network predictions with a Gaussian filter consistently increased the estimation error. While smoothing may help mitigate noise, in this context, it appears to degrade structural detail. The smoothing operation likely removed localized intensity variations that the U-Net was otherwise able to preserve, thereby reducing the fidelity of the predicted field distributions.

This effect is most clearly observed in Figure 7.3.5. In all of the aforementioned figures except Figure 7.3.5, the network-estimation error is lower than the default-motion error. Figure 7.3.5 represents the case where both global normalization and Gaussian filtering were applied. Both are conditions which, when combined, appear to degrade estimation quality. Global normalization may hinder the network's ability to resolve spatial variations by suppressing inter-channel intensity differences, thereby reducing contrast essential for learning channel-specific features. The application of a Gaussian filter further smooths spatial details, which may remove or blur localized variations in the $E_x$-field that the network otherwise captures. These compounding effects appear detrimental in channels 1 and 5, which already exhibit high motion error, thereby resulting in network predictions that are less accurate than the uncorrected, motion-affected baseline.

The results for $E_x$-field phase estimation, however, were less favorable. The results suggest that that U-Nets may be even less suited for phase prediction in this context. On the other hand, the errors are not astronomical and visual error results show that the estimated phases largely resemble both the ground-truth and initial position distributions. This means that 5 mm head motion minimally affects phase data and that the networks are indeed capable of generating nuanced phase images. In light of the above, it is reasonable to state that the results presented here are inconclusive and must be tested in future work.

Taken together, these findings suggest that U-Nets can perform well for magnitude-based $E_x$-field estimation when appropriate normalization is used and spatial filtering is avoided. However, their performance is inconclusive when applied to phase estimation or when preprocessing choices disrupt the local structure of the data. When re-uniting the best-performing magnitude results with the phase fields, the results remain unfavorable.

The performance of the U-Net on $E_x$-field data demonstrates that while the network can perform well under specific preprocessing settings, its accuracy is

sensitive to both input representation and postprocessing choices. These observations are pertinent when applying the same architecture to Q-matrix data, which introduces further intricacies due to the channel interactions. The following section examines how the U-Net manages these conditions compared to the cGAN and evaluates its effectiveness across different Q-matrix representations, including magnitude, phase, and the separate real and imaginary parts.

## 7.4 Q-matrices

### 7.4.1 Methods - Magnitude and Phase

The P5 mm U-Nets were nearly identical to the ones used in Chapter 6, except that they were designed to accommodate 36 channels.

Phase networks were nearly identical to the magnitude ones with the exception of filter sizes, yielding two different networks to be trained on the Q-matrix phases. One U-Net trained with 3x3 filters like in Chapter 6, and another trained with 8x8 filters like in Chapter 4 and Ref. [8].

Because of previous observations in Chapter 4 that 5 mm head motion has a minimal effect on the presentation of phase distributions, an additional 10 mm post hoc follow-up study is included in this section. The 10 mm data was generated by means of the cascading method described in Chapter 6, where the 5 mm evaluation networks were combined twice to achieve 10 mm. The phase network expanded to 10 mm was the one using the best-performing hyperparameters observed in the 5 mm investigations.

### 7.4.2 Results - Magnitude and Phase

#### Magnitude

Each U-Net ran for approximately four hours. This is a 71.43% reduction in training time compared to the 14 hours required to train the cGANs.

The U-Net successfully estimated the effects of motion on the magnitudes of the Q-matrices. While network estimation nRMSE was 5.86%, motion error was 19.51%. This is a 70% error reduction compared to the 62% error reduction achieved when using cGANs. The magnitudes of the Q-matrices are visualized in Figures 7.4.1 and 7.4.2. In both figures, it is evident that the network estimations

resemble the ground-truth motion-affected images more than the initial position images do.

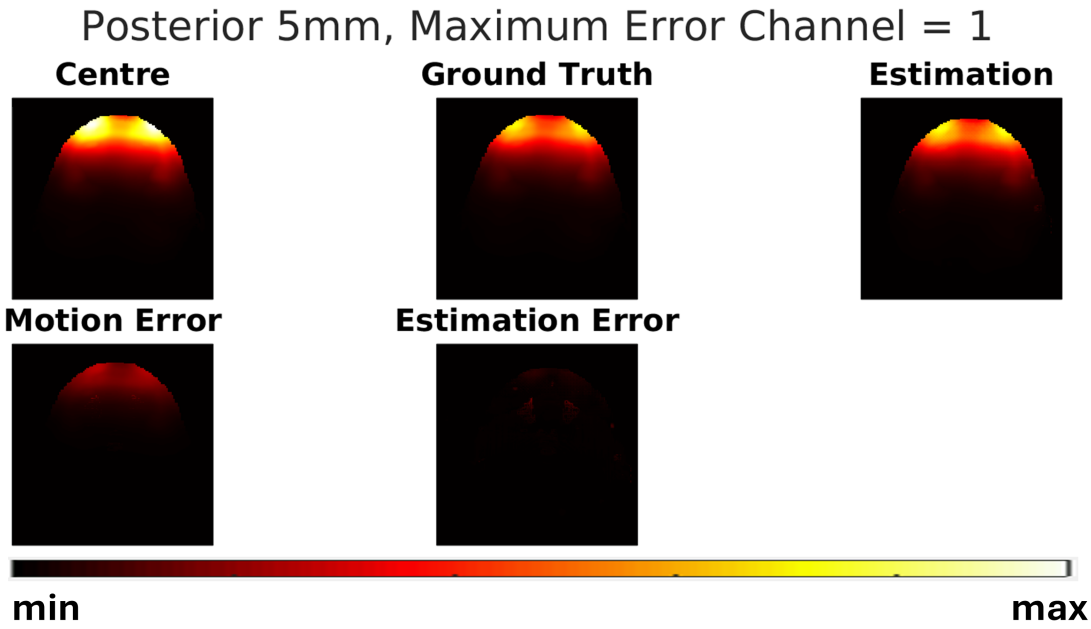


Figure 7.4.1: Q-matrix magnitude maximum intensity projections along z. This figure shows a central slice for channel 1 ($Q_{1,1}$) which yielded the worst default-motion error.

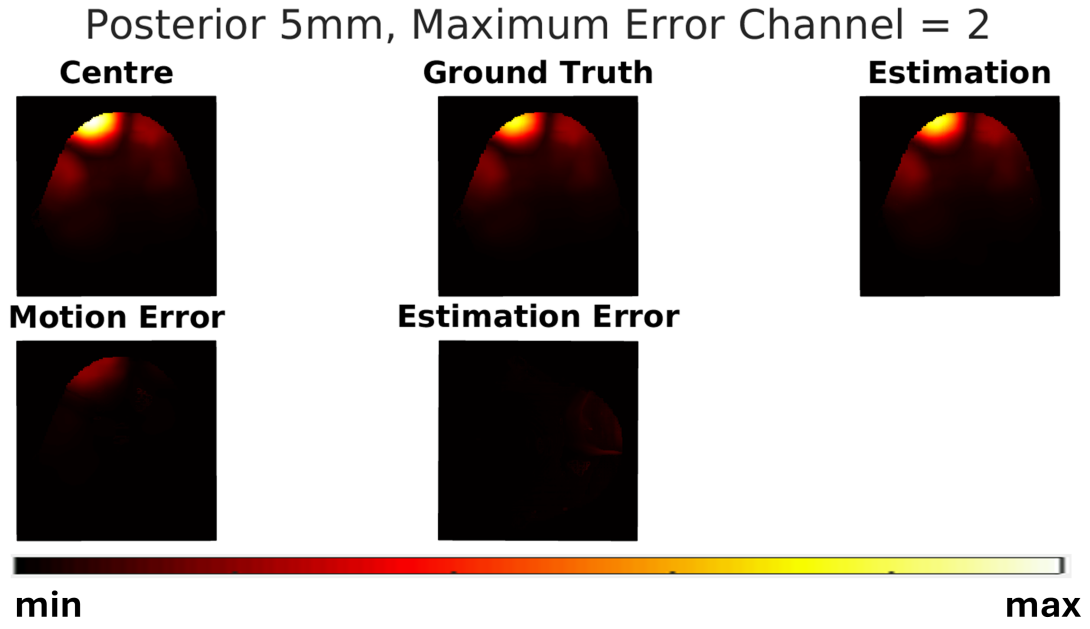


Figure 7.4.2: Q-matrix magnitude maximum intensity projections along z. This figure shows a central slice for channel 2 ($Q_{1,2}$) which yielded the worst network-estimation error.

The nRMSEs per channel are also favorable in terms of improved network-estimation error compared to default-motion error. The U-Net predictions yield lower nRMSE than default motion in 32 out of 36 channels (88.89%). Substantial improvements are seen in channels 1, 2, 8, and 27, where the default-motion error exceeds 25%, but is reduced to under 10% in the network's predictions. A smaller margin of improvement is observed in mid-error channels 3, 6, and 28.

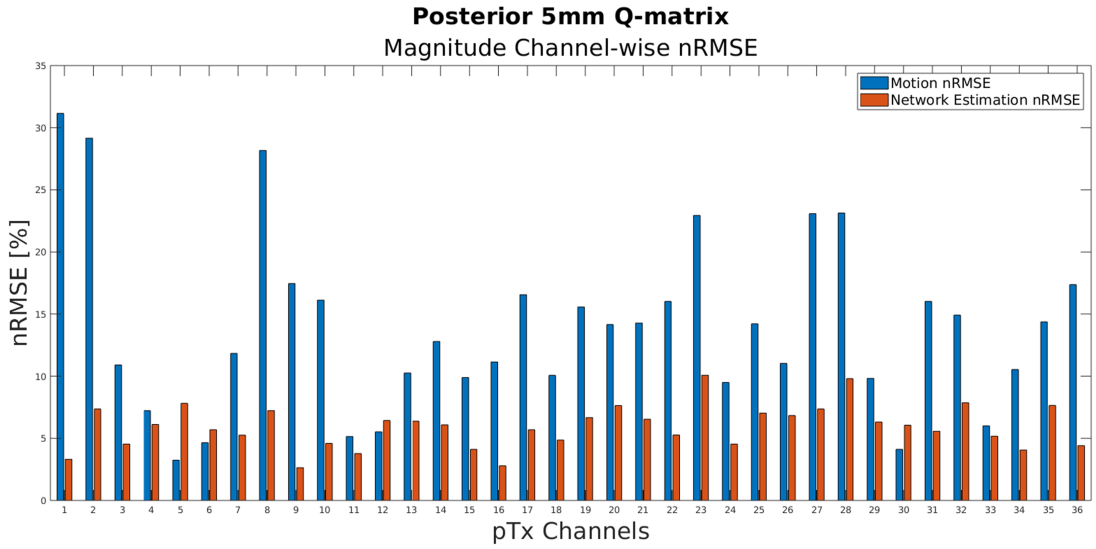


Figure 7.4.3: Bar chart of nRMSE per channel. The blue bars are default-motion error, while the orange bars are network-estimation error.

### Phase

The U-Net with a 3x3 filter size performed better (1.9° error) than the U-Net containing an 8x8 filter (2.25° error), but both were worse than the original motion-induced error (1.52° error). The estimations produced by the U-Net containing the 3x3 filters yielded a 25% error increase, while the estimations produced by the cGAN from Chapter 4 yielded a 66% error increase. While still unsuccessful, the U-Net performed better than the cGAN with a 62.12% reduction in error increase.

In the first row and third column, Figures 7.4.4, 7.4.5, 7.4.6, and 7.4.7 all show that in the estimation image (top row, third column), the U-Net prediction captures the overall structure of the Q-matrix phase distribution, but introduces distinct high-frequency artifacts at the phase wrap boundaries, despite the jittering application. These sharp, well-defined lines occur where the phase value transitions abruptly across the $\pm\,\pi$ threshold. These artifacts are further reflected

in the estimation error map, where the phase wrapping regions exhibit prominent error bands.

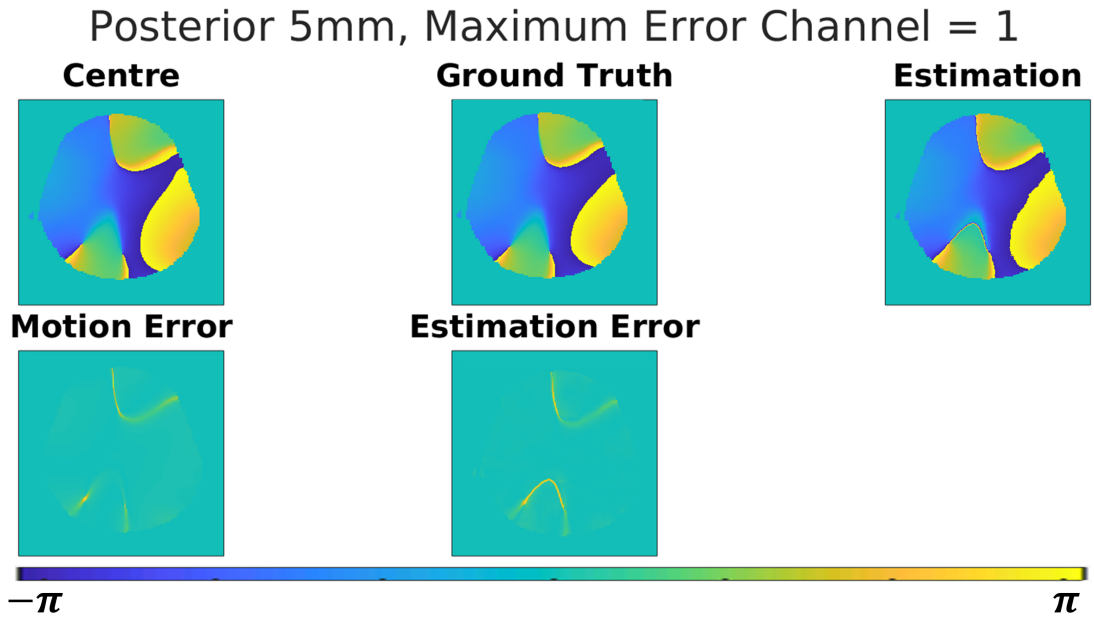


Figure 7.4.4: Error of Q-matrix phases. Estimations arise from a U-Net containing a 3x3 filter size. This figure shows a central slice for channel 1 ($Q_{1,1}$) which yielded the worst default-motion error. The scale is between -pi and pi.

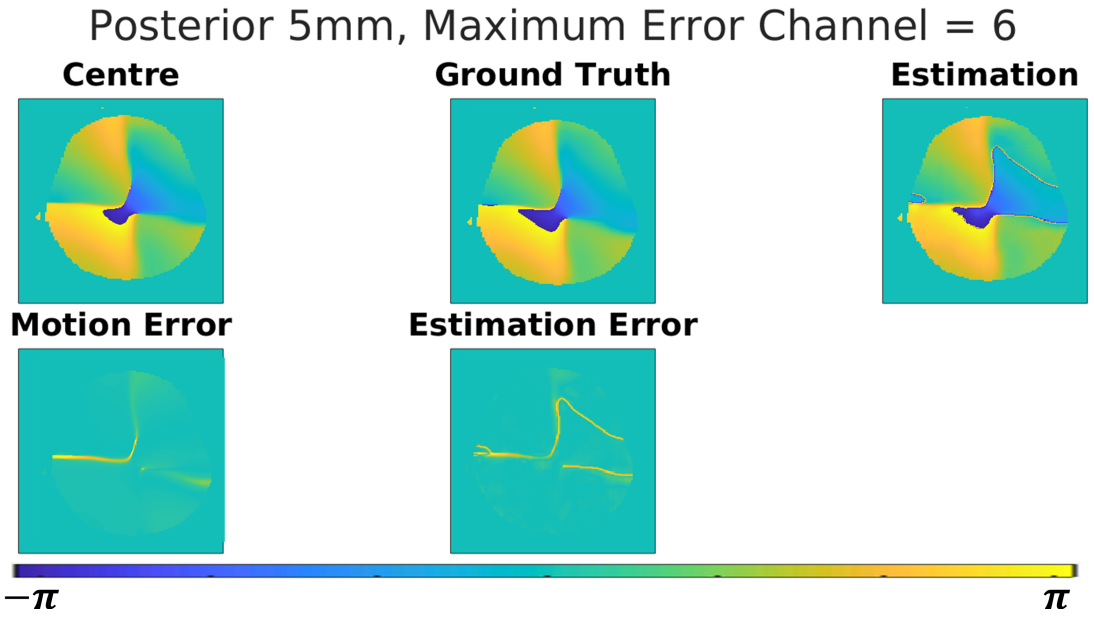


Figure 7.4.5: Error of Q-matrix phases. Estimations arise from a U-Net containing a 3x3 filter size. This figure shows a central slice for channel 6 ($Q_{1,6}$) which yielded the worst network-estimation error. The scale is between $-\pi$ and $\pi$.

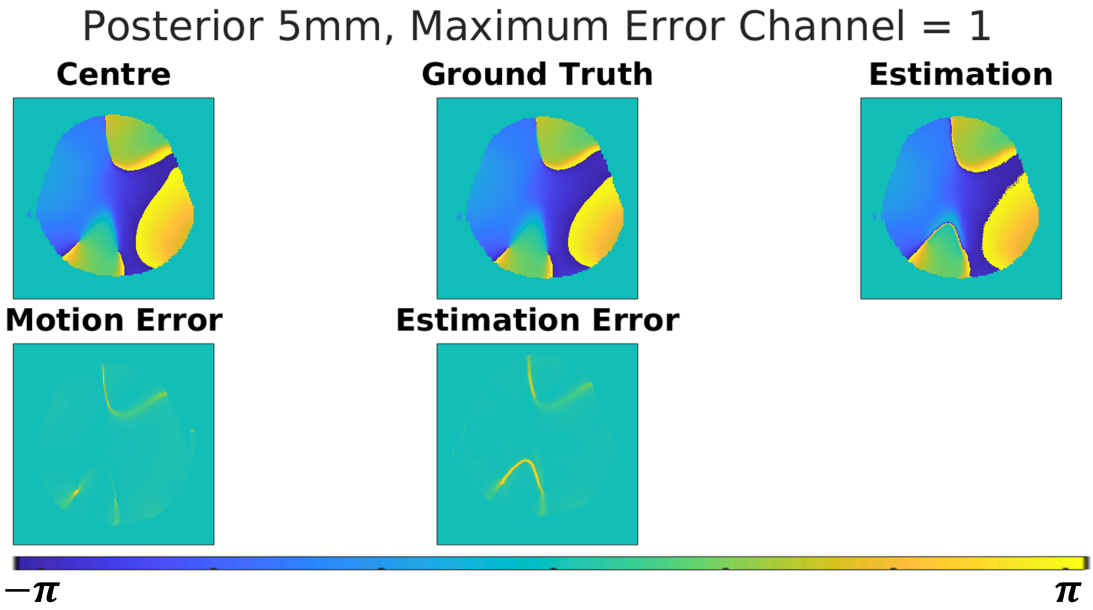


Figure 7.4.6: Error of Q-matrix phases. Estimated images arise from a U-Net containing an 8x8 filter size. This figure shows a central slice for channel 1 ($Q_{1,1}$) which yielded the worst default-motion error. The scale is between $-\pi$ and $\pi$.

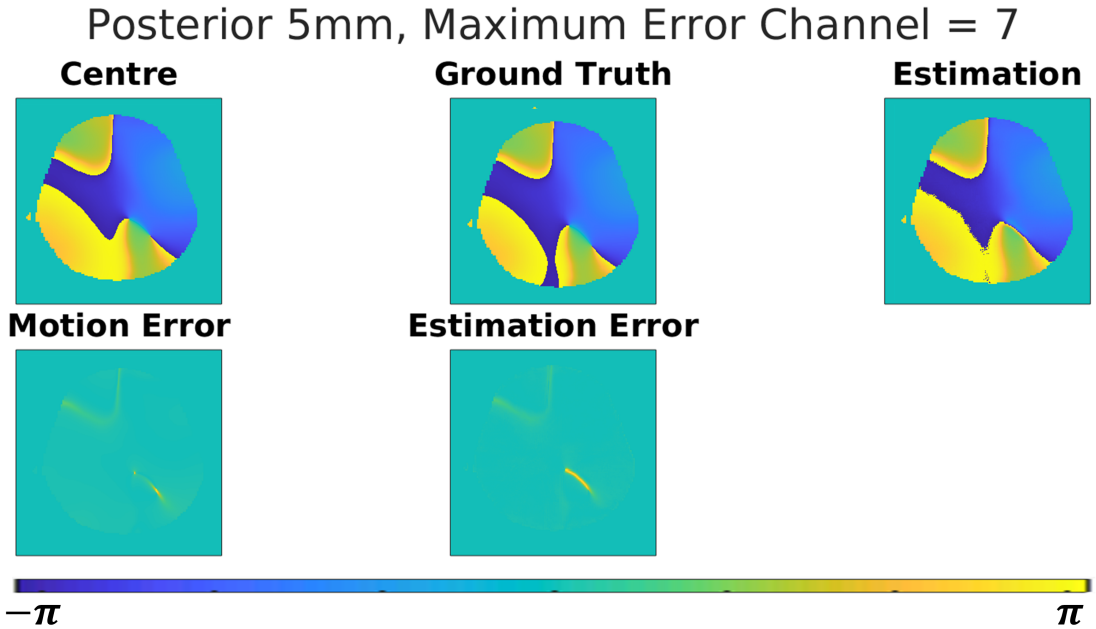


Figure 7.4.7: Error of Q-matrix phases. Estimated images arise from a U-Net containing an 8x8 filter size. This figure shows a central slice for channel 7 ($Q_{1,7}$) which yielded the worst estimation error. The scale is between $-\pi$ and $\pi$.

Figures 7.4.8 and 7.4.9 show ° error per channel. With the exception of channel

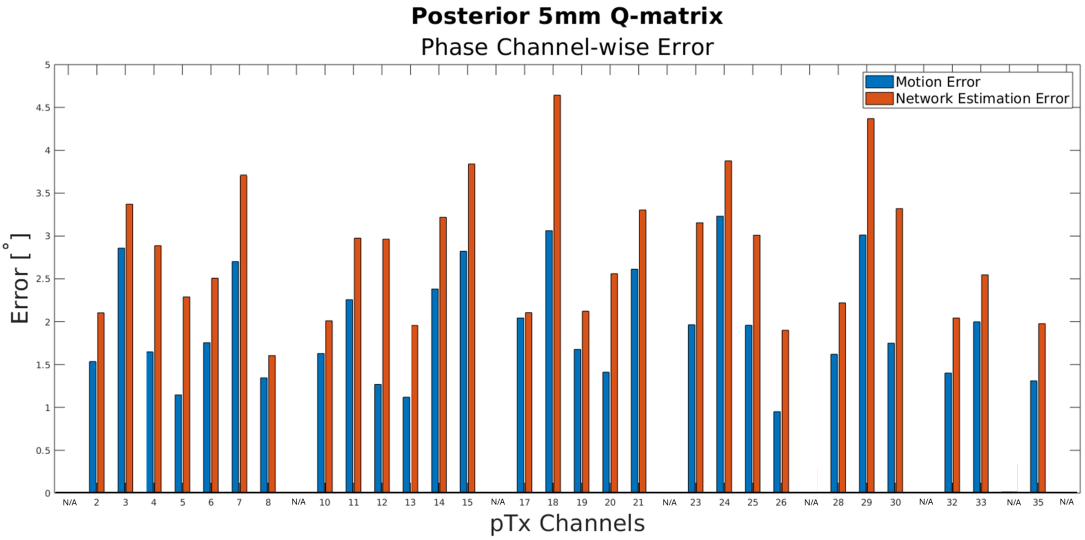


Figure 7.4.8: Bar chart of Motion and Network Estimated ° error per pTx channel for the phases of Q-matrices which were generated by a U-Net training with a **3x3 filter**. The blue bars are default-motion error, while the orange bars are network-estimation error.

8 in the case of the phases generated by the U-Net containing an 8x8 filter, none of the network-estimation errors are lower than the default-motion errors. In both phase cases, the network-estimation error exceeds the default-motion error in the majority of channels. Specifically, for the 3×3 filter network, only 1 out of 36 channels (channel 8) exhibited improved estimation relative to default-motion error, while for the 8×8 filter network, none of the channels showed improvement.

Importantly, the 1st, 9th, 16th, 22nd, 27th, 31st and 36th channels are self-interaction channels where the resulting error is inconsequential. As with Chapter 4, this information can be disregarded, as it is not included in the overall analysis and error metrics.

### P10 mm - Affected Phase

The 10 mm phase investigations yielded a default-motion error of 3.2° compared to a slightly higher network estimation error of 3.42°, yielding a 6.88% error increase caused by the networks. Figures 7.4.10 and 7.4.11 shows that in this case, the errors are also mainly evident around phase wrap boundaries.

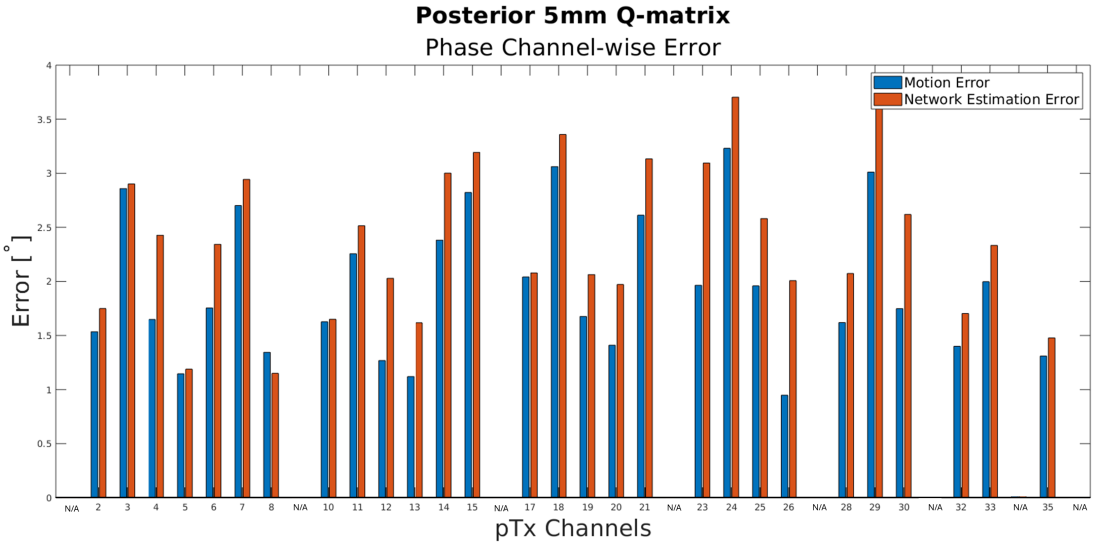


Figure 7.4.9: Bar chart of Motion and Network Estimated ° error per pTx channel for the phases of Q-matrices which were generated by a U-Net training with an **8x8 filter**. The blue bars are default-motion error, while the orange bars are network-estimation error.

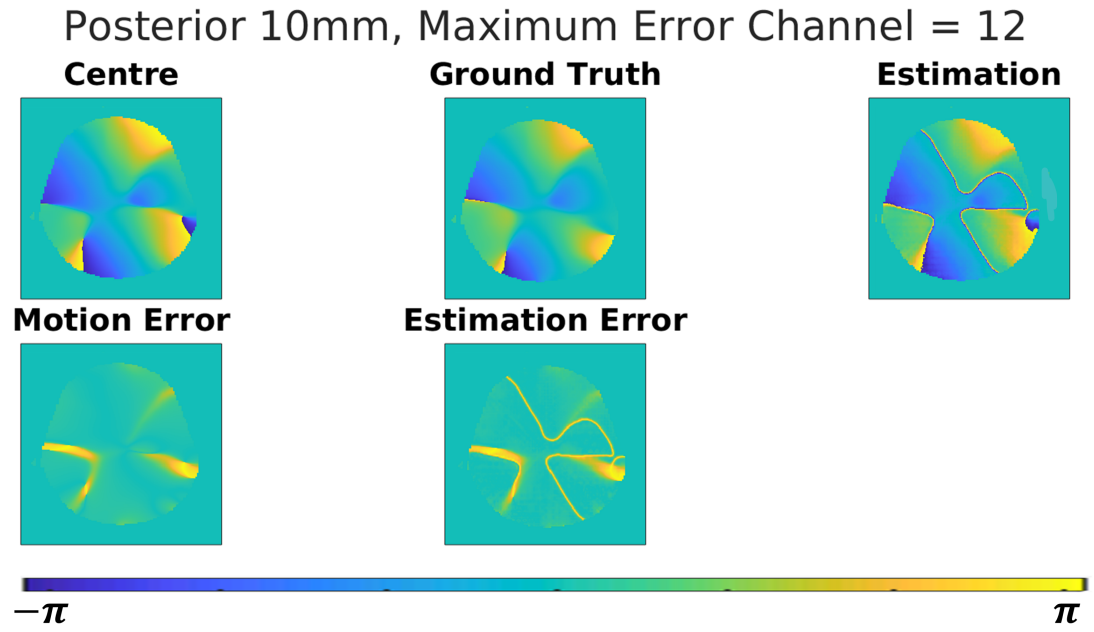


Figure 7.4.10: Error of Q-matrix phases for a posterior 10 mm displacement. Estimated images arise from a U-Net containing an 3x3 filter size. This figure shows a central slice for channel 12 ($Q_{2,5}$) which yielded the worst motion error. The scale is between $-\pi$ and $\pi$.

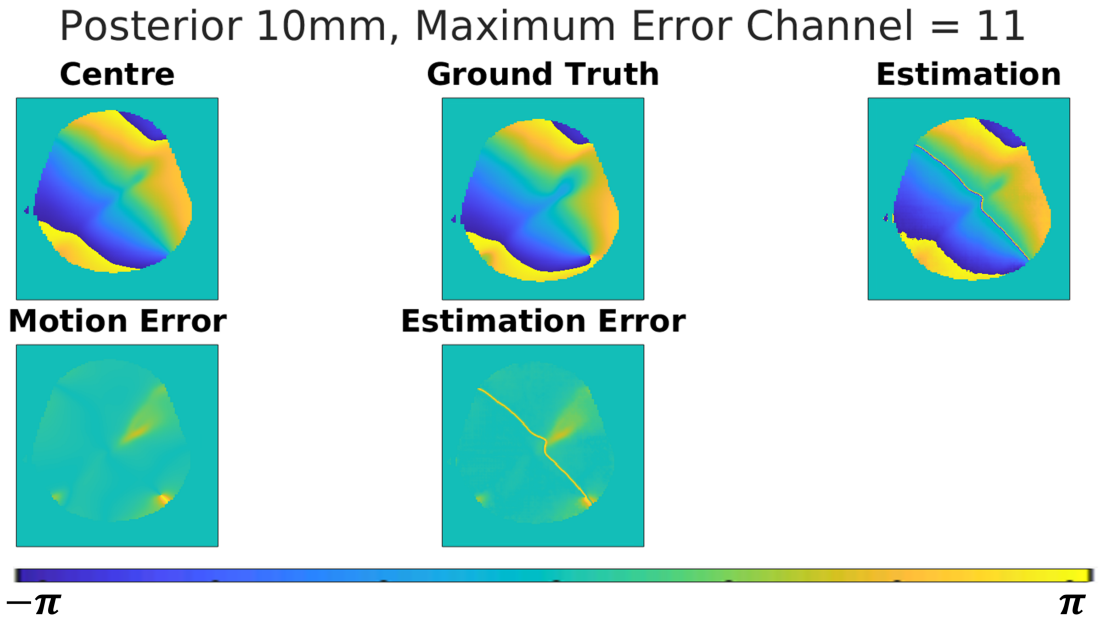


Figure 7.4.11: Error of Q-matrix phases for a 10 mm displacement. Estimated images arise from a U-Net containing an 3x3 filter size. This figure shows a central slice for channel 12 ($Q_{2,4}$) which yielded the worst estimation error. The scale is between $-\pi$ and $\pi$.

Figure 7.4.12 shows degree error per channel, where 46.43% of the channels see improvement by networks.

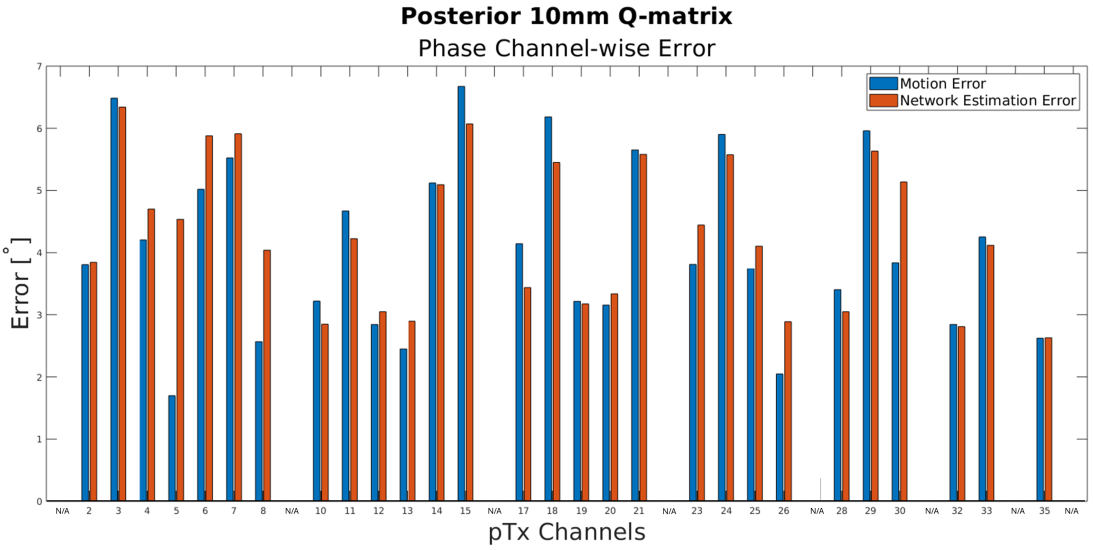


Figure 7.4.12: Bar chart of Motion and Network Estimated ° error per pTx channel for the phases of posterior 10 mm Q-matrices which were generated by a U-Net training with an **3x3 filter**. Half of the channels show some improvement by networks. The blue bars are default-motion error, while the orange bars are network-estimation error.

### Complex Magnitude and Phase

The nRMSE from the 5 mm motion-affected complex Q-matrices using the phase predictions from the neural network using a 3x3 filter size was 20%. while the network-predicted nRMSE was 12.5%. When combining the magnitude estimations with the phase estimations created by a network with an 8x8 filter size, the nRMSE was 13.7%. Figures 7.4.13 and 7.4.14 show the channel-wise error for each case in the order reported.

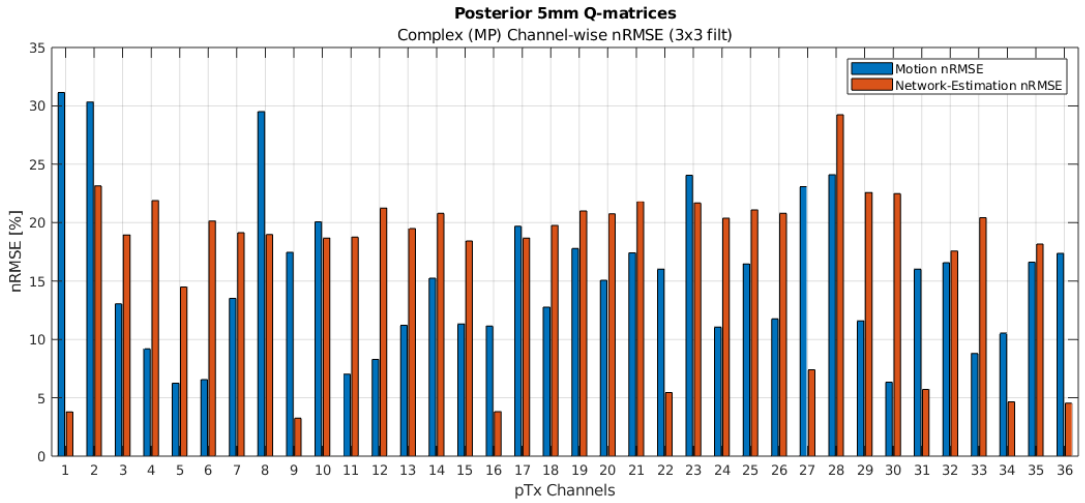


Figure 7.4.13: Bar chart of Motion and Network Estimated ° error per pTx channel for the complex Q-matrices which were generated by a U-Net training with an **3x3 filter**. The blue bars are default-motion error, while the orange bars are network-estimation error.

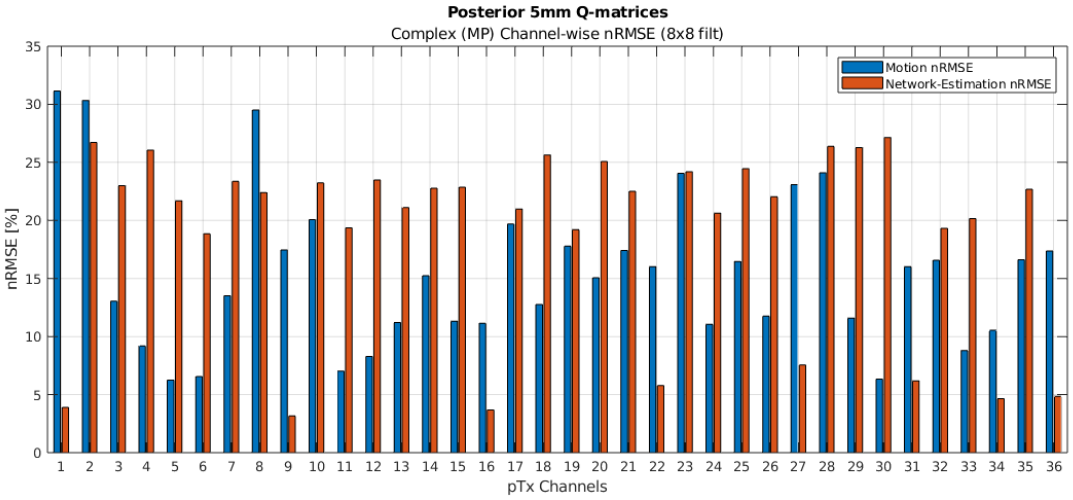


Figure 7.4.14: Bar chart of Motion and Network Estimated ° error per pTx channel for the complex Q-matrices which were generated by a U-Net training with an **8x8 filter**. The blue bars are default-motion error, while the orange bars are network-estimation error.

### 7.4.3 Section Discussion - Magnitude and Phase

The performance of the U-Net on Q-matrix data shows a wide range of outcomes depending on the representation used. For Q-matrix magnitude, the network yielded strong results, indicating that the U-Net generalizes well to this representation. Notably, the U-Net improved performance in most channels, with the largest gains observed in channels with high baseline motion error, suggesting that the network is capable of modeling substantial spatial changes in magnitudes due to displacement. Channels with mid-range motion error showed more modest improvements, indicating that estimation quality may correlate with the scale of field change.

Q-matrix phase estimation was less successful. The U-Net outperformed the cGAN in relative terms, reducing the error increase, but still failed to produce absolute improvements over the default motion baseline. Using a 3×3 filter yielded slightly better results than an 8×8 filter, but both exceeded the default-motion error. Channel-wise analysis revealed that, with the exception of a single case (channel 8 in the 3×3 network), every channel exhibited higher estimation error than baseline motion. Visual inspection of the central slices (Figures 7.4.4, 7.4.5, 7.4.6, and 7.4.7) confirmed that phase wrap boundaries were handled poorly.

As discussed in the $E_x$-field section and Chapter 4, it could be the case that 5

mm translations barely affect the presentation of Q-matrix phases, and therefore that phase-related investigation is inconclusive. Again, while the phase wrap boundaries seem dramatic, they are artificial and could likely be inconsequential if the phase and magnitude Q-matrices were combined and subsequently applied to realistic pTx pulses to calculate predicted local SAR distributions. For that reason, the effect of 10 mm head motion was tested, but results show not improvement by networks. Interestingly, in Figures 7.4.10 and 7.4.11, the network estimated distributions are more similar to the initial position distributions. This could be because the networks that were cascaded to reach the 10 mm motion-affected phases were trained on data where the ground truth and initial position distributions did not differ drastically, and they therefore transferred this pattern to the 10 mm data. It is possible that the pipeline, as it is implemented now, is not suitable for the phase distributions. Additional 10 or more mm trained networks would benefit the pipeline in the future. This would of course create a less coherent and elegant preparation workflow, where the phase and magnitude pipelines differ.

Given the less successful phase estimations, the complex, combined, network-estimated Q-matrices still showed lower nRMSE than motion-affected ones. This investigation does confirm that lower nRMSE of phase predictions does yield overall lower nRMSE of complex Q-matrix predictions, since the better-performing 3x3-filtered Q-matrix, when combined with magnitude Q-matrices into a complex whole yielded lower nRMSE than the complex Q-matrix composed of the same magnitude estimation but the worse-performing 8x8-filtered Q-matrix phase estimation. So while a less favorable phase estimation by networks does not necessarily mean that the method is worse than if the networks were not employed, it does mean that more optimal phase predictions are worth striving for.

### 7.4.4 Methods - Imaginary and Real

The real and imaginary data were processed in the same way as the Q-matrix magnitude data, with normalization applied as described in Chapter 4. The only difference was the use of a global normalization factor instead of a channel-wise approach, to align with the processing pipeline outlined in Chapter 6. The U-Net architecture used was identical to the one employed for the Q-matrix magnitude data in the previous section.

### 7.4.5 Results - Imaginary and Real

#### Imaginary

The U-Net estimated imaginary Q-matrices poorly, yielding a 55.05% error compared to 15.62% motion error, resulting in a 252.48% error increase. The excessive error is evident in Figures 7.4.15 and 7.4.16. This estimation error increase is far worse than the 5% increase resulting from the cGAN estimations. Based on these values, the U-Net was not a better tool than the cGAN.

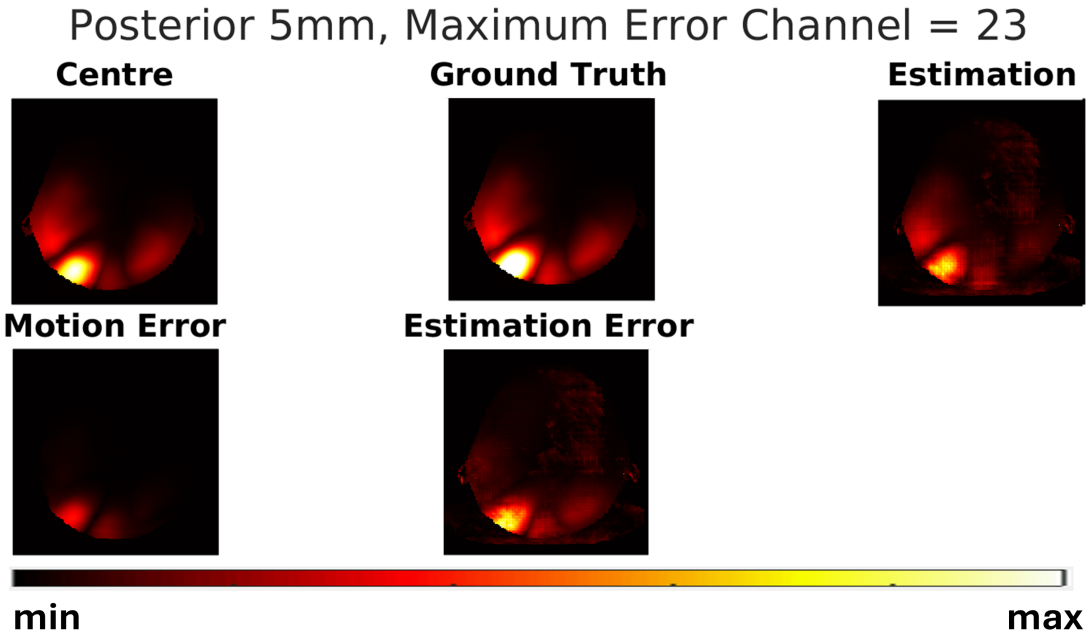


Figure 7.4.15: Error of imaginary component of Q-matrices. This figure shows a maximum intensity projection along the z axis for channel 23 ($Q_{4,8}$) which yielded the worst default-motion error.

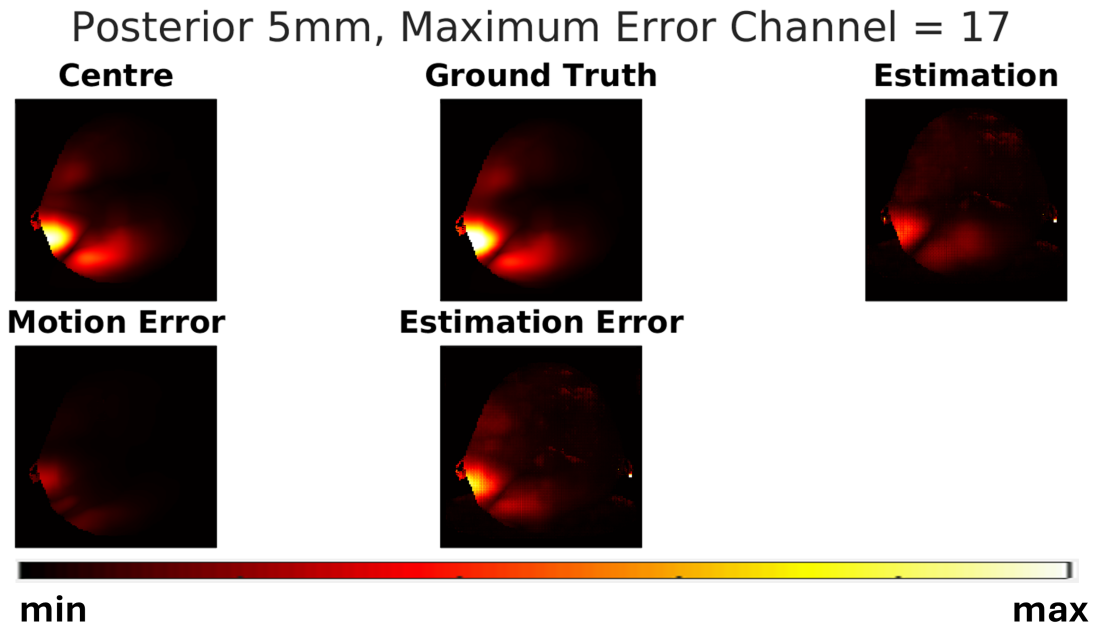


Figure 7.4.16: Error of imaginary component of Q-matrices. This figure shows a maximum intensity projection along the z axis for channel 17 ($Q_{3,7}$) which yielded the worst network-estimation error.

Figure 7.4.17 shows nRMSE per channel for all off-diagonal entries. No channels show error improvement from U-Net estimations.

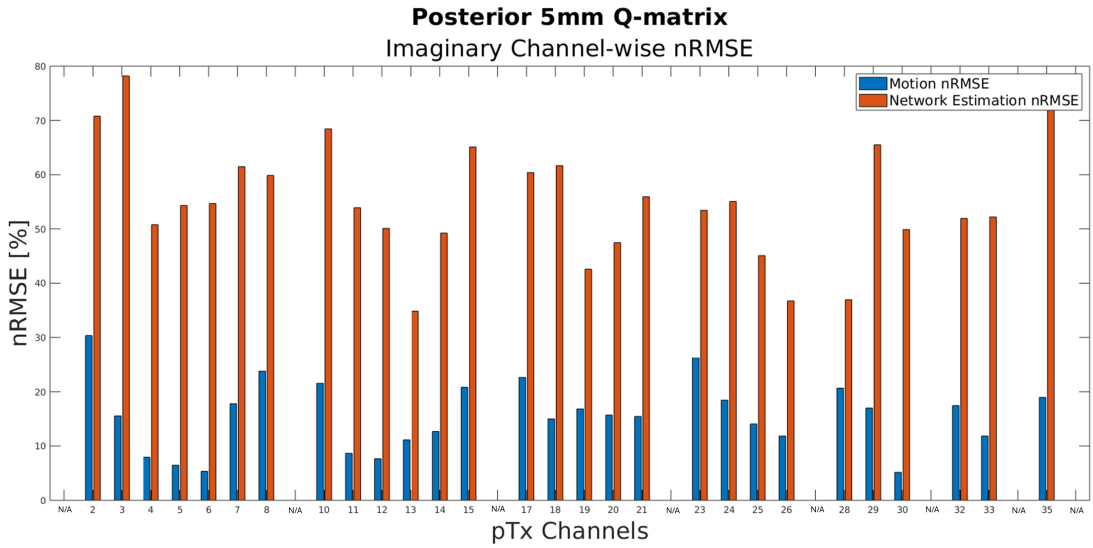


Figure 7.4.17: Bar chart of imaginary Q-matrix nRMSEs per channel. Default motion nRMSE is blue while network-estimation error is orange.

### Real

The U-Net estimated real Q-matrices better than the imaginary Q-matrices, but still did not offer an improvement over default-motion error (Figure 7.4.20). While the network-estimation error was 20.46%, motion error was 19.96%, yielding an error increase (which is markedly lower compared to the cGAN results). On the other hand, the maximum intensity projections (Figures 7.4.18 and 7.4.19) clearly show that the maximum point error is significantly lower, and that the network-estimated images more closely resemble the simulated ground truth compared to the centre-position images. The overall error distribution in the network-estimated images is also more compact. Small regions of high-intensity error are visible near the bottom-left edge of the network-estimated slices and the corresponding error maps in Figure 7.4.18.

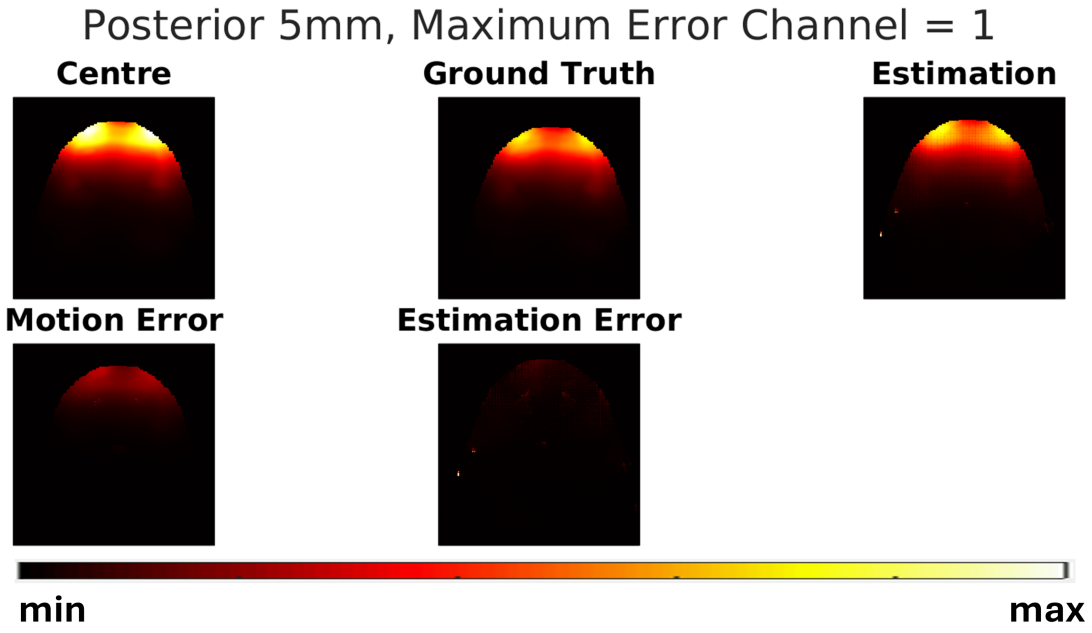


Figure 7.4.18: Error of real component of Q-matrices. This figure shows a maximum intensity projection along the z axis for channel 1 ($Q_{1,1}$) which yielded the worst default-motion error.

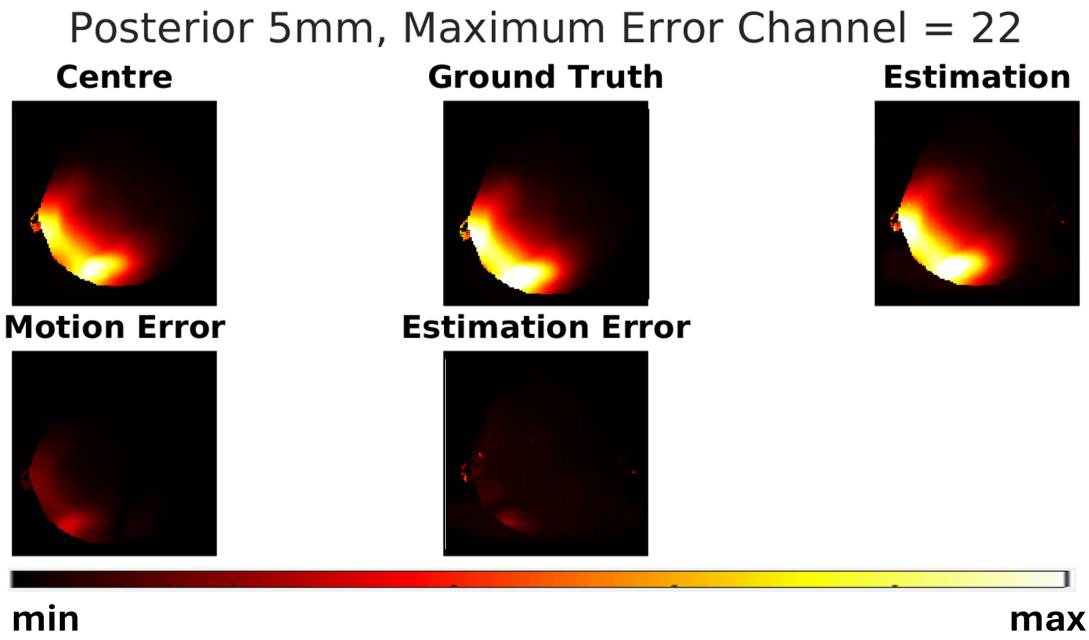


Figure 7.4.19: Error of real component of Q-matrices. This figure shows a maximum intensity projection along the z axis for channel 22 ($Q_{4,7}$) which yielded the worst network-estimation error.

Figure 7.4.20 shows default motion and network estimation nRMSE of the real Q-matrices per channel. Just 22.22% of channels see improvement from U-Net estimations.

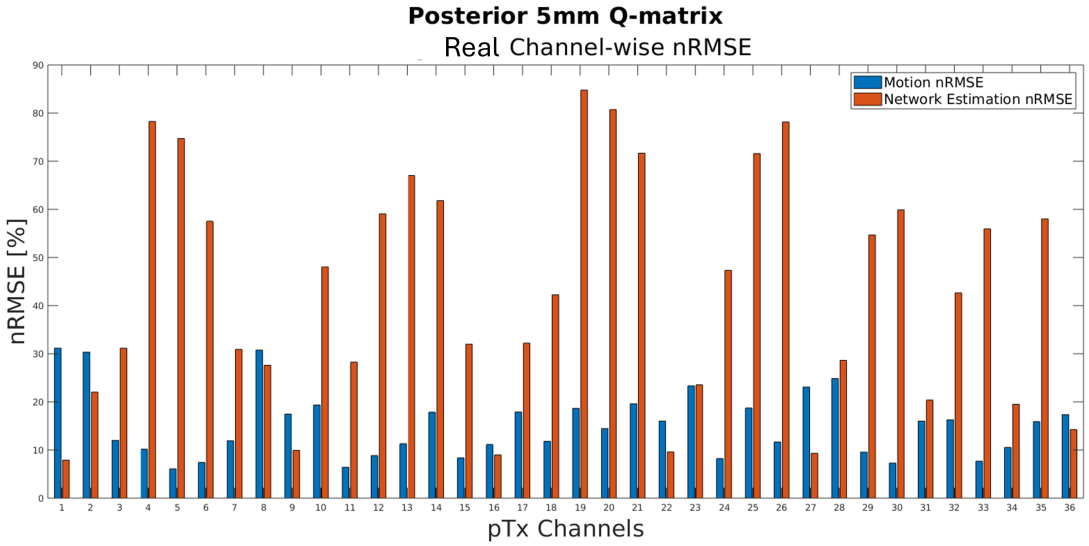


Figure 7.4.20: Bar chart of real Q-matrix nRMSEs per channel. Default motion nRMSE is blue while network-estimation error is orange.

### Complex Real and Imaginary

The nRMSE of the motion-affected complex Q-matrices composed of re-united real and imaginary parts was 20%, while the network-estimated nRMSE was 26.5%. Figure 7.4.21 shows channel-wise nRMSE.

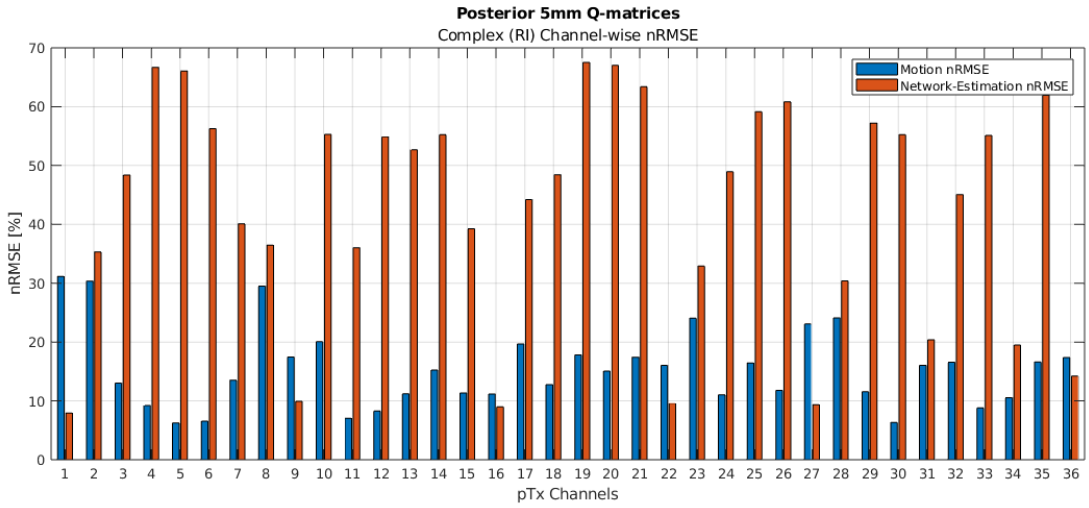


Figure 7.4.21: Bar chart of complex (real and imaginary) Q-matrix nRMSEs per channel. Default motion nRMSE is blue while network-estimation error is orange.

## 7.4.6 Section Discussion - Imaginary and Real

The network's predictions of the imaginary component of the Q-matrix were markedly worse than baseline.

In contrast, results for the real component of the Q-matrix were more promising, as were the re-united real-imaginary Q-matrices. Although the U-Net just failed to outperform default-motion error overall, it significantly outperformed the cGAN. The real component's self-coupling channels had lower network error than motion error, while the cross-coupling channels did not show improvement. It is likely the relatively well-performing real network which dampened the effects of the poor imaginary estimations when interpreting the results by means of the complex Q-matrix.

The pipeline for real and imaginary network estimations can be improved in a similar way discussed in 4, where the real and imaginary parts can be included as individual channels within one slice used as data (ie. one slice is of the dimensions 256x256x36xRxI). This method would be beneficial because it has the potential

to capture motion-induced shifts of energy between the real and imaginary states, thereby eliminating the abstractness of the data. In other words, this way, the network would be trained on more well-rounded contextual information than it is in the methods presented in this chapter.

As it stands, the U-Net performs best on Q-matrix magnitudes, moderately on real components, and poorly on imaginary and phase components. Its success in magnitude prediction likely stems from the spatial smoothness and real-valued quality of the data, while failures in phase and imaginary components indicate that improvement is needed. Some of the poorer performance on phase and imaginary data is alleviated when the correspoding parts are re-united into their complex wholes.

## 7.5 Cumulative Discussion and Conclusion

The results of this chapter demonstrate that the U-Net architecture can effectively model the spatial transformations induced by motion across multiple data representations, but its performance is dependent on the type of data being estimated. Across both $E_x$-field and Q-matrix datasets, U-Nets outperformed cGANs in nearly all scenarios in terms of reduced nRMSE, visualized error maps, training time, and consistency. However, this success was confined to magnitude-based data, with performance degrading for phase, real, and imaginary components. The mediocre or outright poor performance of the phase and imaginary components were dampened by the relatively decent or outright good performance of the real and magnitude networks, respectively.

For $E_x$-field magnitude estimation, the U-Net performed well under channel-wise normalization without Gaussian filtering. Its performance not only surpassed the cGAN, but also did so with a notable reduction in training time. The choice of preprocessing and postprocessing made a difference. Channel-wise normalization preserved local intensity patterns, enabling better learning, while Gaussian filtering increased error by removing high-frequency structural information. In this case, the U-Net's performance exceeded expectations, since $E_x$-field magnitudes are visually complex compared to Q-matrix magnitudes. This may not only be chalked-up to the U-Net itself, but potentially also the optimized preprocessing methods.

In in terms of the $E_x$-field phase predictions, As expected, the more nuanced data

was not more effectively predicted by the less complicated network architecture. The U-Net failed to outperform default-motion error. This was slightly worse than the cGAN' s performance, and visualizations revealed clear artifacts at phase wrap boundaries, though they are artificial and could be inconsequential to subsequent pTx applications. When combined with the promising magnitude estimations into the whole complex variable, the phase estimations dampened the positive magnitude results. Another perspective which could be adopted in this case is that the promising magnitude estimations alleviated the poorer performance of the phase estimations when both parts were re-united into their complex whole. Either way, phase estimation performance cannot be dismissed.

In line with expectations, Q-matrix magnitude estimation was the most successful application of the U-Net overall as well as relative to the cGAN performance. The largest improvements occurred in channels with high motion-induced baseline error, indicating the network's capacity to learn significant spatial transformations. Contrary to expectations for the Q-matrix results outlined in the chapter's introduction, phase estimation for Q-matrices followed a similar pattern to the $E_x$-field results. Although the U-Net reduced the overall error increase relative to the cGAN, it still failed to surpass the default motion baseline. Phase wrapping artifacts were evident, and most channels exhibited increased error, further confirming that the U-Net is not well-suited to wrapped phase estimation. It is still maintained that the results are inconclusive, even given the additional P10 mm tests. The investigations presented in this chapter should not be discarded as unsuccessful or not worth pursuing in the future. On the contrary, interested researchers can be enthused by the cGAN vs. U-Net improvements in magnitude data and seek to concretely determine whether the phase data can be used for MRI safety applications.

The network performed moderately well on the real component of the Q-matrices relative to the cGAN. Interestingly, this representation showed improved performance specifically in self-coupling channels. All of the imaginary component's channels performed worse than motion alone. The imaginary results go against the expectations of the study, since it was expected that the U-Net would perform decently on smoother, more "blob-like" data. It is likely that the Q-matrix's imaginary part is still too nuanced for a more simplified network. On the other hand, expectations were moderately met in terms of the Q-matrix's real part, since it is smoother and therefore likely more suitable to a relatively shallow U-Net than a deeper cGAN.

In line with the above, this investigation seems to confirm that the extent to which data is nuanced and complicated will affect network performance and suitability. At first, this expectation was seemingly not the case since the U-Net outperformed the cGAN for Q-matrix phases but not $E_x$-field phases. On one hand, it was expected that $E_x$ phases would not be successful. On the other hand, the U-Net was also not expected to perform well on Q-matrix phases. A possible explanation for the improved Q-matrix phase predictions compared to the $E_x$-field phase predictions is that the $E_x$-fields are more objectively elaborate and detailed than the Q-matrix phases. This evaluation appears to be true upon visual comparison of the datatypes. While they both contain phase wrap boundaries, the Q-matrix phases appear noticeably smoother between wraps than the $E_x$-field phases. Perhaps there is a threshold of nuance appropriate for the U-Net; likely this threshold is surpassed by the $E_x$ field data but not its Q-matrix counterpart.

Based on the results of these investigations, directions for future work can be suggested. First, increased degrees of motion to estimate the Q-matrix and $E_x$-field phases would reveal the true extent to which the U-Nets or cGANs can predict the effects of motion on these data types. Further postprocessing to disregard the negative effects of the phase wrap boundaries on the nRMSE would result in more accurate interpretations of the network's predictive accuracy. Additionally, combining the magnitude and phase Q-matrices into complex datatypes and using them in the pTx application described in Chapter 6 would reveal whether these estimations are indeed accurate enough to be useful in practice. Other future directions are concerned with altering the pipeline in general, by combining the parts which make up complex-valued data types into separate channels of a single input rather than separating them and running a network for each (magnitude and phase or real and imaginary). Networks given the full picture have the potential to yield more accurate results due to having fuller contextual information. This can also be said for the 3-dimensional E-field vectors which can also be combined into separate channels of a single input to the network. Both applications would alleviate the pipeline from the compounding error produced by the results of individual networks ultimately working towards the estimation of a single complex data type.

Taken together, these results suggest that while the U-Net is a fast and effective tool for estimating motion-induced field transformations, the results are inconclusive for phase and complex-valued inputs. Further investigation for larger

displacements of such datatypes is warranted.

# Chapter 8

# Discussion

## 8.1 Reproducing the Python Environment and Previous Deep Learning Results from the Lab: concluding remarks

This chapter discussed python environment reproducibility issues. It could be the case that the issues stemmed from the moment that the original environment used for the work in Ref. [8] was created. Given all of the information that the author now possesses about python environment creation, it is valid to suspect that the reproducibility issue was caused by not locking the Python environment. Doing so is a necessary step toward reproducibility. When dependencies are allowed to float, even minor version updates can introduce changes that can alter results or forbid a code from running. These discrepancies are rarely flagged by the interpreter or highlighted by any other means, and without version control, it becomes difficult to isolate the source of variation. Freezing the environment, whether through Conda, pip's `requirements.txt`, or Docker images, allows future users to recreate the original software conditions that produced a result. It does not solve every problem, but it removes one major source of ambiguity. Without that, reproducing code after a given period of time can become a Sisyphean task, without clarity or direction for whether errors are occurring as a result of the code or a poorly created environment.

Moreover, this replication effort was conducted at the same research centre where the original Python environment was developed. This provided direct access to

the original researchers and the local computing setup. Such conditions are not usually available to others trying to reproduce similar work. In most cases, reproduction attempts are done remotely, using GitHub repositories and occasional email contact. These efforts often face obstacles including incomplete documentation, unavailable data, or the original authors no longer being reachable or affiliated with the institution.

Even with the benefit of being in the same physical location and having direct communication, the process remained slow and difficult. This shows that if reproduction is this demanding under favorable circumstances, it is likely much harder for researchers working without such access. This experience confirms a widespread issue in scientific research, showing that there is a need for clearer, more accessible, and better-documented outputs so that others can effectively reuse and build on existing work. The lessons learned from the environment reproduction issue relate to the SAR safety problem discussed in this thesis in terms of future applications. If this pipeline is developed for in vivo application, its implementation will inevitably still rely on the foundational python packages and associated builds. The process of introducing such tools into the real world, no matter how distant in the future such an application may be, would be more streamlined if proper documentation and locking procedures are adhered to.

In future work, locking the Python environment should be prioritized as early as possible, ideally before any model training begins. It should be standard practice to document all software versions (including CUDA, cuDNN, Python, and every installed package, as part of the project itself) and use containerization tools where feasible, since it was discovered that even Conda environments can fail to maintain system-level dependencies. An environment should not be treated as an afterthought, but rather as a reproducible, sharable component of the research pipeline.

## 8.2 Exploring The Suitability of Data Types for cGAN Estimation: concluding remarks

A cGAN was employed in this chapter to maintain continuity with previous work in the lab [8] and because of successful cGAN implementation in relevant literature [39]. Moreover, cGANs seemed like a suitable tool to predict the effects of motion on E-fields, Q-matrices, and local SAR distributions because it can

be trained to learn mappings between paired datasets (ie. before and after motion). In this type of data, the spatial location of peaks and the geometry of local hotspots are relevant. It is not sufficient for a predictive model to produce outputs with correct global statistics or mean values. The spatial structure must also be preserved. cGANs are suitable for this task because the discriminator component enforces spatial realism in the predicted outputs, in contrast to models that may overlook structural fidelity.

Despite the potential benefits of cGANs, training them is inherently unstable and sensitive to hyperparameter selection, which can lead to inconsistent performance across datasets or motion conditions.

In scenarios where safety is the most important variable, the lack of interpretability and difficulty in quantifying uncertainty limit the utility of cGANs. Unlike physics-informed models or deterministic regressors, cGANs provide little insight into why a particular prediction was made, which complicates validation and risk assessment. In the case of this project, the only physics information they are provided are the Sim4Life simulated datasets.

Using a cGAN to predict the effects of motion on electromagnetic field data, like E-fields, Q-matrices, and local SAR distributions, was difficult because of the presentation of these data types. While the network architecture was able to generate plausible-looking outputs, quantitative results showed that the predictions often introduced more error than motion itself. The exception in this case was the complex Q-matrix composed of the magnitude and phase part estimations which were no longer as penalized by the effects of the phase-wrap boundaries.

Although initial results with ilSAR were limited, the approach remained promising for its ability to simplify the prediction task by appearing relatively smooth and avoiding explicit phase modeling. It also offered a streamlined, single-network method. The SAR safety problem is likely better approached with a pipeline which has the potential to lead to less compounding error. In other words, since network estimation error is the biggest obstacle for the utility of the tool, minimizing it is one of the main goals of the project. Therefore, a streamlined approach with minimal potential for more error than baseline is most desirable. This could be achieved through the ilSAR method or through including real and imaginary parts of complex datasets as separate channels within a single training datapoint As shown in later chapters, refining this method led to substantial improvements in the accuracy and efficiency of the SAR estimation pipeline.

## 8.3 The Effects of Simulated SAR Data Processing Methods and Network Parameter Tuning on Gridding Artifacts and Network Estimation Accuracy: concluding remarks

The gridding artifact appeared as a structured, checkerboard-like pattern in the predicted outputs. Its recurrence across multiple body models and parameter sets showed that it was more than a cosmetic issue; it reflected deeper problems in how the network was learning spatial structure. Adjustments to the architecture removed the visible artifact but did not consistently improve empirical estimation accuracy. This indicated that the artifact was not the root cause of poor performance but rather a byproduct of the network's limited contextual understanding. As a result, the artifact served as a useful diagnostic marker, revealing where the model failed to form reliable spatial representations that aligned with the ground truth.

Point-wise convolution was likely not the right choice for this task because it removed spatial variation across the receptive field and replaced it with uniform filtering. Although it initially appeared to eliminate the gridding artifact, it is likely that the artifact had not been resolved but rather buried under multiple layers of spatial averaging. The network output looked smoother, but this came at the cost of removing meaningful variation and structure from the ilSAR maps. The uniformity introduced by the $1\times1$ filters prevented the model from learning spatial relationships across neighboring voxels. What looked like an improvement in visual quality was instead likely a flattening of the prediction.

At the same time, removing the gridding artifact can lead to improvements in network performance and a measurable drop in prediction error. In the experiments, configurations that removed gridding often produced outputs with smoother transitions and better visual alignment with ground truth ilSAR maps. This implies that removing the artifact improves the visual appearance of the output and reduces numerical error.

## 8.4 Using U-Nets to Predict the Effects of Head Motion on Simulated Specific Absorption Rate During Hltra-high Field Magnetic Resonance Imaging with Parallel Transmission

A prominent feature of this chapter is the transition from a cGAN to a U-Net as a predictive tool, prompted by the gridding artifact's interference with the network's estimation capabilities. The discriminator in GANs is typically employed to align the output distribution with a target distribution, which is useful in scenarios where multiple valid solutions exist, allowing for the selection of any feasible outcome. However, it was discovered that omitting the discriminator can lead to faster and more stable network convergence.

In our approach, we aimed to mitigate high-frequency components responsible for the gridding, staircase-like artifacts. We observed that the discriminator was not effectively learning the underlying distribution and was instead potentially introducing these artifacts. Given that our desired output inherently exhibits low-frequency characteristics, adopting the MMSE framework proved to be a more appropriate choice for ensuring accurate and artifact-free reconstruction.

With the removal of the discriminator, the network was reduced to a straightforward, deterministic U-Net with standard hyperparameters and fewer layers. This shift brought several advantages. Training time decreased by roughly 16 hours, and the results improved considerably, leading to a functional proof-of-concept for predicting the effects of motion on local SAR. The improvement is likely due to the one-to-one mapping structure of the U-Net, in contrast to the adversarial setup of a cGAN.

Along with a simplification of the networks, the simulated data also underwent a refinement. Since ilSAR distributions are often relatively repetitive across subjects, especially within similar anatomical regions, simplified training data would prevent the network from overfitting to noise or subject-specific patterns that do not generalize well. A smaller, carefully selected subset seemed to help the model focus on learning the dominant spatial relationships, rather than memorizing outliers. In other words, the network performed better when trained on just one-third of the original dataset used in earlier chapters, contradicting the common assumption that more training data always leads to better results.

By training U-Nets to estimate the effects of rigid head motion on ilSAR distributions and converting these into Q-matrices, the method eliminates the need to rely on overly conservative, position-agnostic safety models. This has immediate relevance for improving the safety and performance of pTx protocols at 7T, where safety constraints are a limiting factor. In the future, this approach could be integrated into real-time pTx systems to dynamically update safety margins based on detected motion, potentially reducing the need for broad corrective factors and enabling more efficient pulse design. It also opens the possibility of combining data-driven estimation with subject-specific anatomy, extending its application beyond rigid-body motion to more general variations in body shape, pathology, or population diversity. As simulation resources grow and richer datasets become available, the framework introduced here can serve as a foundation for motion-adaptive safety systems in UHF MRI.

Despite these prospective benefits, any motion-adaptive safety framework must be evaluated against the regulatory requirements of RF safety, where predictive error has direct clinical implications. The clinically relevant consequence of neural network estimation error is SAR underestimation because underestimation has the potential to surpass regulatory limits set by the IEC and FDA. Rather than interpreting nRMSE resulting from just the network estimations in isolation, since network estimation results of ilSAR are indirectly relevant, worst-case underestimation of psSAR following pulse design was reported. Motion alone (ie. without intervention by neural networks) resulted in up to a 2.14-fold increase in psSAR relative to the initial position model. When network-estimated ilSAR maps were used, this worst-case factor was reduced to 1.3. Importantly, even under a conservative worst-case correction, the usable RF power fraction increases from 21% (position unaware model) to 68% (position aware model), indicating that safety margin reduction is 3.2-fold. An important consideration to address is that since the local SAR safety problem concerns live patient safety, no amount of heating-caused damage is permissible. if this method were ever to be used in clinics, the FDA's *Assessing the Credibility of Computational Modeling and Simulation in Medical Device Submissions* [185] would likely be consulted to evaluate model credibility requirements and acceptable uncertainty. In line with these guidelines, the uncertainty would likely be defined relative to the model's context of use and the potential consequences of incorrect estimations. Overall, in the proposed application, the relevant acceptability criterion must consider clinically meaningful underestimation of psSAR.

While this work focused on U-Nets due to a culmination of trial and error and the subsequent realization that they are suitable because of their simplicity and stability, other neural network architectures may also be applicable. While an alternative avenue without a machine learning component at all could also have been taken, the projects described here were focused on exclusively machine learning based solutions to the local SAR-motion problem. Upon reflection, a transformer-based model [186] may also be a suitable tool. A transformer learns how different parts of an input relate to one another, regardless of their position. Unlike CNNs, which use fixed-size filters to process local regions, transformers operate using a mechanism called self-attention. This allows the model to compare all positions in the input simultaneously and assign importance to relationships across distant locations. For example, in text, a word might depend on another word several steps earlier; in an ilSAR map, a change in one region may result from fields transmitted elsewhere in the head. The architecture typically consists of an input embedding layer, one or more self-attention layers, and a feedforward network that produces the final output. In the context of motion-aware local SAR prediction, transformers offer an advantage in that they can account for spatial shifts across the entire body without relying on deep stacks of convolutions to propagate information.

Similarly, physics-informed neural networks (PINNs) [187] may be worth exploring, as they can incorporate known electromagnetic relationships into the learning process, potentially improving physical plausibility in the outputs. They function in contrast to most standard machine learning approaches, which are trained only on data, which can result in outputs that match the data but ignore the physical principles that should govern the system. PINNs incorporate the system's governing equations directly in the loss function. As a result, the network learns to minimize both the difference between its predictions and the data, and how much those predictions deviate from established physics.

This study could also have benefited from more up-to-date software like the pycharm IDE and the pytorch framework as opposed to spyder and TensorFlow. This is mostly for reasons of what is more commonly used by the deep learning community online, and would have provided the author with more guidance during the trouble-shooting stage of the project.

During the troubleshooting phase, several data normalization strategies were tested. Initially, each model was assigned its own normalization factor. This

was followed by separate normalization factors for the training, validation, and testing sets. Next, a channel-wise normalization approach was applied across the entire dataset. Ultimately, it was found that using a single global normalization factor produced the most consistent and reliable results. This process serves as a precautionary example of how different normalization strategies can significantly affect network performance. Small changes in how data is scaled, whether across channels, datasets, or individual models, can lead to large differences in estimation accuracy. Without careful consistency in preprocessing, the network may learn unstable or misleading patterns, making it difficult to interpret results or compare experiments reliably. Normalization is often treated as a routine step, but experience shows that it can be a central factor in whether a model flies or flops.

The study in this chapter is limited by the number of simulated body models available. Our leave-one-out approach showed that our networks can be applied across patient anatomies, since they were trained on an adult female, obese adult male, and elderly male, validated on an adult male, and successfully tested on a female child. This body model configuration was also shuffled to further demonstrate versatility. The results of this approach are more reliable than if we had implemented a interleaved slices approach (like Ref. [184] where both the training and testing networks would contain data from all of the body models, but not overlapping slices), since SAR is 3D, and therefore adjacent slices do not differ enough from one another to be confidently applied as the required 'unseen testing data'.

While the body models configuration was reshuffled, Duke was left out of the testing dataset. In order to successfully apply Duke as a testing dataset, the training dataset would have had to include another body model which reached the same dramatic boundaries as Duke's nose. Similar expansions would be necessary to better accommodate the anatomies of Fats and Glenn, though the network's estimations were acceptable for these cases. Multiple obese, elderly, young, and adult models covering more drastic anatomical variations would be advantageous and yield a more diverse and reliable training dataset which would account for wider anatomical boundaries, and would therefore be more generalizable in vivo. Larger anatomical variabilities are necessary for future successful implementation. An alternative approach to the proposed one-size-fits-all pipeline could be that certain networks are made available given certain patient cases- for example there could be pediatric, adult, obese, and elderly networks created and made avail-

able to technologists for scan preparation, but this approach is less user-friendly because it could cause radiologists and technologists to experience categorization issues, since human anatomy is difficult to discretely categorize.

In the future, generalizability could be improved. As mentioned before, this could arise from gaining more funding to purchase licenses to more body models from the Virtual Population. These body models could then be included in the training and validation datasets, and tests could be conducted to evaluate whether reconfiguring the body models in the data groups for the network inputs would yield more favorable results.

The problem of oversafe or unspecific SAR limits impedes the utility of 7T MRI with pTx. Realistically, considering the variety of morphological characteristics a patient can embody, the number of sequences in which natural patient motion could occur, safety and ethics hurdles for in vivo testing, and IP-protected hardware specification variabilities between different RF head coils located at different imaging centers across the globe, the lengthy timeline from the date of this study until in vivo translation becomes more apparent: this study is an initial implementation meant to instigate further research which could yield results which influence the literature closer to translational implementation. In other words, the problem being tackled by the present study is an important one, but its resolution is met with unaddressed obstacles which are beginning to be addressed here. Even with the noteworthy improvements made to the initial conceptions of this study, the present work is a necessary foundation.

In future work, it would be worth exploring whether a similar pipeline could be used to predict the effects of motion directly on local SAR, using only the initial position $B_1^+$ maps as input. This would remove the intermediate ilSAR or Q-matrix step and instead estimate SAR distributions in their final, safety-relevant form. Doing so would allow for a motion-aware extension to existing studies that use deep learning to predict SAR from $B_1^+$ maps [39, 85], but which assume a static subject position. Incorporating motion into this estimation process would move the field closer to real-time safety assessment, and potentially reduce the need for conservative safety margins that assume worst-case displacement. It would also test whether motion-induced variation can be learned directly from baseline field maps, without requiring separate simulations.

## 8.5 Applying the Successful Network Architecture to the Original Tested Data Types: concluding remarks

The U-Net proved to be a strong performer when applied to relatively smooth data like ilSAR and the magnitudes of Q-matrices, where its structure-preserving architecture was well-matched to the continuity of the signal. That success wasn't as obvious with phase data which need more investigations where the motion-induced error is much larger, as current investigations might have suffered from having motion-induced error being lower than network estimation error floor. It is still reasonable to claim that these studies are inconclusive because none of the results have been evaluated by pTx application.

# Chapter 9

# Conclusion

This thesis investigated how deep learning can support patient safety maintenance during ultra-high field MRI scanning by predicting local SAR distributions under rigid head motion. Rather than focusing on a single technical innovation, the work brought together multiple strands, including model architecture, data preparation, and software complications, into an analysis of how machine learning methods can be applied to this problem.

A prominent goal throughout was to discover ways of responding more effectively to subject motion during imaging. Standard SAR safety margins are often excessively conservative because they must account for unknown variations in position and prioritize patient safety. This work aimed to reduce that uncertainty by using machine learning as a tool to predict the effects of motion on SAR distributions. In doing so, it further develops the existing literature focused on safety planning that is more adaptive to individual patients who may be unable to remain still during scanning.

The work also contributes several considerations for the design of dependable deep learning pipelines in applied research contexts. One area of focus was the role of software infrastructure in ensuring continuity and reproducibility. The reconstruction of earlier environments demonstrated that, without version control and formal documentation, computational workflows are prone to disruption, even when conducted within the same institution. It is stressed that environment specification should be included as a formal component of the research process from the beginning. In parallel, the analysis of network behavior offered insight into model performance through the presence of an artifact. Although unintended, the

artifact provided a means of evaluating how the network processed spatial structure, and its persistence across architectures revealed underlying limitations in representational capacity. Architectural changes removed the visible artifact but did not consistently improve predictive accuracy, indicating that visual quality alone is not a reliable measure of performance. Together, these findings outline a set of practices for improving the transparency and stability of deep learning workflows, by integrating software configuration with model evaluation based on both quantitative results and structured patterns in the output.

Overall, this thesis presents a pathway for continued work in the field. It confirms that machine learning can be a useful tool for local SAR prediction. The research demonstrates how deep learning can be integrated into safety-focused MRI research. It supports future inquiry for further exploration and contributes practical insights for researchers and engineers working on machine learning for medical imaging safety development.

# Appendix A

# Appendix

## A1

The most common errors arising from the process of creating a local environment using the .yml file from $env_{orig}$ were

`egg_info`

and

`Solving environment: failed ResolvePackageNotFound: <every package>`

and

```
Pip subprocess error:
- ERROR: Cannot install -r
<user_path>/condaenv.l_m9jbd9.requirements.txt
(line 13), -r <user_path>/condaenv.l_m9jbd9.requirements.txt
(line 4), -r <user_path>/condaenv.l_m9jbd9.requirements.txt
(line 5) and typing-extensions==3.7.4.3 because these package
versions have conflicting dependencies.
ERROR: ResolutionImpossible
failed
CondaEnvException: Pip failed
```

These error messages indicated that outdated tools caused dependency and package version conflicts. These issues have been reported in other projects [188–190]. To address them, recent studies have proposed tools like PyCRE, which automatically infer compatible environments using domain knowledge graphs (ie. inter-environment compatibility evaluations) [191]. Additionally, the Davos package allows users to specify dependencies directly within Jupyter Notebooks [192] [1] , facilitating reproducibility [193].

This is problematic for reproducing old Python projects which were written before dependency and package upgrades in the pertaining channels and package repositories.

# A2

The following describes the exact ways in which 5 interim environments were created before the final, optimal environment was ultimately discovered. To match the original environment as much as possible, the first tested environment, *env1*, was created with the following commands (the order in which commands are called matters):

```
conda create env -n env1 Python=3.7.11
pip install tf==2.6.0 matplotlib==3.4.3
scipy==1.7.1 keras==2.6.* spyder
conda install cudatoolkit=11.3.1
conda install -c anaconda cudnn
```

[1] Jupyter Notebooks is an open-source web-based tool that allows users to create and share documents that contain live code, equations, visualizations, and text. It is widely used in data science, research, and education because it supports interactive computing and works with many programming languages, including Python. A valuable feature is that it allows users to write and run code in small, separate sections called cells. This makes it easier to test parts of a program, make quick adjustments, and see immediate results without running the entire document each time.

| pip install | conda install | conda install -c anaconda |
|---|---|---|
| tf == 2.6.0 | cudatoolkit = 11.3.1 | cudnn |
| matplotlib == 3.4.3 | | |
| scipy == 1.7.1 | | |
| keras == 2.6.* | | |
| spyder | | |

Table A.0.1: The installed packages (rows) and corresponding installation methods (columns) for *env*1, where tf is tensorflow, matplotlib is a data visualization library, scipy (short for scientific Python) is a computing library, and keras is an open source framework used with tensorflow for machine learning projects. The double equal signs followed by numbers indicate the exact version of the package to be installed. The asterisk at the end of the keras command indicates that the version of keras must be compatible with the version of tf. The '-c' in the Conda install command shown in the final column is to specify that cudnn is installed from the anaconda channel, as opposed to another channel like Conda forge.

Since it is unclear which version of TensorFlow was used for Ref. [8] (though the article reports version 2.3), the newest version at the time of the reproduction, 2.6, was trialed first. Ref. [110] reports not having specified the TensorFlow version when initially creating the environment. The Python version specified for this environment was 3.7.11 because that was the Python version used by Ref. [8], and the most recent Python version available at the time of the environment creation for the pipeline published by Ref. [8].

Subsequent environments were created similarly to the one mentioned above, but with various build and package alterations motivated by error messages and poor or unexpected results from the tested cGAN training scripts. For example, iterations of *env1* were amended with:

*env2*:

```
conda create env -n env2 Python=3.7.11
pip install tensorflow-gpu==2.6.0
matplotlib==3.4.3 scipy==1.4.1 keras==2.3.*
conda install spyder cudatoolkit==11.3.1
conda install -c anaconda cudnn
```

| pip install | conda install | conda install -c anaconda |
|---|---|---|
| tensorflow-gpu == 2.6.0 | spyder | cudnn |
| matplotlib == 3.4.3 | cudatoolkit == 11.3.1 | |
| scipy == 1.4.1 | | |
| keras == 2.3.* | | |

Table A.0.2: The installed packages (rows) and corresponding installation methods (columns) for *env*2. Changes from *env*1 are highlighted in yellow. tf from *env*1 has been replaced by tensorflow-gpu. A different but still tensorflow-gpu 2.6.0 - compatible version of keras was also trialed. Finally, spyder was installed with Conda rather than Pip.

*env*2 (Table A.0.2) was created with TensorFlow-gpu instead of standard TensorFlow because TensorFlow-gpu is optimized to operate on GPUs instead of CPUs. GPUs are more suitable for heavy machine learning tasks, since they allow programs to run on thousands of parallel computing cores, compared to CPUs which only provide tens of cores at most. The alternative version of keras was installed because it is compatible with other necessary libraries in the original environment from Ref. [8]. Finally, spyder was installed through Conda rather than `pip` because installing through `pip` is more risky due to a higher likelihood of package dependency conflicts. The official Anaconda documentation states that installing packages with Pip modifies the Conda environment without Conda's awareness, increasing the likelihood of dependency conflicts when Conda subsequently attempts to modify the environment [194]. Additionally, a discussion on Stack Overflow advises against mixing Pip and Conda installations, as `pip` can install compiled libraries that may be incompatible with those required by Anaconda packages [195].

*env3*:

```
conda create env -n env3 Python=3.7.11
conda install spyder
conda install matplotlib==3.4.3
conda install scipy==1.7.1
conda install scikit-image==0.18.3
pip install tensorflow-gpu==2.6.0
pip install keras==2.6.0
```

| conda install | pip install |
|---|---|
| spyder | tensorflow-gpu == 2.6.0 |
| matplotlib == 3.4.3 | keras == 2.6.0 |
| scipy == 1.7.1 | |
| scikit-image == 0.18.3 | |

Table A.0.3: The installed packages (rows) and corresponding installation methods (columns) for *env*3. Changes from *env*2 are highlighted in yellow.

Installing spyder first in *env*3 (Table A.0.3) was a measure taken to prevent potential dependency conflicts and ensure that Spyder operates correctly within the environment. matplotlib, scipy, and scikit-image were all installed through Conda rather than Pip, although Ref. [110] confirmed that all of the packages in the environment used for Ref. [8] were installed through Pip. This is further potential evidence for the volatility of compatibility in Python packages, libraries, channels and environments over time. Finally, the keras version installed was identical to the version of TensorFlow-gpu.

*env4*:

```
conda create env -n env4 Python=3.7.11
conda install spyder
conda install matplotlib==3.4.3
conda install scipy==1.7.1
conda install scikit-image==0.18.3
pip install tensorflow-gpu==2.3.0
pip install keras==2.3.*
```

| conda install | pip install |
|---|---|
| spyder | tensorflow-gpu == 2.3.0 |
| matplotlib == 3.4.3 | keras == 2.3.* |
| scipy == 1.7.1 | |
| scikit-image == 0.18.3 | |

Table A.0.4: The installed packages (rows) and corresponding installation methods (columns) for *env*4. Changes from *env*3 are highlighted in yellow. The specified Python version was 3.7.11.

*env*4 (Table A.0.4) was nearly identical to *env*3, apart from the TensorFlow-gpu

and keras versions, which were changed to match the TensorFlow version reported in Ref. [8].

*env5*:

```
conda create env -n env5 Python=3.7.9
conda install cudatoolkit==10.0.130
conda install -c anaconda cudnn==7.6.5
conda install tensorflow-gpu==2.3.0
conda install keras==2.3.*
conda install matplotlib==3.3.2
conda install scipy==1.5.2
conda install spyder==4.1.5
```

| **conda install** | conda install -c anaconda |
|---|---|
| cudatoolkit == 10.0.130 | cudnn == 7.6.5 |
| tensorflow-gpu == 2.3.0 | |
| keras == 2.3.* | |
| matplotlib == 3.3.2 | |
| scipy ==1.5.2 | |
| spyder == 4.1.5 | |

Table A.0.5: The installed packages (rows) and corresponding installation methods (columns) for *env*5. Changes from *env*4 are highlighted in yellow. The specified Python version was 3.7.9.

These iterations shown in *env*5 (Table A.0.5) reflect adjustments made to address dependency constraints and ensure compatibility between TensorFlow, CUDA, and cuDNN. The error messages (verbatim undocumented) revealed that using an alternative version of Python could be beneficial, therefore Python version 3.7.9 replaced version 3.7.11. Opposing official TensorFlow documentation [196], version 2.3.0 was installed with Conda instead of Pip. *env*5 was created exclusively through Conda and the anaconda channel. Package and library versions were altered from previous environments to be compatible, but nonetheless, *env*5 proved unusable (errors displayed below).

# A3

Examples of false-alarm OOM (out of memmory) errors are:

```
W tensorflow/core/common_runtime/bfc_allocator.cc:338]
Garbage collection: deallocate free memory regions
(i.e., allocations) so that we can re-allocate a
larger region to avoid OOM due to memory fragmentation.
```

and

```
W tensorflow/core/common_runtime/bfc_allocator.cc:272]
Allocator (GPU_0_bfc) ran out of memory trying to allocate
1.09GiB with freed_by_count=0.
```

# A4

The following are examples of GPU recognition errors:

```
W tensorflow/core/common_runtime/gpu/gpu_device.cc:1835]
Cannot dlopen some GPU libraries. Skipping registering GPU devices...
```

and

```
E tensorflow/stream_executor/cuda/cuda_event.cc:29]
Error polling for event status: failed to query event:
CUDA_ERROR_LAUNCH_FAILED: unspecified launch failure
F tensorflow/core/common_runtime/device/device_event_mgr.cc:221]
Unexpected Event status: 1
```

# A5

### Google Colab Results

Google Colab was first trialed with two epochs. While training was smooth for both magnitude and phase networks, the testing results still yielded NaNs for the phase data when using the identical pipeline to Ref. [8].

Interestingly, when using the same network designed by Ref. [8] for magnitude data for both magnitude and phase data, the results were acceptable, given the hyperparameter allowances (nb. while still training over just two followed by five epochs). Exact error metrics were not recorded, but it was subsequently concluded that the data and the scripts were not the source of the issue, but rather that our local Python environments were to blame.

Because of subscription constraints, training over 10 or more epochs was not permitted on the platform.

# A6

The instructions provided by the authors of StarGAN2 are:

```
conda create -n stargan-v2 Python=3.6.7
conda activate stargan-v2
conda install -y pytorch=1.4.0
torchvision=0.5.0
cudatoolkit=10.0
-c pytorch
conda install x264==’1!152.20180717’
ffmpeg=4.0.2 -c conda-forge
pip install opencv-Python==4.1.2.30
ffmpeg-Python==0.2.0
scikit-image==0.16.2
pip install pillow==7.0.0
scipy==1.2.1 tqdm==4.43.0 munch==2.5.0
```

# A7

## Foreword

This abstract was presented at the 2024 meeting of the International Society for Magnetic Resonance in Medicine in Singapore. It was co-authored by Alix Plumley, Shaihan Malik, and Emre Kopanoglu. Katherine Blanter is first author.

## Synopsis

Specific absorption rate (SAR), a proxy measure for tissue heating, is affected by patient motion. SAR safety factors during MRI scanning are intentionally overconservative. While designed to ensure patient safety, it impedes the utility of scanning with parallel-transmit (pTx) 7T MRI. We successfully used deep learning to predict the location of hot spots during head motion and applied them to a pTx design method which considers patient motion. We report that

hot spots are overcalculated almost 1.5-fold when the degree of patient motion is not included compared to when it is.

### Impact

Deep learning-estimated local specific absorption rate (SAR) variations caused by patient motion may be combined with within-scan motion detection and single-position, subject-specific models for personalized SAR predictions to create personalized SAR models for patients who cannot remain still.

## Introduction

Transmitted radio frequency (RF) wavelengths for ultra-high field (UHF) MRI are often inhomogenous and cause an increase in the prevalence of tissue heating, or specific absorption rate (SAR) [197]. RF inhomogeneity can be mitigated by parallel RF transmission (pTx) which can lead to localized SAR increases in unexpected locations. The literature shows that these effects worsen with unplanned patient motion [177–180, 198]. Previous work [8] used conditional Generative Adversarial Networks (cGANs) to predict variations in $B_1^+$ from head motion. The current work applies a similar approach to predict variations in local SAR from head motion and validates them with a near-real-time pTx pulse redesign method [47].

[199] proposed using cGANs to predict the effects of rightward-posterior (R-P) motion on the magnitudes of 36-channel Q-matrices (8x8 channels of complex conjugates yielding 36 unique entries) derived from simulated head slabs. Predicting the phases of Q-matrices was unsuccessful, therefore a fraction of the pipeline was conceived. An algorithm [132] was applied to recover phase information by creating real-valued 64-channel intermediary local SAR matrices (ilSAR). Last year's preliminary pipeline was rewritten, completed, and is presented here.

### Methods

The dataset used was the same as the one generated for [177], which consists of Sim4Life simulations (ZMT, Zurich, Switzerland) of body models (BMs) Billie, Duke, Fats, Ella and Glenn from the Virtual Population (IT'IS, Zurich, Switzerland) [50] at 34 positions (combinations of rightward-leftward (R-L):-20/-

10/-5/0/5/10/20 and anterior-posterior (A-P):-10/-5/0/5/10). 8x8-channel Q-matrices were derived using the approach in [177].

ilSAR were calculated with an algorithm [132] designed to predict the power absorption consequence of a given RF pulse design (Figure A.0.1).

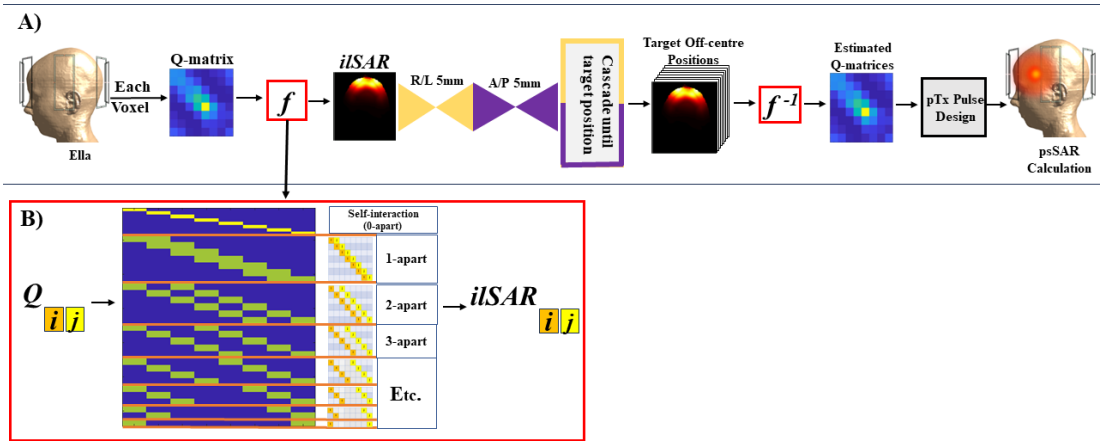


Figure A.0.1: A) The preprocessing and evaluation workflow, where the first trained generator (trained on R/L/5/P 5mm translations) receives ilSAR distributions at the center position before running sequentially until a determined translated position is reached (cascaded). The evaluated ilSAR are inverted and introduced to the pulse-redesign protocol which yields peak spatial SAR calculation. B) The algorithm which calculated ilSAR from Q-Matrices.

The cGAN was implemented in TensorFlow 2.4.1 and Python 3.7.11 on a NVIDIA DGX GPU in Linux. The architecture was identical to pix2pix [83], excluding these parameters: epochs = 60; filter size = 1; strides = 1; a ReLu replaced with a leaky ReLu and no batch normalization in the generator network during downsampling. The data was normalized channel-wise, labeled with the degree of motion, and paired as given input and output ilSAR distribution. A leave-one-out approach was used to create the training, validation, and testing data to prevent cross-talk (ie. TRAINING: Fats, Duke, and Billie; VALIDATION: Glenn; TESTING: Ella). To create the training pairs, the degrees of motion were paired both center-out and with displacement given an off-center reference (eg. R0 mm $\rightarrow$ R5 mm and R5 mm $\rightarrow$ R10 mm both represent R5 mm displacements). The training data contained 5,040 pairs for the P-A and 4,200 for the R-L networks. Testing was conducted using the same magnitude and direction of movement, and with the multiples of that trained magnitude direction and movement (ie. 'cascading' (Figure A.0.1)).

Network estimation quality was evaluated with L1 norm and averaged across slices per channel at each position by comparing ground truth (GT) voxel-wise with network estimated (NE) ilSAR. This was compared to L1 norm of motion-induced (MI) error (MIerr) (displaced GT simulations vs. centered ilSAR).

R-P ilSAR estimations were remapped to Q-matrices [42] and applied to a tailored near-real time pTx pulse redesign method [47] (Figure A.0.1).

## Results and Discussion

Figure A.0.2 shows that the NE ilSAR distributions resemble the GT across slices and movement types. 0-3 cascades are displayed from channel combinations yielding the worst error. MIerr maps (GT-centred) are more pronounced than the NE error (NEerr) maps (GT-NE).

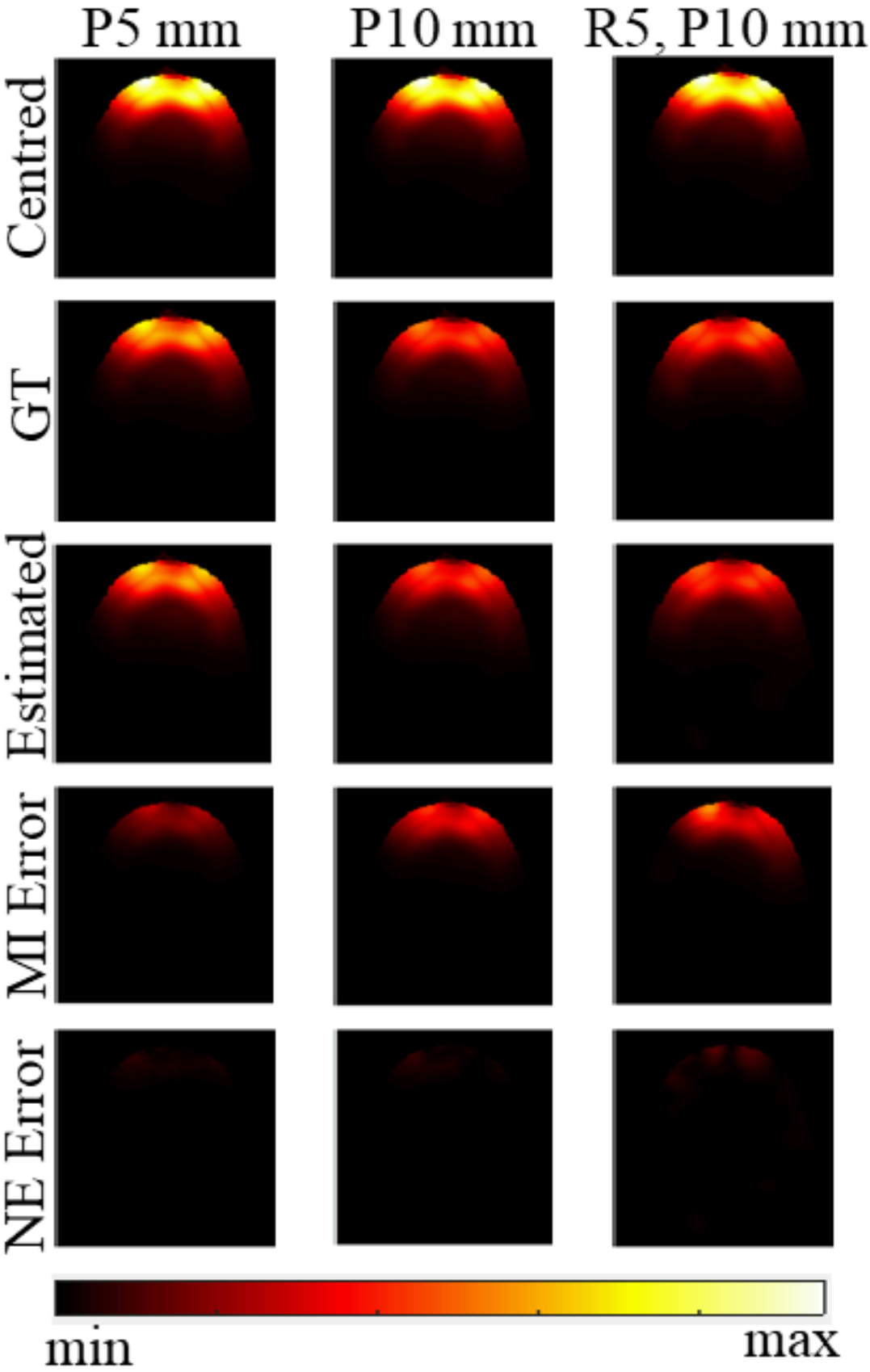


Figure A.0.2: Each column contains maximum intensity projections along the z axis for the P5mm (0x cascades), P10mm (2x cascades), and R5, P10 mm (3x cascades) displacements arising from the channel interactions which yield the worst error. NEerr is far lower than MIerr.

Figure A.0.3 plots the mean and maximum L1 errors across all pTx channel combinations and slices, per degree and direction of motion. In all cases, the mean and maximum MIerrs exceed NEerrs. Mean NEerr remains low throughout. While mean NEerr was 0.1%, never exceeding 0.7%, mean MIerr was 0.3%, reaching 1.5%.

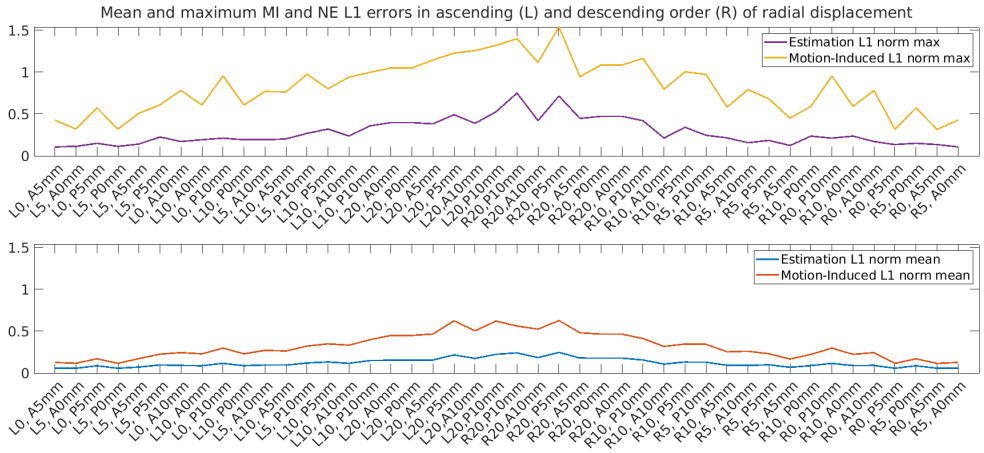


Figure A.0.3: For all movement types, channels and slices, the mean and maximum NE L1 error was lower than MI L1 error. R and L are for rightward and leftward, respectively, and P and A are posterior anterior, respectively. Values are displayed in ascending(L)/descending(R) order of radial displacement from the centre.

Figure A.0.4 displays the similarity between the R-P Q-matrices derived from the NE ilSAR distributions and the GT when including both as constraints in a near-real-time pTx design protocol. The resulting worst case underestimation is 2.5-fold when calculating psSAR with the centered BM (cBM), and 1.3-fold with the NE-BM. Using the cBM leads to 45% overcalculation, while using the NE-BM leads to 0.04% undercalculation.

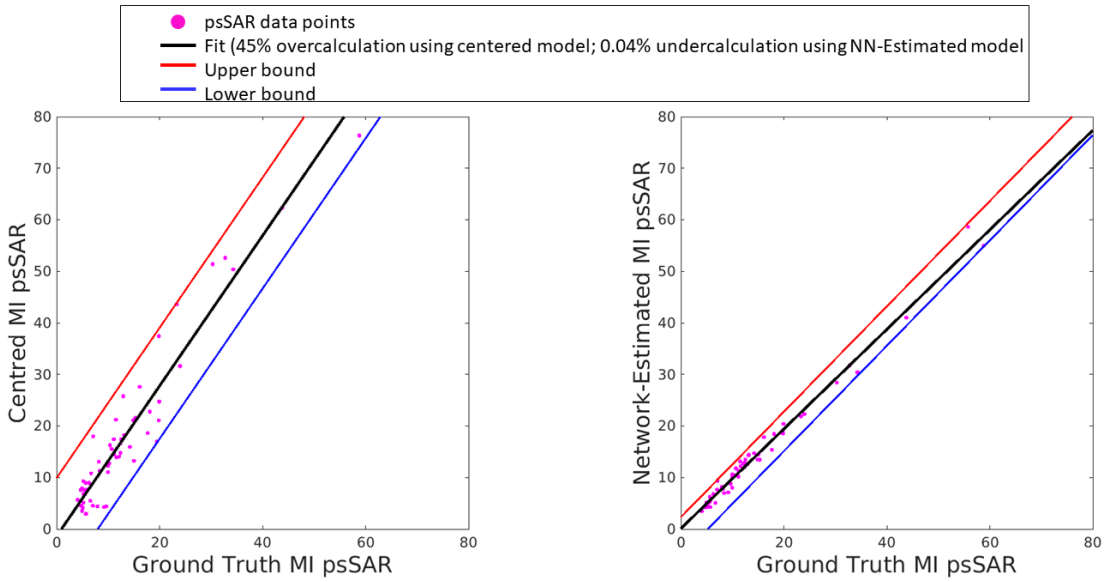


Figure A.0.4: When used as real time pTx pulse design constraints, NE maps yield psSAR values nearly identical to those output when simulated GT maps are used, unlike the difference between centered and simulated GT values. The estimation error margin is one-third smaller when using the NE-BMs compared to the cBMs. Moreover, the slope of the black line indicates that using the cBMs leads to 45% psSAR overcalculation compared to 0.04% undercalculation when using the NE-BMs.

## Conclusion

We have established a pipeline to estimate local SAR variations arising from head motion during 8-channel pTx at UHF MRI. Since our NE and GT ilSAR distributions are compatible, this approach can be developed for near-real-time pTx with revised, less overconservative safety factors. Future work will expand the pipeline to include yaw and 2mm degrees of motion.

## Acknowledgments

This project was supported in part by the Wellcome Trust [204824/Z/16/Z], Welsh Government [Wales Data Nation Accelerator project], and EPSRC [Doctoral Training Program].

# Bibliography


[1] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[2] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241, 2015.

[3] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2016: 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part II 19*, pages 424–432, 2016.

[4] Moez Krichen. Generative adversarial networks. *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 1–7, 2023.

[5] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[6] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[7] MathWorks. *Train Conditional Generative Adversarial Network (cGAN).* The MathWorks, Inc., 2025. https://se.mathworks.com/help/deeplearning/ug/train-conditional-generative-adversarial-network.html.

[8] Alix Plumley, Luke Watkins, Matthias Treder, Patrick Liebig, Kevin Murphy, and Emre Kopanoglu. Rigid motion-resolved prediction using deep learning for real-time parallel-transmission pulse design. *Magnetic Resonance in Medicine*, 87:2254–2270, 5 2022.

[9] A. Walker R. Fisher, S. Perkins and E. Wolfart. Gaussian smoothing, 1997. Accessed: 2025-03-05.

[10] Hann function. https://en.wikipedia.org/wiki/Hann_function. Wikipedia, accessed on March 13, 2025.

[11] Donald B Plewes and Walter Kucharczyk. Physics of mri: a primer. *Journal of magnetic resonance imaging*, 35(5):1038–1054, 2012.

[12] Thomas C Cosmus and Michael Parizh. Advances in whole-body mri magnets. *IEEE Transactions on applied superconductivity*, 21(3):2104–2109, 2010.

[13] John David Jackson. *Classical Electrodynamics*. Wiley, 3rd edition, 1998.

[14] SG Patching. Nmr-active nuclei for biological and biomedical applications. *Journal of Diagnostic Imaging in Therapy*, 3(1):7–48, 2016.

[15] Robert W Brown, Y-C Norman Cheng, E Mark Haacke, Michael R Thompson, and Ramesh Venkatesan. *Magnetic resonance imaging: physical principles and sequence design*. John Wiley & Sons, 2014.

[16] GH Glover, CE Hayes, NJ Pelc, WA Edelstein, OM Mueller, HR Hart, CJ Hardy, M O'donnell, and WD Barber. Comparison of linear and circular polarization for magnetic resonance imaging. *Journal of Magnetic Resonance (1969)*, 64(2):255–270, 1985.

[17] Tamer S Ibrahim, Robert Lee, Brian A Baertlein, Amir M Abduljalil, Hui Zhu, and Pierre-Marie L Robitaille. Effect of rf coil excitation on field inhomogeneity at ultra high fields: a field optimized tem resonator. *Magnetic resonance imaging*, 19(10):1339–1347, 2001.

[18] Felix Bloch. Nuclear induction. *Physical Review*, 70(7–8):460–474, 1946.

[19] Gregory J Stanisz, Elizabeth E Odrobina, J Pun, M Escaravage, Simon J Graham, Michael J Bronskill, and R Mark Henkelman. T1, t2 relaxation and magnetization transfer in tissue at 3t. *Magnetic Resonance in Medicine*, 54(3):507–512, 2005.

[20] Sunder S. Rajan. *Magnetic Field Gradient Pulses and Spatial Encoding of MR Signal*, pages 12–24. Springer New York, New York, NY, 1998.

[21] Catherine Westbrook and John Talbot. *MRI in Practice*. John Wiley & Sons, 2018.

[22] Michael Markl and Jochen Leupold. Gradient echo imaging. *Journal of Magnetic Resonance Imaging*, 35(6):1274–1289, 2012.

[23] Paul M Margosian, Gordon DeMeester, and Haiying Liu. Partial fourier acquisition in mri. *Encyclopedia of nuclear magnetic resonance*, 5:3463–3467, 1996.

[24] Christopher M Collins and Zhangwei Wang. Calculation of radiofrequency electromagnetic fields and their effects in mri of human subjects. *Magnetic resonance in medicine*, 65(5):1470–1482, 2011.

[25] International Electrotechnical Commission et al. Medical electrical equipment-part 2-33: Particular requirements for the basic safety and essential performance of magnetic resonance equipment for medical diagnosis. *IEC 60601-2-33 Ed. 3.0*, 2010.

[26] Frank G. Shellock and Emanuel Kanal. Safety of magnetic resonance imaging: Implications for cardiovascular patients. *Cardiology Clinics*, 18(3):623–651, 2000.

[27] C Kc Chou, Howard Bassen, J Osepchuk, Q Balzano, R Petersen, M Meltz, R Cleveland, JC Lin, and L Heynick. Radio frequency electromagnetic exposure: Tutorial review on experimental dosimetry. *Bioelectromagnetics: Journal of the Bioelectromagnetics Society, The Society for Physical Regulation in Biology and Medicine, The European Bioelectromagnetics Association*, 17(3):195–208, 1996.

[28] Harry H Pennes. Analysis of tissue and arterial blood temperatures in the resting human forearm. *Journal of applied physiology*, 1(2):93–122, 1948.

[29] Astrid LHMW van Lier, Alexis NTJ Kotte, Bas W Raaymakers, Jan JW Lagendijk, and Cornelis AT van den Berg. Radiofrequency heating induced by 7t head mri: Thermal assessment using discrete vasculature or pennes' bioheat equation. *Journal of Magnetic Resonance Imaging*, 35(4):795–803, 2012.

[30] Rolf Pohmann, Oliver Speck, and Klaus Scheffler. Signal-to-noise ratio and mr tissue parameters in human brain imaging at 3, 7, and 9.4 tesla using current receive coil arrays. *Magnetic resonance in medicine*, 75(2):801–809, 2016.

[31] Kamil Uğurbil. Imaging at ultrahigh magnetic fields: History, challenges, and solutions. *Neuroimage*, 168:7–32, 2018.

[32] Kamil Uğurbil. Magnetic resonance imaging at ultrahigh fields. *IEEE transactions on biomedical engineering*, 61(5):1364–1379, 2014.

[33] Thomas M Fiedler, Mark E Ladd, and Axel K Bitz. Uhf-mri: Applications, challenges, and solutions. *Journal of Magnetic Resonance Imaging*, 47(6):1460–1474, 2018.

[34] Andrew G Webb. Dielectric materials in magnetic resonance. *Concepts in Magnetic Resonance Part A*, 38A(4):148–184, 2011.

[35] Francesco Padormo, Arian Beqiri, Joseph V Hajnal, and Shaihan J Malik. Parallel transmission for ultrahigh-field imaging. *NMR in Biomedicine*, 29(9):1145–1161, 2016.

[36] Anuj Sharma, Roland Bammer, V Andrew Stenger, and William A Grissom. Low peak power multiband spokes pulses for b1+ inhomogeneity-compensated simultaneous multislice excitation in high field mri. *Magnetic resonance in medicine*, 74(3):747–755, 2015.

[37] Xiaotong Zhang, Sebastian Schmitter, Pierre-François Van de Moortele, Jiaen Liu, and Bin He. From complex rm b1 mapping to local sar estimation for human brain mr imaging using multi-channel transceiver coil at 7t. *IEEE transactions on medical imaging*, 32(6):1058–1067, 2013.

[38] T Voigt, H Homann, U Katscher, and O Doessel. Patient-individual local sar determination: in vivo measurements and numerical validation. *Magnetic resonance in medicine*, 68(4):1117–1126, 2012.

[39] E. F. Meliadò, A. J.E. Raaijmakers, A. Sbrizzi, B. R. Steensma, M. Maspero, M. H.F. Savenije, P. R. Luijten, and C. A.T. van den Berg. A deep learning method for image-based subject-specific local sar assessment. *Magnetic Resonance in Medicine*, 83:695–711, 2 2020.

[40] Emre Kopanoglu, Cem Deniz, Tolga Cukur, Kamil Uğurbil, and Cem M Deniz. The importance of subject positioning in local sar for parallel-transmit mri. *Magnetic Resonance in Medicine*, 84(3):1446–1460, 2020.

[41] Fernando Bardati, Antonello Borrani, Annamaria Gerardino, and Giorgio A Lovisolo. Sar optimization in a phased array radiofrequency hyperthermia system. *IEEE Transactions on biomedical engineering*, 42(12):1201–1207, 1995.

[42] Arian Beqiri, JV Hajnal, and SJ Malik. Local q-matrix computation for parallel transmit mri using optimal channel combinations. *Proceedings of the 24th Annual Meeting of ISMRM, Singapore*, page 3658, 2016.

[43] Emre Kopanoglu, Cem Deniz, Tolga Cukur, Kamil Uğurbil, and Cem M Deniz. Simulation-based safety evaluation of parallel transmission (ptx) protocols under subject motion. *Magnetic Resonance in Medicine*, 86(1):359–373, 2021.

[44] Minghui Tang and Toru Yamamoto. Progress in understanding radiofrequency heating and burn injuries for safer mr imaging. *Magnetic Resonance in Medical Sciences*, 22(1):7–25, 2023.

[45] Jana G Delfino, Daniel M Krainak, Stephanie A Flesher, and Donald L Miller. Mri-related fda adverse event reports: A 10-yr review. *Medical physics*, 46(12):5562–5571, 2019.

[46] Emre Kopanoglu. Actual patient position versus safety models: Specific absorption rate implications of initial head position for ultrahigh field magnetic resonance imaging. *NMR in Biomedicine*, 36(5):e4876, 2023.

[47] Emre Kopanoglu. Near real-time parallel-transmit pulse design. *Proceedings of the 27th Annual Meeting of ISMRM. Paris, France*, 2018.

[48] Emre Kopanoglu. Redesigning parallel-transmit pulses in runtime to correct for the effect of patient motion on b1+-maps for ultraagehigh-field mri. May

2024. Presented at: 2024 ISMRM & ISMRT Annual Meeting & Exhibition, 4-9 May, 2024.

[49] Esra Neufeld, Marie-Christine Gosselin, Dominik Szczerba, Martin Zefferer, and Niels Kuster. Sim4life: A medical image data based multiphysics simulation platform for computational life sciences. *Proceedings of the VPH 2012 Congress*, 2012.

[50] Andreas Christ, Wolfgang Kainz, Eckhart G Hahn, Katharina Honegger, Marcel Zefferer, Esra Neufeld, Wolfgang Rascher, Rolf Janka, Werner Bautz, Ji Chen, et al. The virtual family—development of surface-based anatomical models of two adults and two children for dosimetric simulations. *Physics in Medicine & Biology*, 55(2):N23, 2009.

[51] Martin D. Maas. Open-source electromagnetic simulation: Fdtd, fem, mom. Blog post on Epsilon Forge, March 2025. An up-to-date review of open-source electromagnetic simulation tools.

[52] ZMT Zurich MedTech AG. *Sim4Life*. ZMT Zurich MedTech AG, Zurich, Switzerland, 2024.

[53] Tamer S Ibrahim, Robert Lee, Brian A Baertlein, and PM L Robitaille. B1 field homogeneity and sar calculations for thebirdcage coil. *Physics in Medicine & Biology*, 46(2):609, 2001.

[54] Thomas Wolf, Kay Nehrke, and Peter Bornert. Impact of realistic modeling of rf shimming on sar prediction and b1+ homogeneity in a 7 tesla head coil. *Magnetic Resonance in Medicine*, 69(5):1621–1630, 2013.

[55] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[56] Michael A Nielsen. *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA, USA, 2015.

[57] Antonio Gulli and Sujit Pal. *Deep learning with Keras*. Packt Publishing Ltd, 2017.

[58] M Sornam and M Poornima Devi. A survey on back propagation neural network. *International Journal of Communication and Networking System*, 5(1):70–74, 2016.

[59] Andrew Ng. Deep learning specialization. `https://www.coursera.org/specializations/deep-learning`, 2025. Coursera specialization offered by DeepLearning.AI.

[60] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[61] TensorFlow Authors. Tensorboard: Tensorflow's visualization toolkit. `https://www.tensorflow.org/tensorboard`, 2024. Accessed: 2025-05-26.

[62] Florian Knoll, Kerstin Hammernik, Chi Zhang, Steen Moeller, Thomas Pock, Daniel K Sodickson, and Mehmet Akcakaya. Deep-learning methods for parallel magnetic resonance imaging reconstruction: A survey of the current approaches, trends, and issues. *IEEE signal processing magazine*, 37(1):128–140, 2020.

[63] Yun Liu, Qing Zhang, and Christopher M Collins. Deep learning-based electromagnetic field predictions for mri safety. *NeuroImage*, 220:117082, 2020.

[64] Elisa Moya-Sáez, Óscar Peña-Nogales, Rodrigo de Luis-García, and Carlos Alberola-López. A deep learning approach for synthetic mri based on two routine sequences and training with synthetic data. *Computer Methods and Programs in Biomedicine*, 210:106371, 2021.

[65] Toygan Kilic, Patrick Liebig, Omer Burak Demirel, Jürgen Herrler, Armin M Nagel, Kamil Ugurbil, and Mehmet Akçakaya. Unsupervised deep learning with convolutional neural networks for static parallel transmit design: A retrospective study. *Magnetic resonance in medicine*, 91(6):2498–2507, 2024.

[66] Essam A Rashed, Jose Gomez-Tames, and Akimasa Hirata. Deep learning-based development of personalized human head model with non-uniform conductivity for brain stimulation. *IEEE transactions on medical imaging*, 39(7):2351–2362, 2020.

[67] Felix Krueger, Christoph Stefan Aigner, Kerstin Hammernik, Sebastian Dietrich, Max Lutz, Jeanette Schulz-Menger, Tobias Schaeffter, and Sebastian Schmitter. Rapid estimation of 2d relative b 1+-maps from localizers in the human heart at 7t using deep learning. *Magnetic Resonance in Medicine*, 89(3):1002–1015, 2023.

[68] Ana Carolina Alves, André Ferreira, Behrus Puladi, Jan Egger, and Victor Alves. Deep dive into mri: Exploring deep learning applications in 0.55 t and 7t mri. *arXiv preprint arXiv:2407.01318*, 2024.

[69] Maciej A Mazurowski, Mateusz Buda, Ashirbani Saha, and Mustafa R Bashir. Deep learning in radiology: An overview of the concepts and a survey of the state of the art with focus on mri. *Journal of magnetic resonance imaging*, 49(4):939–954, 2019.

[70] Jin Liu, Yi Pan, Min Li, Ziyue Chen, Lu Tang, Chengqian Lu, and Jianxin Wang. Applications of deep learning to mri images: A survey. *Big Data Mining and Analytics*, 1(1):1–18, 2018.

[71] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[72] Yuchen Pei, Lisheng Wang, Fenqiang Zhao, Tao Zhong, Lufan Liao, Dinggang Shen, and Gang Li. Anatomy-guided convolutional neural network for motion correction in fetal brain mri. *Machine Learning in Medical Imaging: 11th International Workshop, MLMI 2020, Held in Conjunction with MICCAI 2020, Lima, Peru, October 4, 2020, Proceedings 11*, pages 384–393, 2020.

[73] Stanislav Motyka, Paul Weiser, Beata Bachrata, Lukas Hingerl, Bernhard Strasser, Gilbert Hangel, Eva Niess, Fabian Niess, Maxim Zaitsev, Simon Daniel Robinson, et al. Predicting dynamic, motion-related changes in b 0 field in the brain at a 7t mri using a subject-specific fine-trained u-net. *Magnetic resonance in medicine*, 91(5):2044–2056, 2024.

[74] Xi Wang, Xiaofan Jia, Shao Ying Huang, and Abdulkadir C Yucel. Rapid specific absorption rate estimation of high-field mri via 3d u-net architectures for mri safety. 2024.

[75] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[76] Cassandra Czobit and Reza Samavi. Cyclegan models for mri image translation. *arXiv preprint arXiv:2401.00023*, 2023.

[77] Yingchao Cai, Mengxiao Li, Shiqi Liu, and Changhao Zhou. Cyclegan-based image translation from mri to ct scans. *Journal of Physics: Conference Series*, 2646(1):012016, 2023.

[78] Leonardo Crespi, Samuele Camnasio, Damiano Dei, Nicola Lambri, Pietro Mancosu, Marta Scorsetti, and Daniele Loiacono. Leveraging multimodal cyclegan for the generation of anatomically accurate synthetic ct scans from mris. *arXiv preprint arXiv:2407.10888*, 2024.

[79] Chenjie Ni. Image-to-image translation based on cyclegan: From ct to mri. *Proc. 1st Int. Conf. on Data Analysis and Machine Learning-DAML*, pages 229–33.

[80] Bamidele O Awojoyogbe and Michael O Dada. Democratizing access to magnetic resonance imaging: Capacity building in quantum computing. In *Digital Molecular Magnetic Resonance Imaging*, pages 345–348. Springer, 2024.

[81] Lukas Fetty, Mikael Bylund, Peter Kuess, Gerd Heilemann, Tufve Nyholm, Dietmar Georg, and Tommy Löfstedt. Latent space manipulation for high-resolution medical image synthesis via the stylegan. *Zeitschrift für Medizinische Physik*, 30(4):305–314, 2020.

[82] Daniel Güllmar, Wei-Chan Hsu, and Jürgen R Reichenbach. Predicting disease-related mri patterns of multiple sclerosis through gan-based image editing. *Zeitschrift für Medizinische Physik*, 34(2):318–329, 2024.

[83] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. pages 1125–1134, 2017.

[84] Ettore Flavio Meliadò, Alessandro Sbrizzi, Cornelis AT van den Berg, Bart R Steensma, Peter R Luijten, and Alexander JE Raaijmakers. Conditional safety margins for less conservative peak local sar assessment: a probabilistic approach. *Magnetic Resonance in Medicine*, 84(6):3379–3395, 2020.

[85] Sayim Gokyar, Chenyang Zhao, Samantha J Ma, and Danny JJ Wang. Deep learning-based local sar prediction using b 1 maps and structural mri of the head for parallel transmission at 7 t. *Magnetic resonance in medicine*, 90(6):2524–2538, 2023.

[86] Wyger M Brink, Sahar Yousefi, Prernna Bhatnagar, Rob F Remis, Marius Staring, and Andrew G Webb. Personalized local sar prediction for parallel transmit neuroimaging at 7t from a single t1-weighted dataset. *Magnetic resonance in medicine*, 88:464–475, 2022.

[87] Yida Wang, David Joseph Tan, Nassir Navab, and Federico Tombari. Forknet: Multi-branch volumetric semantic completion from a single depth image. *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8608–8617, 2019.

[88] Xi Wang, Xiaofan Jia, Shao Ying Huang, and Abdulkadir C Yucel. Deep learning-based prediction of specific absorption rate induced by ultra-high-field mri rf head coil. *IEEE Journal of Electromagnetics, RF and Microwaves in Medicine and Biology*, 2025.

[89] Suchita Mukherjee, Abigail Almanza, and Cindy Rubio-González. Fixing dependency errors for python build reproducibility. *Proceedings of the 30th ACM SIGSOFT international symposium on software testing and analysis*, pages 439–451, 2021.

[90] NumPy contributors. BUG: np.min promotion changed from 1.26 to 2.0. `https://github.com/numpy/numpy/issues/26710`, 2024. Accessed: 2025-05-21.

[91] Moataz Sayagh, Martin Dagenais, and Bram Adams. Apiscanner: Detecting and classifying deprecated apis in python libraries. *arXiv preprint arXiv:2102.09251*, 2021.

[92] python-poetry contributors. Documentation for older versions of poetry. `https://github.com/python-poetry/poetry/issues/9297`, 2024. Accessed: 2025-05-21.

[93] Jiawei Wang, Li Li, and Andreas Zeller. Restoring execution environments of jupyter notebooks. *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 1622–1633, 2021.

[94] Anaconda Inc. *Conda: A Cross-platform Package Manager*, 2022.

[95] Oren Ben-Kiki, Clark Evans, and Ingy döt Net. *YAML Ain't Markup Language (YAML) Version 1.2*. 2009.

[96] Conda Development Team. *Conda Issues: Version Discrepancies in Dependencies*, 2020.

[97] Anaconda, Inc. *Managing Packages with Anaconda*, 2024. Accessed: 1 March 2024.

[98] Conda-Forge Community. *What is Conda-Forge?*, 2024. Accessed: 19 June 2024.

[99] Python Package Index (PyPI). *pip: The Python Package Installer*, 2024.

[100] ActiveState. *Dependency Management with Pip: Python's Package Manager*, 2024.

[101] Spyder Development Team. *Spyder Installation Guide*, 2024. Accessed: 3 April 2024.

[102] IT'IS Foundation. *Virtual Population (ViP)*. IT'IS Foundation, Zurich, Switzerland, 2024.

[103] Tim Shaffer, Kyle Chard, and Douglas Thain. An empirical study of package dependencies and lifetimes in binder python containers. *2021 IEEE 17th International Conference on eScience (eScience)*, pages 215–224, 2021.

[104] ActiveState. *Python Dependencies: Everything You Need to Know*, 2021.

[105] Inc. Anaconda. Conda cheat sheet. `https://docs.conda.io/projects/conda/en/latest/user-guide/cheatsheet.html`, Accessed: 2024. Accessed on July 3, 2024.

[106] Stack Overflow. Python version change after importing environment .yml file, 2022. Accessed: 2024-07-05.

[107] Carl Boettiger. An introduction to docker for reproducible research. *ACM SIGOPS Operating Systems Review*, 49(1):71–79, 2015.

[108] Panagiotis Gkikopoulos, Valerio Schiavoni, and Josef Spillner. Analysis and improvement of heterogeneous hardware support in docker images. *IFIP*

*International Conference on Distributed Applications and Interoperable Systems*, pages 125–142, 2021.

[109] ContinuumIO. Conda environment creation from .yml file results in version discrepancies, 2018. Accessed: 2024-07-05.

[110] Alix Plumley. Interview with alix plumley from the cardiff university brain research imaging centre. *Interview*, 2021. Conducted at Cardiff University Brain Research Imaging Centre.

[111] NVIDIA Corporation. *NVIDIA: Company History and Innovation*, 2024. Accessed: 7 April 2025.

[112] NVIDIA Corporation. *CUDA Toolkit Documentation*, 2024. Accessed: 7 April 2025.

[113] NVIDIA Deep Learning Team. *cuDNN: GPU-Accelerated Deep Learning*, 2024. Accessed: 10 February 2025.

[114] Jun Wang, Guanping Xiao, Shuai Zhang, Huashan Lei, Yepang Liu, and Yulei Sui. Compatibility issues in deep learning systems: Problems and opportunities. *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 476–488, 2023.

[115] Subham Chatterjee. Memory hygiene with tensorflow during model training and deployment for inference. `https://medium.com/ai-for-real/memory-hygiene-with-tensorflow-during-model-training-and-deployment-for-i` 2021. Accessed: 2025-05-21.

[116] Omi AI. Why is tensorflow not detecting gpu? `https://www.omi.me/blogs/tensorflow-guides/why-is-tensorflow-not-detecting-gpu`, 2023. Accessed: 2025-05-21.

[117] Google Research. Google colaboratory. `https://colab.research.google.com/`, 2024. Accessed: 2024-07-04.

[118] Odd Erik Gundersen, Saeid Shamsaliei, and Richard Juul Isdahl. Do machine learning platforms provide out-of-the-box reproducibility? *Future Generation Computer Systems*, 126:34–47, 2022.

[119] Harald Semmelrock, Simone Kopeinik, Dieter Theiler, Tony Ross-Hellauer, and Dominik Kowald. Reproducibility in machine learning-driven research. *arXiv preprint arXiv:2307.10320*, 2023.

[120] Tianfeng Chai and Roland R. Draxler. Root mean square error (rmse) or mean absolute error (mae)? arguments against avoiding rmse in the literature. *Geoscientific Model Development*, 7(3):1247–1250, 2014.

[121] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 843–852, 2017.

[122] Monica Welfert, Gowtham R Kurri, Kyle Otstot, and Lalitha Sankar. Addressing gan training instabilities via tunable classification losses. *IEEE Journal on Selected Areas in Information Theory*, 2024.

[123] Emre Kopanoglu. Personal communication, February 2024. Personal communication.

[124] Phil Schmidt. Personal communication, July 2024. Personal communication.

[125] TensorFlow Team. What's new in tensorflow 2.18. `https://blog.tensorflow.org/2024/10/whats-new-in-tensorflow-218.html`, 2024. Accessed: 2025-05-21.

[126] Clova AI Research. Stargan v2: Diverse image synthesis for multiple domains. `https://github.com/clovaai/stargan-v2`, 2020. Accessed: 2024-07-10.

[127] Leandro Ostera. Python dependency hell: A compromise between virtualenv and global dependencies. `https://stackoverflow.com/questions/54475042/python-dependency-hell-a-compromise-between-virtualenv-and-global-depende` 2020. Accessed: 2025-01-28.

[128] AI Stack Exchange Community. Does changing cuda, cudnn, os, etc., versions affect deep learning training results?, 2025. Accessed: 7 April 2025.

[129] Conda Developers. Conda environment .yml files issue #9257, 2025. Accessed: 4 June 2025.

[130] Python Speed. Managing conda dependencies for reproducible environments, 2025. Accessed: 3 April 2025.

[131] NVIDIA Developer Forum. Cuda version affects inference results during batching, 2025. Accessed: 7 April 2025.

[132] Yudong Zhu, Leeor Alon, Cem M. Deniz, Ryan Brown, and Daniel K. Sodickson. System and sar characterization in parallel rf transmission. *Magnetic Resonance in Medicine*, 67:1367–1378, 5 2012.

[133] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[134] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. *Proceedings of the International Conference on Machine Learning*, 2013.

[135] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

[136] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[137] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5 - rmsprop: Divide the gradient by a running average of its recent magnitude, 2012. Coursera: Neural Networks for Machine Learning.

[138] S Gabriel, R W Lau, and C Gabriel. The dielectric properties of biological tissues: I. literature survey. *Physics in Medicine Biology*, 41(11):2231, 1996.

[139] Brain Imaging and Analysis Center. MRI Safety Tutorial. `https://www.biac.duke.edu/research/safety/mri-safety-tutorial`. Accessed: 2025-02-26.

[140] Jennifer Frankel, Jonna Wilén, and Kjell Hansson Mild. Assessing exposures to magnetic resonance imaging's complex mixture of magnetic fields for in vivo, in vitro, and epidemiologic studies of health effects for staff and patients. *Frontiers in Public Health*, 6:66, 2018.

[141] Allen Taflove and Susan C. Hagness. *Computational Electrodynamics: The Finite-Difference Time-Domain Method*. Artech House, Boston, MA, USA, third edition, 2005.

[142] The MathWorks, Inc. *MATLAB R2019a*. The MathWorks, Inc., Natick, Massachusetts, United States, 2019.

[143] Alix Plumley, Nathan Goodwin, and Emre Kopanoglu. Inter-subject differences in sar sensitivity to motion with parallel-transmit at 7t. *Proceedings of the Joint Meeting of ISMRM and ESMRMB. London, UK*, 457, 2022.

[144] Shaihan Malik. In-person meeting with shaihan malik. In-person meeting, 2024. King's College London.

[145] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 1(10):e3, 2016.

[146] Giuseppe Carluccio, Eros Montin, Riccardo Lattanzi, Michalis Latoussakis, and Christopher M. Collins. Deep learning networks trained with eccentric and concentric spherical phantoms for sar prediction, 2022.

[147] National Instruments. Understanding ffts and windowing, 2021. Accessed: Feb 10, 2025.

[148] Fredric J. Harris. On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE*, 66(1):51–83, 1978.

[149] MathWorks. Hann window, 2024. Accessed: 2025-05-28.

[150] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[151] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016.

[152] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[153] François Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1610.02357*, 2017.

[154] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. pages 807–814, 2010.

[155] Yoshua Bengio. Learning deep architectures for ai. *Foundations and Trends® in Machine Learning*, 2(1):1–127, 2012.

[156] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[157] Chiyuan Zhang, Yoshua Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):66–75, 2021.

[158] Lutz Prechelt. Early stopping-but when? *Neural Networks: Tricks of the Trade*, pages 55–69, 1998.

[159] Rich Caruana, Stephen Lawrence, and C Lee Giles. Overfitting in neural networks: Backpropagation, conjugate gradient, and early stopping. *Proceedings of the 13th International Conference on Neural Information Processing Systems*, pages 402–408, 2000.

[160] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 2234–2242, 2016.

[161] Shaihan Malik Emre Kopanoglu Katherine Blanter, Alix Plumley. Estimating variations in sar calculations due to within-scan patient motion using cgans for parallel rf transmission at ultrahigh field mri. May 2024. Presented at: 2024 ISMRM & ISMRT Annual Meeting & Exhibition, 4-9 May, 2024.

[162] suyodhanj6. Exploring pointwise convolution in cnns: Replacing fully connected layers. 2023.

[163] Mark E Ladd, Peter Bachert, Martin Meyerspeer, Ewald Moser, Armin M Nagel, David G Norris, Sebastian Schmitter, Oliver Speck, Sina Straub, and Moritz Zaiss. Pros and cons of ultra-high-field mri/mrs for human

application. *Progress in nuclear magnetic resonance spectroscopy*, 109:1–50, 2018.

[164] Kâmil Uğurbil, Gregor Adriany, Peter Andersen, Wei Chen, Michael Garwood, Rolf Gruetter, Pierre Gil Henry, Seong Gi Kim, Haiying Lieu, Ivan Tkac, Tommy Vaughan, Pierre Francoise Van De Moortele, Essa Yacoub, and Xiao Hong Zhu. Ultrahigh field magnetic resonance imaging and spectroscopy. *Magnetic Resonance Imaging*, 21:1263–1281, 12 2003.

[165] Pierre-François de Moortele, Can Akgun, Gregor Adriany, Steen Moeller, Johannes Ritter, Christopher M Collins, Michael B Smith, J Thomas Vaughan, and Kāmil Uğurbil. B1 destructive interferences and spatial phase patterns at 7 t with a head transceiver array coil. *Magnetic resonance in medicine*, 54:1503–1518, 2005.

[166] Michael Garwood and Lance DelaBarre. The return of the frequency sweep: designing adiabatic pulses for contemporary nmr. *Journal of magnetic resonance*, 153:155–177, 2001.

[167] Toshiharu Nakai, Naoki Kamiya, Michihiko Sone, Hiroyuki Muranaka, Toshio Tsuchihashi, Naoki Yamada, and Sachiko Yamaguchi. Medical electrical equipment-part 2-33 : Particular requirements for the basic safety and essential performance of magnetic resonance equipment for medical diagnosis. *IEC 60601-2-33 Ed. 3.0*, 11:253–264, 2010.

[168] US Food, Drug Administration, et al. Criteria for significant risk investigations of magnetic resonance diagnostic devices-guidance for industry and food and drug administration staff. *Silver Spring, MD: US Food and Drug Administration*, page 2014, 2014.

[169] F D A Food. Drug administration-guidance for industry considering whether an fda-regulated product involves the application of nanotechnology. *Biotechnol Law Rep [Internet]*, 30:613–616, 2014.

[170] Yudong Zhu. Parallel excitation with an array of transmit coils. *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, 51(4):775–784, 2004.

[171] U Katscher, J Röhrs, and P Börnert. Basic considerations on the impact of the coil array on the performance of transmit sense. *Magnetic Resonance Materials in Physics, Biology and Medicine*, 18:81–88, 2005.

[172] Özlem Ipek, Alexander J. Raaijmakers, Jan J. Lagendijk, Peter R. Luijten, and Cornelis A.T. Van Den Berg. Intersubject local sar variation for 7t prostate mr imaging with an eight-channel single-side adapted dipole antenna array. *Magnetic Resonance in Medicine*, 71:1559–1567, 2014.

[173] Zhangwei Wang, James C. Lin, Weihua Mao, Wanzhan Liu, Michael B. Smith, and Christopher M. Collins. Sar and temperature: Simulations and comparison to regulatory limits for mri. *Journal of Magnetic Resonance Imaging*, 26:437–441, 8 2007.

[174] Bob Van Den Bergen, Cornelis A.T. Van Den Berg, Dennis W.J. Klomp, and Jan J.W. Lagendijk. Sar and power implications of different rf shimming strategies in the pelvis for 7t mri. *Journal of Magnetic Resonance Imaging*, 30:194–202, 7 2009.

[175] Ingmar Graesslin, Hanno Homann, Sven Biederer, Peter Börnert, Kay Nehrke, Peter Vernickel, Giel Mens, Paul Harvey, and Ulrich Katscher. A specific absorption rate prediction concept for parallel transmission mr. *Magnetic Resonance in Medicine*, 68:1664–1674, 2012.

[176] Gabriele Eichfelder and Matthias Gebhardt. Local specific absorption rate control for parallel transmission by virtual observation points. *Magnetic Resonance in Medicine*, 66:1468–1476, 2011.

[177] Emre Kopanoglu, Cem M. Deniz, M. Arcan Erturk, and Richard G. Wise. Specific absorption rate implications of within-scan patient head motion for ultra-high field mri. *Magnetic Resonance in Medicine*, 84:2724–2738, 11 2020.

[178] Emre Kopanoglu. Actual patient position versus safety models: Specific absorption rate implications of initial head position for ultrahigh field magnetic resonance imaging. *NMR in Biomedicine*, 5 2022.

[179] Morgane Le Garrec, Vincent Gras, Marie France Hang, Guillaume Ferrand, Michel Luong, and Nicolas Boulant. Probabilistic analysis of the specific absorption rate intersubject variability safety factor in parallel transmission mri. *Magnetic Resonance in Medicine*, 78:1217–1223, 9 2017.

[180] Amer Ajanovic, Joseph V Hajnal, Raphael Tomi-Tricot, and Shaihan Malik. Motion and pose variability of sar estimation with parallel transmission at

7t. *Proceedings of the 29th Annual Meeting of ISMRM. Vancouver, Canada*, 2487, 2021.

[181] TensorFlow Authors. Tensorflow, 2021. Software available from tensorflow.org.

[182] Andrey N. Kolmogorov. Sulla determinazione empirica di una legge di distribuzione. *Giornale dell'Istituto Italiano degli Attuari*, 4:83–91, 1933.

[183] Nikolai V. Smirnov. Table for estimating the goodness of fit of empirical distributions. *Annals of Mathematical Statistics*, 19(2):279–281, 1948.

[184] Sayim Gokyar, Fraser J. L. Robb, Wolfgang Kainz, Akshay Chaudhari, and Simone Angela Winkler. MRSaiFE: An AI-based approach towards the real-time prediction of specific absorption rate. *IEEE Access*, 9:140824–140834, 2021.

[185] U.S. Food and Drug Administration. Assessing the credibility of computational modeling and simulation in medical device submissions. `https://www.fda.gov/media/154985/download`, 2023. Accessed: 2026-02-20.

[186] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[187] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[188] Stack Overflow Community. Resolvepackagenotfound: Create env using conda and yml file on macos, 2024. Accessed: 10 March 2025.

[189] Google Research Team. Issue: Condaenvexception: Pip failed in environment creation, 2024.

[190] QIIME 2 Community. Resolvepackagenotfound: Installing with anaconda prompt on windows, 2024.

[191] Wei Cheng, Xinxin Zhu, and Weipeng Hu. Conflict-aware inference of python compatible runtime environments with domain knowledge graph. *arXiv preprint arXiv:2201.07029*, 2022.

[192] Project Jupyter. Project jupyter, 2024. Accessed: 2025-05-27.

[193] Peter C Fitzpatrick and John R Manning. Davos: a python "smuggler" for constructing lightweight reproducible notebooks. *arXiv preprint arXiv:2211.15445*, 2022.

[194] Anaconda, Inc. *Using pip in a Conda Environment*, 2024.

[195] Stack Overflow Community. Conflict using conda and pip in miniconda environment, 2019. Accessed: 7 April 2025.

[196] TensorFlow Developers. Install tensorflow, 2025. Accessed: 2025-02-20.

[197] Frank Seifert, Gerd Wübbeler, Sven Junge, Bernd Ittermann, and Herbert Rinneberg. Patient safety concept for multichannel transmit coils. *Journal of Magnetic Resonance Imaging*, 26:1315–1321, 11 2007.

[198] Amer Ajanovic, Joseph V Hajnal, and Shaihan Malik. Positional sensitivity of specific absorption rate in head at 7t. *Proceedings of the 28th Annual Meeting of ISMRM*, 4251, 2020.

[199] Katherine Blanter, Alix Plumley, Shaihan Malik, and Emre Kopanoglu. Towards applying deep learning to predict rigid motion-induced changes in q-matrices from uhf-mri ptx simulations. *Proceedings of the 31st Annual Meeting of ISMRM. Toronto, Canada*, 2023.