An Investigation into Explanations for Convolutional Neural Networks

A thesis submitted in partial fulfilment of the requirement for the degree of Doctor of Philosophy

Thomas Hartley

Cardiff University
School of Computer Science and Informatics

March 2021

Abstract

As deep learning techniques have become more prevalent in computer vision, the need to explain these so called 'black boxes' has increased. Indeed, these techniques are now being developed and deployed in such sensitive areas as medical imaging, autonomous vehicles, and security applications. Being able to create reliable explanations of their operations is therefore essential.

For images, a common method for explaining the predictions of a convolutional neural network is to highlight the regions of an input image that are deemed important. Many techniques have been proposed, however these are often constrained to produce an explanation with a certain level of coarseness. Explanations can be created that either score individual pixels, or score large regions of an image as a whole. It is difficult to create an explanation with a level of coarseness that falls in between these two. A potentially even greater problem is that none of these explanation techniques have been designed to explain what happens when a network *fails* to obtain the correct prediction. In these instances, current explanation techniques are not useful.

In this thesis, we propose two novel techniques that are able to efficiently create explanations that are neither too fine or too coarse. The first of these techniques uses superpixels weighted with gradients to create explanations of any desirable coarseness (within computational constraints). We show that we are able to produce explanations in an efficient way that have a higher accuracy than comparable existing methods. In addition, we find that our technique can be used in conjunction with existing techniques such as LIME to improve their accuracy. This is subsequently shown to generalise well for use in networks that use video as an input.

The second of these techniques is to create multiple explanations using a rescaled input image to allow for finer features to be found. We show this performs much better than comparable techniques in both accuracy and weak-localisation metrics. With this technique, we also show that a common metric, faithfulness, is a flawed metric, and recommend its use be discontinued.

Finally, we propose a third novel technique to address the issue of explaining failure using the concepts of surprise and expectation. By

building an understanding of how a model has learnt to represent the training data, we can begin to explore the reasons for failure. Using this technique, we show that we can highlight regions in the image that have caused failure, explore features that may be missing from a misclassified image, and provide an insightful method to explore an unseen portion of a dataset.

Acknowledgments

I would like to begin by thanking my supervisors Professor David Marshall, Dr Kirill Sidorov, and Dr Christopher Willis. Thank you for all your encouragement, and guidance. Your support has been invaluable, and without all of your insights I would not have progressed as much as I have.

Of course, none of this would have been possible without the sponsorship of both BAE Systems and the EPSRC. For this opportunity and support I am massively grateful.

A big thank you to my fellow PhDs in S/0.45. In particular, my thanks go to Aled for our chats on papers, food and code. I think all three were very important to the completion of this Thesis.

To Dan, thank you for your friendship. You were with me at the start of my computer science journey and, although you may not have realised it, our evenings of boardgames have been a welcome moment of calmness in the storm.

To my sister, Emy. Thank you for being such a part of all of this. Our lives have been intertwined since I was 16 months old, and I am so glad we have managed to stay close to each other. And remember, I love you by default.

To my parents, William and Gwen, thank you for your unwavering support through all the years and career changes. Who would have thought that I would go from following you both into the world of TV, to almost having a PhD. Talk about the black sheep of the family!

I would like to thank my wife. Sharan, your understanding and encouragement have allowed me to push through with this. I could not have done any of this without you. We've made sacrifices to get here, but hopefully it will all start to pay off.

Finally, I would like to thank my daughters, Iona and Ada. Although neither of you have a clue what I am doing, and your erratic sleep patterns have introduced me to the wonders of coffee, your love and laughter have given me an extra boost to pursue this PhD.

Contents

List of Figures

List of Tables

CHAPTER

1 Introduction

Convolutional Neural Networks (CNNs) are a technique, based on deep learning, that allows data with a grid-like structure [1] (typically images) to be processed. CNNs can be used for a wide range of applications. In this thesis, we will focus on the task of classification. Classification tasks are where the CNN learns to represent features from the input, and combine them in such a way that they can predict the class of an object present within the input.

CNNs were first introduced in 1989 with LeNet5 [2]. However, their recent popularity only began in 2014 with the introduction of AlexNet [3]. Since this point there has been an explosion in the prevalence of CNNs, touching all areas of computer vision. A striking example of this is that in a 2018 press release [4], Facebook reported that Caffe2 (their production deep learning framework) was responsible for more than *200 trillion* predictions per day. Whilst only a subset of these predictions will be using CNNs, even a fraction of this figure would still be a huge number of predictions. Indeed, such is the prevalence of CNNs, they have found themselves at the heart of many complex or critical applications [5–8].

A by-product of how a CNN is able to predict a class successfully is that the CNN itself, learns how to represent the features of an image. In pre-deep learning techniques, feature engineering was typically used. This is where the features in an input are explicitly specified through the use of domain knowledge. A toy example of this is that an engineer could specify a value for

how pointy an ear is to predict if an image is of a cat or a dog. However, with a CNN these features are learnt by the network, and therefore, known only to the network. This is problematic when a designer or end user of a CNN would like to know *why* a prediction has been made. This has led CNNs to be referred to as black boxes [9–11].

Numerous examples [12–15] have shown that CNNs can perform well at a classification task by using features from within an image that would be unacceptable to a human. For example, using gender to classify between nurses and doctors [15], using properties of the camera that took an image to determine patch location [12], and learning the image background rather than the object to be classified [13, 14]. All of these issues would potentially have gone unnoticed had there not been a way to visually inspect the reasons behind a network's prediction. If an explanation can be made of the network's reasoning, then that reasoning can be analysed by the developer, user of the system, or external auditor.

However, the concept of an explanation remains vaguely defined [5]. Robnik-Šikonja and Bohanec [16] consider an explanation in this context to be a way to capture the relationship between inputs and outputs of a model. The form that this can take also varies depending on the approach taken. A common way of creating explanations that we will concentrate on throughout this thesis is to generate a heatmap that assigns values to regions or pixels based on how important they are to the output of the network. However, even within this subcategory of explanations there are multiple approaches, each with their own advantages and limitations. One of the core differences in these approaches, is the granularity of the explanation. As we will investigate in this thesis, there is often a trade off between an explanation being too fine so as to not be easily understandable, or too coarse so as to not be able to precisely represent how the model has made its prediction. Alternatively, methods that find a middle ground between fine and coarse explanations can be found. However, these come at a computational disadvantage as they rely on perturbation techniques [9, 13, 17] which often require thousands of passes through a network to create a single explanation.

To allow a better understanding of this crucial concept, which is used throughout the thesis, we show examples of fine, medium and coarse grained explanations in Figure 1.1. No set definition about what constitutes a certain

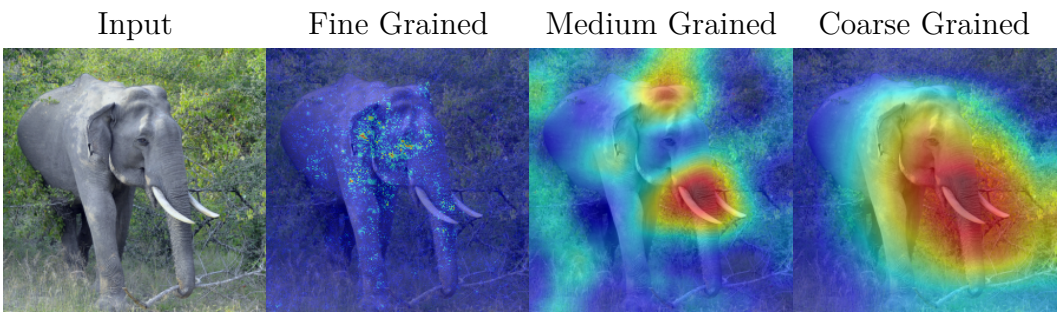

FIGURE 1.1: Examples of differing levels of explanation granularity. The medium grained explanation shown here required 8,000 passes through the network to create.

granularity of explanation exists. There is not a discrete range of explanation granularity levels, but rather it is open to user interpretation. However, as we see in the figure, there are clear indicators of where an explanation may be on this scale, with fine grained explanation having a much more complex and complicated explanations compared to a coarse one. A useful framework to have for this is the concept of soundness and completeness of an explanation, introduced by Kulesza *et al.* [18] and discussed further by Miller [19]. The core of this concept is summed up by Kulesza *et al.* as: a users time and interest is finite, and an explanation may not be better simply because there is more information. Instead they propose two dimensions of an explanation; soundness (how truthful or accurate the explanation is), and completeness (how much of the underlying system is explained by the explanation). Miller suggests the three principles to take from Kulesza's work is that an explanation must be sound, must be complete, and must not overwhelm the user. He highlights that the principle of not overwhelming the user is at odds with the first two. Here we see the crux of how these concepts relate to the idea of explanation granularity. A fine grained explanation may be sound and complete, but assigning a score to every individual pixel is overwhelming. By reducing the granularity, we reduce the soundness and completeness, but also reduce how much the user has to understand to appreciate the explanation.

What is needed are methods that can bridge this fine/coarse gap which are achievable in a computationally efficient way. Doing so would allow us to create explanations which are able to accurately locate the regions that are used to inform a model's prediction, whilst also being interpretable to the viewer of the explanation.

This is not to say that the only method for understanding a models prediction is to generate a heatmap. Indeed, there are other methods that we will discuss, such as finding dataset samples that are indicative of how a network has learnt to represent the dataset classes [20]. Alternatively, an explanation can be created for individual components within a network, depending on how they have learned to represent the training data [21]. A recent set of techniques has also begun to incorporate specific domain knowledge into explanations to allow for semantically meaningful regions to be labelled and explained, i.e body parts of a bird [22].

In this thesis, we will primarily focus on image classification tasks, and as such the majority of explanation techniques that we will examine are created to understand why a model has predicted a class in a given way. However, these techniques are almost exclusively used to explain *correct* predictions, where the model has accurately classified an image. There has been comparatively little research in the area of explaining why a network has *failed* to predict the correct class. Arguably this is an area of research that is of equal importance to explaining correct classifications, for both developers of deep learning solutions and end users alike.

There are a number of methods that have been introduced to measure certain aspects of an explanation once it has been created. These range from measuring how well the explanation can locate regions important to the classification of the object (accuracy) to how well the explanation can locate the object being classified (weak-localisation). We will use a range of these metrics throughout this thesis and importantly will also present baselines for use with these metrics. Indeed, it is notable that a number of metrics are introduced alongside explanations, with no measurements done on baselines to assure us that they are actually performing as expected [23, 24].

1.1 Motivation

The primary motivation behind this thesis comes from the impact that deep learning has had on all fields of computer vision, particularly applications which require a robust understanding of why they perform the way they do. Examples of these are medical diagnostics [6, 25, 26], surveillance and security applications [7, 27], or autonomous vehicles [8, 28]. In these applications, the

predictions made by deep learning systems are required to be trustworthy. In this context, being trustworthy is not simply about achieving a high accuracy when training the models, but being able to understand why a model has made a specific prediction. For example, a doctor using a deep learning based method for breast cancer prediction may like to see which regions of an X-ray the model has used to inform the classification.

Since the ascendancy of deep learning techniques in computer vision, attempts have been made to create either interpretable models, or post-hoc explanations [29]. In this thesis, we will investigate post-hoc explanations as it is still much more common to use non-interpretable models for computer vision. Post-hoc explanations are those which are created after the model has been trained for the desired task. There are numerous ways to create post-hoc explanations, from using a single input image as a vehicle for creating an explanation of the models prediction [15,30–34], to using an entire dataset to explain the inner workings of the model [21,35]. In particular, input centric methods have had a large amount of research dedicated to them, as the highlighting of regions within an input image is a popular way of creating explanations. However, in this area of explanations in particular, there is often a trade-off made between how interpretable an explanation is to a human, and how well the explanation represents the regions of the input image used by the model to make its prediction. This is often unavoidable, due to the methods used to obtain the explanation. For example, methods which backpropagate back to the input image give a relevance score to each pixel, whilst those techniques which use activation maps extracted from lower convolutional layers are left with a large contiguous region being assigned a single relevance score. Finding a method that can bridge this divide efficiently would be beneficial to the community, as it would allow a granularity of explanation that was previously the reserve of perturbation methods to be created quickly. This in turn would allow a quicker and more accurate understanding of the models being used at the time.

The majority of explanation techniques in computer vision are aimed at image based deep learning techniques. However, CNNs are also used heavily in the action recognition domain. Previous attempts have been made to take existing explanation methods and apply them to the video domain [23, 36]. However, few techniques are developed specifically for use with video inputs.

This presents an interesting problem as action recognition networks are often crucially different to image based networks, from the inclusion of a temporal element using 3D convolutions [37,38], to the use of two-stream networks [39,40]. Any technique that is developed for action recognition networks should take into account these properties in order to produce explanations, and while these do exist [23, 36, 41, 42], they are poorly represented when compared to those available for image based networks.

Of importance to note at this point in the thesis is that explanation techniques can be developed with a range of stakeholders in mind. For example in the work by Preece *et al.* [43], 4 stakeholder communities are identified (developers, theorists, ethicists, and users), of which the target audience for the work contained in this thesis is developers. This is not to say that members from other communities may not find aspects of this work useful, simply that they are not the intended audience.

This thesis explores how post-hoc explanations can be created for CNNs and their use for both image and video based networks. In particular, we show how explanations can be created for image and video networks that are neither too fine or too coarse in their detail. We also present an approach based on explaining why a network has *failed* to classify an input correctly.

1.2 Research Goals

In this section, we outline the research goals, driven by a desire to be able to better explain CNNs, that guide this thesis:

Research Goal 1: We hypothesise that there is an efficient way of creating medium-grained region based technique without the use of expensive perturbations.

Research Goal 2: We hypothesise our medium-grained explanation technique can create explanations for networks using video as an input.

Research Goal 3: We hypothesise that creating explanations from multiple regions of a single image can be combined to produce an effective medium-grained explanation.

Research Goal 4: We hypothesise that by understanding how the individual filters in a network react to an image and compare it to the activations found in the training data, we can begin to determine the reasons that networks fail to classify inputs correctly.

1.3 Contributions

In this section, we outline the contributions that we have made and how they link to the hypotheses,chapters and, where applicable, papers produced. In this thesis, we make the following contributions:

Contribution 1: We propose a method for combining superpixel and gradients: SWAG (Superpixels Weighted by Average Gradients). We show qualitatively and quantitatively that this method is able to produce explanations that perform well across a number of metrics. This contribution addresses research goal 1 and can be found in Chapter 3. This work has been presented in the following papers:

[44] Thomas Hartley, Kirill Sidorov, Christopher Willis, and David Marshall. Gradient Weighted Superpixels for Interpretability in CNNs. Workshop on Interpretable and Explainable Machine Vision. *BMVC 2019*.

[45] Thomas Hartley, Kirill Sidorov, Christopher Willis, and David Marshall. SWAG: Superpixels Weighted by Average Gradients for Explanation in CNNs. *WACV 2021*.

Contribution 2: We propose a modification of a superpixel creation technique called Simple Linear Iterative Clustering (SLIC) that allows us to use gradient-based explanations to inform the superpixel creation process (N.B we do not train the superpixels to find regions as you might with attention maps). We show that using these superpixels gives improved explanations. This contribution addresses research goal 1 and can be found in Chapter 3. This work has been presented in the following paper:

[45] Thomas Hartley, Kirill Sidorov, Christopher Willis, and David Marshall. SWAG: Superpixels Weighted by Average Gradients for Explanation in CNNs. *WACV 2021*.

Contribution 3: We propose SWAG-V, an extension of SWAG that works with video inputs. We again show that this performs well across a number of metrics for use with action recognition networks. This contribution addresses research goal 2 and can be found in Chapter 4. This work is currently under submission to WACV 2022.

Contribution 4: We propose Jitter-CAM, a method that creates multiple Class Activation Maps (CAM) explanations and combines them to form a single explanation. This contribution addresses research goal 3 and can be found in Chapter 5. This work is currently under submission to BMVC.

Contribution 5: We show that a commonly used metric in the CAM literature is fundamentally flawed and its use should be discontinued. This contribution addresses research goal 3 and can be found in Chapter 5. This work is currently under submission to BMVC.

Contribution 6: We propose a method for scoring how an individual filter reacts to an image. We show that this method performs better than a number of alternative methods. This contribution addresses research goal 4 and can be found in Chapter 6. This work has been presented in the following paper:

[46] Thomas Hartley, Kirill Sidorov, Christopher Willis, and David Marshall. Explaining Failure: Investigation of Surprise and Expectation in CNNs. Workshop on Fair, Data-Efficient and Trusted Computer Vision. *CVPR 2020*.

Contribution 7: Using our method of filter scoring, we propose a method for building distributions how filter react to training data and then comparing unseen images to this data. This contribution addresses research goal 4 and can be found in Chapter 6. This work has been presented in the following paper:

[46] Thomas Hartley, Kirill Sidorov, Christopher Willis, and David Marshall. Explaining Failure: Investigation of Surprise and Expectation in CNNs. Workshop on Fair, Data-Efficient and Trusted Computer Vision. *CVPR 2020*.

Contribution 8: We propose a method for locating regions of an image that are causing an image to be misclassified through the use of the above distributions. This contribution addresses research goal 4 and can be found in Chapter 6. This work has been presented in the following paper:

[46] Thomas Hartley, Kirill Sidorov, Christopher Willis, and David Marshall. Explaining Failure: Investigation of Surprise and Expectation in CNNs. Workshop on Fair, Data-Efficient and Trusted Computer Vision. *CVPR 2020*.

1.4 Thesis Structure

The remainder of this thesis is as follows:

Chapter 2 introduces the reader to the common concepts, techniques, and datasets discussed throughout this thesis.

Chapter 3 describes a novel algorithm for creating explanations in an efficient way that finds a trade-off between accuracy and interpretability.

Chapter 4 extends the novel algorithm from Chapter 3 and explores how it can be applied to CNNs designed for action recognition.

Chapter 5 proposes a method for creating a CAM based explanation that performs better than previous CAM explanations.

Chapter 6 proposes a novel method for generating explanations that proposes why a network fails in it's predictions. Using the techniques from chapter 3 we are able to highlight potential reasons for failure within an input image.

Chapter 7 concludes the thesis and summarises the findings and contributions made throughout.

CHAPTER

2 Background

2.1 Introduction

As deep learning networks have become more commonplace, the creation and examination of explanation techniques has become more rigorous. This chapter aims to provide an overview of both interpretability techniques and the methods for measuring how well they perform.

2.2 Interpretability Techniques

In this section, we explore the body of work that has built up around explaining the reasons behind a networks prediction. As deep learning as a research topic has exploded in the last decade, the amount of techniques for explaining the networks inner workings has also increased. A common description of interpretability is "the degree to which an observer can understand the cause of a decision". [19] It is also common to use the terms interpretability and explainability interchangeably [19, 29]. However, the use of the word 'explanation' is often reserved for discussing the understanding of a single prediction. As we will outline in this chapter, there are many methods available to allow an understanding of a models decision. We will begin by briefly discussing two differing types of interpretability: inherently-interpretable models vs post-hoc explanations.

2.2.1 Inherently Interpretable Models vs Post-hoc Explanations

Interpretability methods can be split many different ways into many different categories and sub-categories. At the highest level, the majority of interpretability methods can be split into two groups, these are: inherently-interpretable models and post-hoc explanations. Inherently-interpretable models are models that are able to produce an explanation of a prediction alongside the prediction itself. A basic example of this is a decision tree [47], whereby a clear path can be followed to understand how a prediction has been made. However, the majority of CNN architectures are black boxes, and while some networks have been designed to be inherently-interpretable (e.g. [48]) they remain rare compared to the non-interpretable models. One prominent reason for this is that inherently-interpretable models are still at an early stage of their development and currently are unable to compete with more established, non-interpretable models [48–50].

Post-hoc methods accept that the model is a black box and attempts to create explanations based on this. For CNNs, these post-hoc methods can be broadly split into two sub-categories depending on whether we are trying to understand the model as a whole, or a single prediction. Varying names have been given to this divide in techniques, for example Du *et al.* [51] call them post-hoc global explanations and post-hoc local explanations respectively while Murdoch *et al.* [52] define them as dataset-level and prediction-level interpretations. For the purpose of this thesis and in order to avoid confusion with a later discussed concept called global accuracy, we will refer to this split in methods as network-centric and input-centric explanation techniques. However the two sub-categories are referred by different authors, the divide in concepts remains the same. Input-centric (post-hoc local explanations/prediction-level interpretations) aim to explain a model's prediction. For CNNs, this is typically through the use of a single image. The subsequent explanation is then combined with the input image, giving a visual representation of the importance of the features in the image to the model. Network-centric (post-hoc global explanations/dataset-level) techniques aim to understand how the model has learnt to represent the training data. This is typically done by passing a dataset through the network and observing how some element of the network reacts. This allows explanations to be generated that seek to show the learnt representations within the model.

In this thesis, we solely explore post-hoc methods for the generation of explanations. We base this decision on how prevalent the current generation of deep learning based techniques have become. As such it is likely that these 'black-box' architectures are going to be in common use for some time throughout industry, in which case their lack of interpretability will still need addressing. For example, models currently deployed for use in a medical setting will have been through a rigorous approval process and there will only be a gradual shift from current models to newer ones [53]. The following subsections will introduce and discuss both input- and network-centric post-hoc explanation techniques.

2.2.2 Input-Centric Explanations

In this section, we discuss input-centric methods for generating explanations. We break this section down into three subsections, these are:

- Gradient based methods.
- Activation based methods.
- Perturbation based methods.

Gradient based methods [30, 31, 33, 54–56] are those that rely on back propagation through the network to the input image to highlight areas of interest. Activation based methods [15, 23, 24, 34] are those methods that use the activation maps produced by the layers of the network as a basis for visualisation. Finally, perturbation methods [9, 11, 13, 17, 57–59] are those that treat the network as a black box and pass multiple modified version of an image to see how the network reacts. These reactions then inform the creation of the explanation.

2.2.2.1 Gradient Based Methods

Gradient based methods are a selection of techniques that create explanations by backpropagating through the network back to the input space. This class of explanations has proven to be enduringly popular [60] and have been consistently built upon over the years. Examples of gradient based methods are shown in Figure 2.1. In this figure, we note that fine-grained explanations are created using gradient-based methods since every pixel is assigned a score.

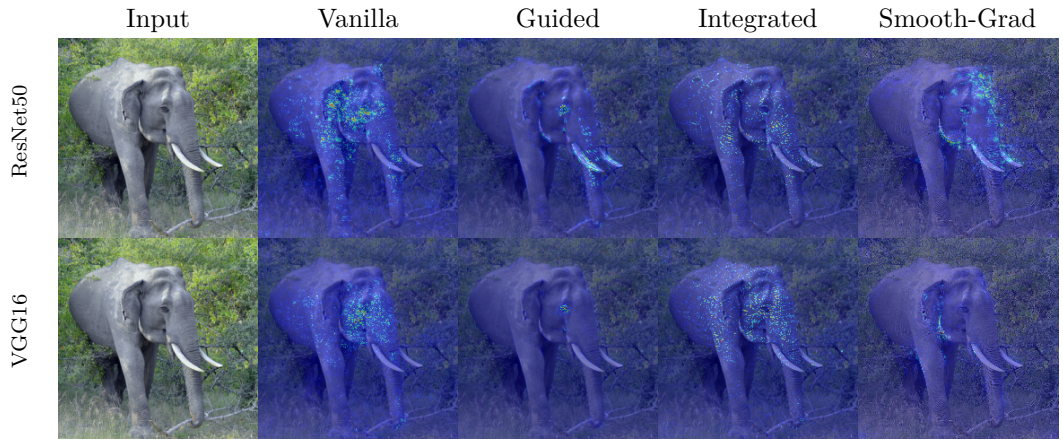

FIGURE 2.1: Examples of gradient-based methods. Of importance to note is how fine grained these explanations are. Every pixel is assigned a score.

The idea of backpropagation through a CNN as an explanation technique was proposed by Simonyan *et al.* [30] based on early, non-CNN based works [54]. They referred to their technique as class saliency extraction but it is now commonly referred to as simply 'vanilla gradients'. The idea behind this technique is simple, for a given image, it is desirable to have an importance score for each pixel showing how important it it to the network. This is achieved by backpropagating from a target classes pre-softmax score to the input image. This results in a set of gradients the same size as the input image. To reduce this down to a single score per pixel, the maximum is taken for each gradient score over the corresponding RGB channels. This produces an explanation that the authors show to be able to localise and segment the target object successfully.

A technique proposed by Zeiler and Fergus [17] called deconvolution aims to invert the network so that the input can be reconstructed based on the output of a network. It does this using by attaching a deconvnet [61] (essentially a network where all layers perform the inverse of those in the original CNN) to the CNN under examination.

Springenberg *et al.* [31] highlight the fact that previous methods (vanilla gradients and deconvolution) fail to produce sharp recognisable image structure. They suggest that this is primarily down to the way in which the signal is backpropagated. Both vanilla gradients and deconvolutions allow negative gradients to flow backwards through the network. Springenberg *et al.* introduce

guided backpropagation as a means of fixing this. Guided backpropagation is a method almost identical to vanilla gradients, but it uses the output of the ReLU activations from the forward pass to help determine which gradients to backpropagate. By only backpropagating gradients at positions that are positive for both the backward pass and in the corresponding activation map, a much clearer explanation is generated.

Layer-Wise Relevance Propagation (LRP), proposed by Bach *et al.* [55], is another backpropagation based technique that aims to identify the pixels important to the network's prediction. However, unlike deconvolution [17] and vanilla gradients [30], they argue that LRP aims to reconstruct the classifier decision, rather than the input. They achieved this by taking into account the signed activations from the layer preceding the one they are currently weighting. This has the effect of only redistributing values to the lower layer, based on the values received from the previous layer. The relevancies of a particular neuron are computed using the product of the activation and the weight between the neurons that produced the activation. A complication of LRP that there is not a single method for computing relevance, and a number of rules have been introduced depending on where in a network the relevance is being computed. LRP was refined further by Montavon *et al.* [62] as Deep Taylor Decomposition. The authors suggest that the rules from LRP can be seen as a series of Taylor expansions. They propose new rules that redistribute relevance scores to neurons through the use of a first-order Taylor expansion around a reference point.

Shrikumar *et al.* [32] highlight that explanations such as guided backpropagation remove negative gradients during the backward pass. They argue that removal of these, results in an explanation not being able to identify regions of an input that contribute negatively to a networks output. They also highlight an issue they call model saturation, whereby a models output will not change even if a region of potential importance is removed while another identical feature is not. For example, given a network that finds the colour yellow, and the input is two yellow pixels. Both are equally important, but if one is set to black the network will still determine yellow is present so a gradient of 0 will be assigned to the turned off yellow pixel at backpropagation. Shrikumar *et al.* [32, 63] address these problems through the introduction of Deep Learning Important FeaTures (DeepLIFT). This technique tries to explain the difference

in output from a network between a reference image and the input. For images, the authors recommend either a black image (all zeros) or a blurred version of the input. They assign a contribution score for each pixel based on the difference from the reference. Crucially, even when the gradients are zero, the contribution score can be non-zero, this helps to alleviate the model saturation problem.

When determining what properties a good explanation should have, Sundararajan *et al.* [33] proposed two axioms, and a novel method called integrated gradients that satisfies them. The two axioms they identify are sensitivity and implementation invariance. The sensitivity axiom says that given two images that vary in *only* one feature, if the prediction of a network is different for both then that feature must be given a non-zero score. The implementation invariance axiom states that, given two networks that are functionally equivalent (they produce the same output but have differing implementations), the explanation should be identical for both. The arguments for implementation invariance is that if an explanation method fails to satisfy it, then it means the explanation could be sensitive to unimportant aspects of the network. Sundararajan *et al.* [33] found that none of the existing explanation techniques at the time satisfied both axioms. To solve this, they introduced Integrated Gradients, a method that takes an input image and a reference image as Shrikumar *et al.* [32] (they call it a baseline image) and multiple images that transition from one to the other. By generating the gradients for each image and integrating them, an explanation is created that the authors find better reflects distinctive features of an image.

Smilkov *et al.* [56] took these techniques and observed that they are all often visually noisy. To alleviate this and reduce the visual noise in an explanation, they introduced a technique called Smooth-Grad. The authors propose that rather than creating an explanation using the gradient directly, the gradient could be smoothed using a Gaussian kernel. They propose to do this by adding Gaussian noise to multiple copies of the input image, calculating their gradients, and then taking the mean of these gradients. This produces an explanation that has significantly reduced amounts of visual noise.

Excitation Backprop was introduced by Zhang *et al.* [64] using a probabilistic winner take all method. Using this technique they define any connection between neurons with a non-negative weight as excitatory. This is determined

during the forward pass through the network. Excitation probabilities are computed in a backward pass using only the excitatory connections. A combination of the activation map and the weights used to generate it on the forward pass is used to compute the probability.

2.2.2.2 Improvements to Gradient Based Methods

A number of techniques have used one of the above gradient based techniques as a basis for creating a novel explanation. This section describes three techniques that have been introduced that can work alongside the aforementioned gradient based methods to help improve them.

As a component of their DeepLift technique, Shrikumar *et al.* [32, 63] suggested that LRP [55] can be reduced to simply the gradient multiplied by the input. This has the benefit of being much simpler to compute than LRP but can also be applied to differing explanation techniques (i.e. guided backpropagation). This technique is referred to as gradient×input. While it does produce visually less noisy explanations, a concern is that black pixels with a value of 0 will never have any attribution applied to them [56].

XRAI is a method introduced by Kapishnikov *et al.* [65] which uses superpixels as the foundation for creating explanations. XRAI uses superpixels to create interpretable regions within the image. Multiple sets of superpixels are generated using differing parameters. Each superpixel in every one of the superpixel sets is assigned a score. This score is generated through the use of integrated gradients [33]. This is achieved by generating an explanation for a given input image using the sum of both black and white baselines for integrated gradients. This produces a single score for each pixel in the image. For each superpixel, a value g_s is assigned where:

$$g_s = \sum_{i \in s/M} \frac{A_i}{area(s/M)}. \tag{2.1}$$

Here, A is he explanation map generated using integrated gradients, s is the current superpixel region and M is the explanation we are creating. The blank explanation is built up in multiple passes by adding the regions that give the highest values per area. This technique scores well in metrics that will be introduced later in this thesis. However, it is computationally inefficient as it requires multiple iterations of the algorithm to build the explanation.

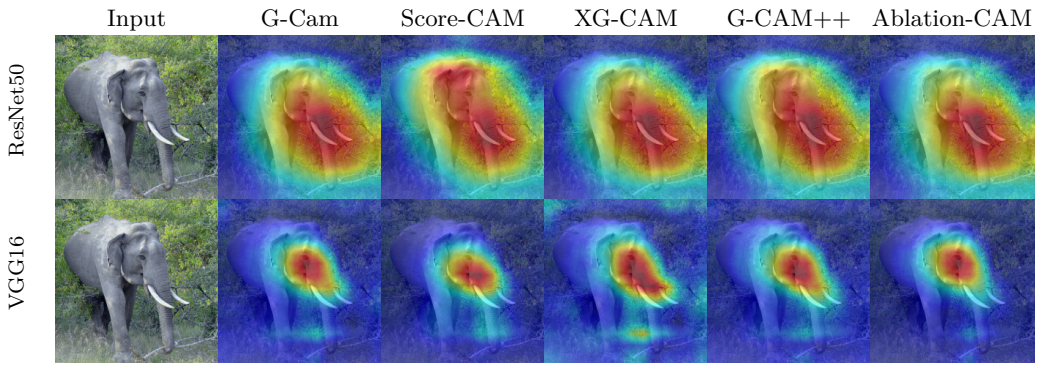

FIGURE 2.2: Examples of different methods for generating Class Activation Mappings (CAM). Note how coarse they are, and how little visual difference there is between methods.

2.2.2.3 Activation Based Methods

An alternative approach to understanding how a network is making a prediction for a given image is to visualise the activation maps produced by individual filters. In many respects this area of input-centric explanations uses similar techniques to those of network-centric explanations, as we are showing how the filters themselves are reacting to an image. However, as this strand of work matured, a distinct input-centric approach has developed using these activations so we will discuss them here.

The simplest method of using activations to create an explanation is to simply input an image into a network, then view the individual activations. An example of this was proposed by Yosinski *et al.* [66] who developed an application that would allow them to capture web-cam footage and view the activations of the network in real time. Using this, Yosinski *et al.* were able to demonstrate that valuable intuitions about how the network was interpreting the image could be gained. For example, they found that a single filter could recognise faces across a range of species, or that a filter learnt to recognise text.

However, there is an element of information overload in trying to view even just the activation maps from the final layer of a CNN. For example, in the final convolution layers of the VGG16 [67] and ResNet-152 [68] architectures there will be 512 and 2,046 activation maps produced respectively. A novel way of reducing these into a single heatmap was proposed by Zhou *et al.* [34] called Class Activation Mapping (CAM).

CAM is a technique that allows a user to discover regions in an image that are discriminative to the network. This is achieved by summing all of the activation maps from the final convolution layer into a single unified heatmap. However, simply combining the activations would be misleading, as often filters that are activated, are not ultimately useful for the models predictions. Instead, a way of weighting the individual activation maps is required. This is achieved through the use of a Global Average Pooling (GAP) [69] layer within a network. A GAP layer has the effect of taking the spatial average from each activation map in the preceding layer. This results in a single value for each activation map which represents a score of the importance. The higher the spatial average of the activation map, the more important it becomes to the discriminative abilities of the network. These values can be then used to weight the activation maps by simply taking the product of each activation map and its corresponding spatial value. The heatmap is then created by summing these weighted activation maps together. As this heat map is the size of the final layer activation map and typically small (e.g. 7×7 for ResNet), it is resized to the same size as the input. Bi-linear interpolation is used during resizing.

While the heatmaps produced by CAM are useful for visualising the regions of an image that are useful to the networks prediction, a major drawback was the limitation of having to use the values from the GAP layer. This limits the use of this technique as it requires the GAP layer to be present during training as it sits between the final convolution layer and the softmax layer. This means any network architecture must either already contain this configuration, or be retrained with it present. Selvaraju *et al.* [15] proposed a generalisation of CAM based on using the gradients instead of a GAP layer. This approach, titled Gradient-weighted Class Activation Mapping (Grad-CAM), uses the the gradients to generate the weights for the activation map rather than the output from a GAP layer. These gradients are generated using the vanilla gradient method introduced by Simonyan *et al.* [30] and discussed previously. Rather than backpropagating back to the input image, the gradients are taken from the convolution layer. To obtain a single weight value from each set of gradients, the spatial average is taken. All subsequent CAM based techniques follow a similar pattern, aiming to create a set of values which are used to weight the activations. As such, we now introduce the notation that we will use to discuss CAMs throughout this thesis. With Grad-CAM we have seen

that weights are generated that corespond to the filters of the layer in a CNN. We show this as α_k^c, which is the weight of filter k for class c. The activation map of the final layer from filter k which are used as the basis of the CAMs are shown as A_k. Z is number of individual values in an activation map. The prediction score for class c is labelled as y. Using this notation, we generate the weights for Grad-CAM as so:

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k}. \tag{2.2}$$

The product of each individual activation map and corresponding weight is then summed together to give a single heatmap. All values below 0 are set to 0 through the use of a Rectified Linear Unit (ReLU):

$$\text{CAM} = ReLU\Big(\sum_k \alpha_k^c A_k\Big). \tag{2.3}$$

This heatmap is then resized to the input image size using bi-linear interpolation. All the following CAM methods use this approach to combine their generated weights with the activations.

Also proposed alongside Grad-CAM was Guided Grad-CAM [15]. This is the element-wise product of Grad-CAM and Guided Backpropagation which produced an explanation that is still as fine as Guided Backpropagation, but better localises to the object of the class being explained.

CAM methods have proven to be enduringly popular, with numerous novel approaches for how to combine the activation maps being introduced. Examples of these CAM methods are shown in Figure 2.2.

Following the introduction of Grad-CAM, a number of approaches have been introduced, all of which attempt to improve CAMs by altering the method for the creation of the weights used. Chattopadhyay *et al.* [23] suggested that, while Grad-CAM was a useful technique, it was deficient in two areas. The first is that while Grad-CAM is able to locate the region of the image important to the model, Chattopadhyay *et al.* suggest that a good explanation should "capture the entire object in completeness". Secondly, they suggest that an explanation technique based on activation maps should also work for action recognition networks. Based on these areas, they introduced Grad-CAM++. This follows the same approach as Grad-CAM, but reformulates how the weights are created from the gradients. Grad-CAM++ aims to only include

positive gradients (again through the use of a ReLU function) when creating the weightings, (similar to deconvolution [17] and guided backpropagation [31]). More concretely, the Grad-CAM++ weights are created as so:

$$\alpha_k^c = \sum_i \sum_j \Big[\frac{\frac{\partial^2 y^c}{(\partial A_{ij}^k)^2}}{2\frac{\partial^2 y^c}{(\partial A_{ij}^k)^2} + \sum_i \sum_j A_{ij}^k \{\frac{\partial^3 y^c}{(\partial A_{ij}^k)^3}\}} \Big] . ReLU\Big(\frac{\partial y^c}{\partial A_{ij}^k}\Big). \tag{2.4}$$

Here i and j are locations of elements within A. They find that doing so produces explanations that highlights more of the object present in the image when compared to Grad-CAM. Of particular interest to this thesis is their attempts to produced explanations for action recognition networks. Their approach is no different to how they would treat an image classification network, except using an architecture specifically designed for using video as an input - the C3D Network [37]. Explanation techniques relating specifically to video will be discussed further in Section 2.3.

Axiom-based Grad-CAM (XGrad-CAM) [70] was introduced as a way of getting CAM based methods to comply with the axioms introduced by Montavon *et al.* [71] and Sundararajan *et al.* [33]. In particular XGrad-CAM, is intended to comply with the axioms of sensitivity [33] and conservation [71]. They achieve this in a similar way to Grad-CAM++ by reworking how the gradients and activation maps are combined to produce the weights:

$$\alpha_k^c = \sum_i \sum_j \Big(\frac{A_{ij}^k}{\sum_i \sum_j A_{ij}^k} \frac{\partial^2 y^c}{\partial A_{ij}^k} \Big). \tag{2.5}$$

Ablation-CAM, introduced by Desai and Ramaswamy [24], removes the gradient component from the weight creation process. Instead they set each of the activation maps from the final convolution layer to zero, one at a time, and observe the effect that removing them has on the networks prediction. The weights are therefore generated as so:

$$\alpha_k^c = \frac{y^c - y_k^c}{y_c}. \tag{2.6}$$

Here y_k^c is the prediction score y for class c when activation map at index k is set to 0. The intuition, here, is that removing an activation map that is important to the networks prediction, will cause a corresponding drop in the *pre-softmax* class scores. These scores are used as the weights by subtracting

them from the pre-softmax class scores and dividing by the pre-softmax class scores. Any activation map that has caused a large drop in the prediction score will now produce a high value showing how important it is to the network.

Following on from this idea of no longer using the gradient to inform the weights, Score-Cam by Wang *et al.* [72] uses the product of the activation maps and the input image to inform the weights. The activation maps from the final layer are extracted, normalised between $[0, 1]$, and then rescaled to the input size. This set of activation masks is then multiplied with the input image creating a new set of input images masked to represent their corresponding activation map. These are then passed to the network and the softmax score for each activation map/image combination stored. These softmax scores are then used as the weights for each activation. They are combined and summed to create the heatmap as before.

Rebuffi *et al.* [73] proposed NormGrad, a method with similarities to Grad-CAM. However, rather than try to explain the reasons behind the output of a model, NormGrad aims to find locations within an image that are useful for training. As with the previous techniques, the gradients and activation maps are combined and the Frobenius norm taken (hence NormGrad). A key difference between Grad-CAM and NormGrad is that GradCam only takes positively aligned activations and gradients. NormGrad on the other hand allows both positive and negative features so as to show regions detrimental to training.

A method separate from CAM work that uses the activation maps from a separate network was introduced by Dabkowski and Gal [10]. The core of the idea is to train a model (separate to the base model which is treated as a black box) to produce an explanation. This is done using an adapted U-Net [74] architecture. A ResNet50 model (pretrained using ImageNet) is used for the contracting path with a series of upsamplers being used for the expansion path. From each of the 5 blocks present within ResNet50, the activations are passed to their corresponding upsampler. A custom loss function is then used which aims to encourage 4 desirable explanation aspects. These are:

- Mask smoothness.
- Compact explanations.
- Ensuring the salient explanation region gives the correct prediction.
- The salient explanation region if removed, stops the models ability to predict the class.

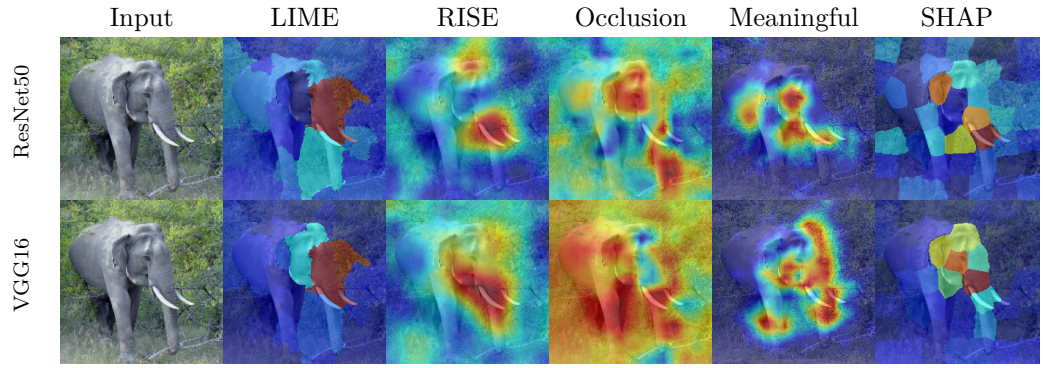

FIGURE 2.3: Examples of perturbation methods. Here we see that explanations can be created which are less coarse than the previous CAM based methods.

2.2.2.4 Perturbation Based Methods

While the previously discussed methods for creating explanations have all required access to the inner working of the CNNs, either through access to the activation maps, or the ability to retrieve gradients, perturbation methods generally do not. Perturbation methods can be broadly defined as those that treat the model as a black box, and that use multiple passes through the network in order to obtain an explanation. Typically this is in the form of an image that is altered in some way and then passed to the black box model and the subsequent prediction score stored for analysis.

Occlusion maps, introduced by Zeiler and Fergus [17], are a very early example of the perturbation technique being applied to CNNs. An occlusion map is generated by sliding a square mask around an image. At the location of the square, the image is 'turned off' by setting the pixel values to 0. At every location the pixels under the square are turned off, the image is passed to the network and the softmax score for the target class is stored. The intuition is that when a region that is important to the models prediction is set to 0, the softmax score will drop, while regions that are detrimental to the networks prediction will see an increase in the softmax score. This work is expanded further by Zintgraf *et al.* [57], who adopt a more rigorous sliding window approach by using nested patches. The authors observed that a pixel depends most strongly on the region surrounding it, and to accommodate this, proposed a conditional sampling method using the nested patches. If a feature within the image is to be labelled as useful to the network it now has to satisfy two criteria. As with the Zeiler and Fergus [17], method the feature must be

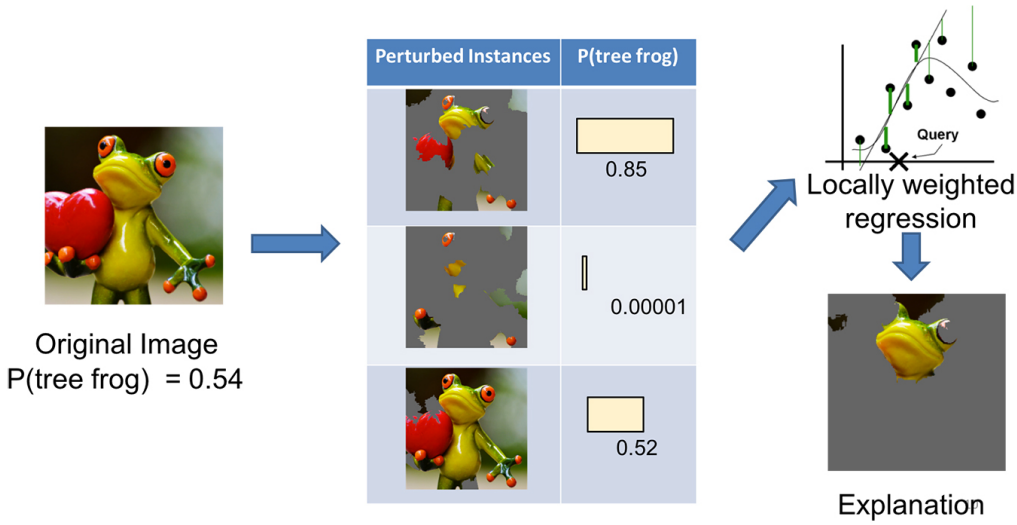

FIGURE 2.4: Overview of the LIME technique. Taken from [13]

relevant to the class under observation but now it must also be hard to predict from within the larger patch.

Local Interpretable Model-agnostic Explanations (LIME) [13] is a popular technique that takes the concept of perturbing regions of the image and expands it further by adding two novel ideas. The first is that rather than using a sliding square to perturb regions of the image, superpixels are used instead. Superpixels are contiguous groups of pixels that are typically similar in colour, although other attribute (such as texture) can be used. By using superpixels instead of square regions we immediately have a foundation that better corresponds to the input image. These superpixels are then perturbed by 'turning them on and off' (setting superpixels to 0). This is done multiple times with multiple random perturbations; by default, 1000 different perturbed version of the image are passed through the network. When the images are passed through the network, the softmax scores are stored. An example of this is shown in Figure 2.4. The second innovation is that by performing multiple perturbations and storing the softmax results from the model, a secondary interpretable model can be trained. This is typically a Least Absolute Shrinkage and Selection Operator (LASSO) or a decision tree. This interpretable model is trained on the perturbed samples and the softmax scores. Using the interpretable model we are then able to see the importance to the prediction of each superpixel to the networks final prediction.

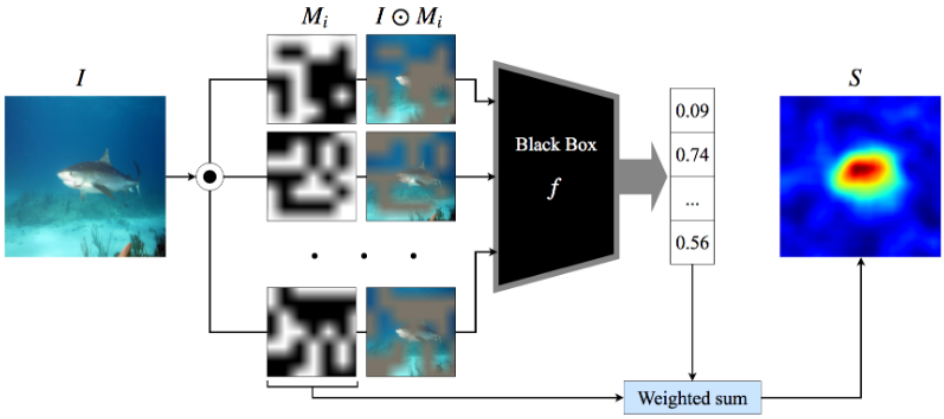

FIGURE 2.5: Overview of the RISE technique. Here I is the input image, M_i is the random mask, and S is the final resized (with interpolation) output. Taken from [9].

While LIME remains a popular method for generating explanations, concerns were raised by Petsiuk *et al.* [9] that superpixels may not correctly capture the regions important to the model. To address these concerns they introduced Randomized Input Sampling for Explanations (RISE). This is a method designed to alleviate the spatial constraints placed on the input space by LIME's superpixels. To avoid the use of superpixels, RISE uses multiple random masks to generate the perturbation regions. The masks are generated by producing random binary images smaller than the original image. These images are then upsampled to the original images size. By upsampling the original binary masks using bilinear interpolation, the values are no longer binary, rather falling in the range $[0, 1]$. Upsampling also causes the masks to have contiguous regions, rather than completely random noise. Once a mask is generated, it is multiplied with the image to be explained, and this new masked image passed to the model. The softmax score for the target class is stored along with the mask. This is repeated multiple times (4,000 and 8,000 for VGG16 and ResNet50 respectively) before a single heatmap is computed through a weighted sum of the scores and the masks. An overview of this is shown in Figure 2.5. The authors show that RISE is able to produce more accurate pixel scores than occlusion maps, Grad-CAM, and LIME. Through the use of a weak localisation metric, which we will discuss later, they also show it can better localise the object in the image compared to comparable techniques.

Similarly to the idea by Dabkowski and Gal [10], Fong and Vedaldi [11] introduced a technique with the aim of producing compact explanations through 'meaningful perturbations'. Unlike previous techniques which perturb the network with modified images and observe the softmax output, this work learns a better perturbation mask through the use of gradient descent. In this way, the perturbations becoming more akin to new training data for the mask to learn from. The authors introduce a custom loss function that aims to create an explanation mask through the use of what they refer to as the 'deletion game'. This is designed to find a mask that find the smallest region that when deleted destroys the networks ability to predict a given class. This work was improved upon with the introduction of 'extremal perturbations' by Fong and Vedaldi [58]. The core difference with their previous technique is that that explanation mask generated is constrained to take up a fixed fraction of the input image area.

SHapley Additive exPlanations (SHAP) is a technique introduced by Lundberg and Lee [59] based on Shapley values [75]. Shapley values are a method found in game theory that, in our context, assumes every feature is a player in a game and that the prediction is the payout. Shapley values aim to find the contribution of each player to the final prediction. This is done by forming coalitions of features and observing the prediction. By having different features present or absent from these coalitions, an insight into how each feature affects the prediction is gained. In the context of explaining CNNs using SHAP, it would be inefficient to treat every pixel within an image as a player as this would require every pixel to be 'turned on and off' for multiple coalitions of pixels. Although it is possible to limit the number of combinations of features used to improve efficiency, the authors solution was to combine SHAP with LIME to create KernalSHAP. The downside to KernalSHAP is that it is very inefficient to compute due to the numerous passes required, making it difficult to generate explanations for large batches of images [29].

2.2.3 Analysis of Input Centric Methods

In this section, we will discuss the pros and cons for each of the above methods and highlight areas for exploration in this thesis. In particular, we will discuss how the explanations can posses different levels of coarseness, their computational requirements, and their ease of implementation.

We begin with gradient based methods. Due to the way in which they are generated, explanations produced in this manner are very fine, that is, every individual pixel is assigned a score. This gives gradient-based explanations the ability to be very accurate with regards to their ability to determine the pixels important to the network. In contrast to this, the activation based methods are unable to reach such a level of precision due to the small sizes of the activations they are based on. The use of activation maps (which is core to the technique and cannot be changed) are typically very coarse as they are located deep into the network. Having such a coarse foundation to an explanation in turn causes the explanation itself to be coarse. However, numerous papers [10, 64, 76, 77]. have reported that this coarseness is not necessarily detrimental to an explanation as it makes it more interpretable to a human viewer.

This begs the question then, is there a family of methods that is able to bridge the gap between these coarse and fine explanations? The answer to this question is the use of perturbation methods. This family of methods allows the explainable region to be defined allowing the user to set the granularity of the explanation. By then perturbing the network in some fashion using these user defined regions, an explanation can be constructed. These explanations have also been shown to give more accurate explanations than their activation based counterparts [9, 65]. This increased accuracy arises because these perturbation techniques are able to directly question which regions of an image the network finds important. The downside to this is an often massive increase in the computational requirements needed to perform the multiple perturbation required. For example, RISE requires 4,000–8,000 perturbation (depending on network architecture) to create a single explanation. This also has a direct effect on the coarseness of the explanation. Theoretically, a perturbation method could work at the pixel level. For example with the occlusion maps technique [17] setting each pixel to 0 and observing how the score changes could produce a very fine grained explanation. This would take a 224×224 (50,176) passes for a 224×224 image. Alternatively, a method such as LIME, which randomly sets regions to 0 and observes how different combinations of regions being removed effects the network, would have an impossible search space of $2^{224\times224}$. This then bounds these perturbation techniques to again using relatively coarse regions as a basis for explanation. Having too many

regions to explain would be prohibitively costly to compute. This suggests that there is a missing method that is capable of producing medium grained explanations in a timely manner.

2.2.4 Network Centric Explanations

In this section, we explore and discuss network-centric methods for understanding and visualising a network's inner workings. We break this topic down into three subsections; these are:

- Deep visualization.
- Filter importance and labelling.
- Prototypes and criticisms.

Deep visualisation is a group of methods that creates synthetic images which produce activations within the network in a desirable way. Filter importance and labelling are methods that try to understand a filter's importance to the network and assign some label to it. Finally, prototypes and criticisms are images within a dataset that are discovered to be in some way typical or atypical of the data used to train the network.

2.2.4.1 Deep Visualization

While deep visualisation techniques all have a common approach to visualise the filter of the network through image modification, the techniques can be split into two areas [78]; these are network inversion and activation maximisation. We begin with network inversion as there seems to be less in the literature and it is not as important to this thesis as activation maximisation.

An initial method for visualising how a network represents the data it was trained on was proposed by Mahendran and Vedaldi [79]. Their idea was to invert the network so that an input image consisting of random noise could be modified to in such a way as to produce activations seen by a known real image at a specific layer. This work was improved upon by Dosovitskiy and Brox [80], who train a network to learn how to reconstruct the initial image based on the activations generated at each layer. Using this method the authors are able to generate much more accurate synthesised images than before.

More related to the work in this thesis is activation maximisation, first introduced as a concept by Erhan *et al.* [81]. They argue that rather than

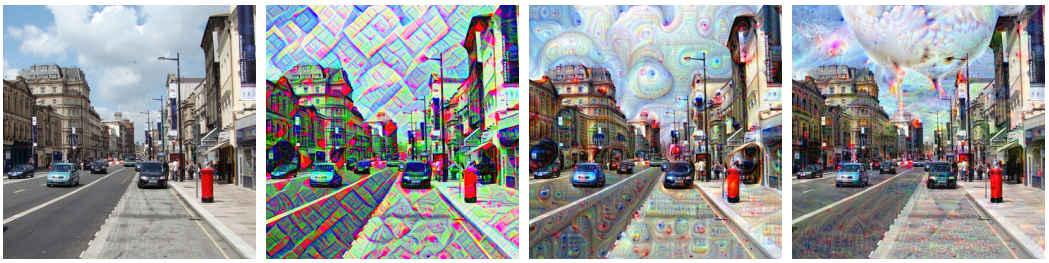

FIGURE 2.6: Examples of DeepDream using a VGG19 network trained on ImageNet. Shown from left to right are the original image, and then the image tuned for activation on the 5th, 9th and 13th convolution layers. Note how the features get more abstract the deeper the layer.

search through a set of images to find one which maximally activates a certain filter, the process can be treated as an optimisation problem. Gradient ascent is performed from the filter to the input space and the input space modified in such as way as to maximise the activation of the filter. Crucially for this implementation, they visualised a model trained in an unsupervised way meaning that a specific class could not be maximised. Class model visualisation was subsequently proposed by Simonyan *et al.* [30] which is similar to the previous work but visualises the classes of a supervised model. Because the model is now supervised, specific class scores can be maximised and a zeroed input image modified to do so. Using this method produces images that contain clear features related to the target class.

DeCAF was introduced by Donahue *et al.* [82] as a method for finding patches within an image dataset that cause a target layer to activate maximally. Zeiler and Fergus [17] built upon this idea with the introduction of visualisation technique that used a Deconvolutional Network (DeconvNet) [61]. DeconvNets, originally designed for use with unsupervised learning techniques, is in-essence a reverse CNN. This results in features being mapped to pixels rather than the other way around. They are used to probe the network by attaching one to each convolution layer. This allows the filters that activate highly to be projected back down to the pixel space for visualisation.

A popular method that expanded this work is called Inceptionism (or more popularly Deep Dream) and was introduced by Mordvinstev *et al.* [83]. There are two core differences to the previous work. The first is that once the initial image is modified, the inversion process is repeated creating a feedback

loop. The second is that adjustment of the image scale is introduced to find features of different sizes within the image. These adjustments produce images with much clearer visualisation of features. Any part of the network can be picked to maximise the input image for allowing an insight into the features learnt by different layers of the network. By altering the input image, the features learnt by a layer are able to be projected back into the image space for visualisation. This allows insights into how objects are learnt by the model, for example, a dumbbell always has an arm and hand attached to it. Olah *et al.* [84] extend this further with feature visualisation, which takes the concept from deep dream's layer visualisation but extends it to allow a greater range of the network to be explored instead of just layers. With their extension of deep dream, they are able to visualise smaller components such as neurons and channels within a layer or larger components such as the class softmax score. These are then combined to give further insights in the authors future work [85].

Another technique is the visualisation of a features through the use of extraction from the network and plotting. In the work by Karpathy [86], for example, he extracts the outputs of a fully connected layer for every image in a validation set. He refers to these as 'CNN codes'. By displaying these codes using t-SNE [87], Karpathy is able to gain an insight into how the network represents an image based on how they are clustered in the visualisation. This technique is advanced by Carter *et al.* [88] who introduced the idea of an activation atlas. They argue that the visualisation of activations as we have discussed previously [30, 83, 84] is limited by the ability to only visualise the activations for a single input. Instead, they propose taking the idea of CNN codes but applying them to neuron activations of image patches.

In order to make this manageable when creating an activation atlas with millions of input images, they reduced the activations to two dimensions as with CNN codes, and plot them. These are then pooled over a grid, and all the activations falling into a grid square are averaged. Feature visualisation [84] is then run on these average activations to create an image that optimally recreates these activations.

Wei *et al.* [89] suggest that while the previous methods have focused on visualising and understanding the differences between classes, there is a lot to be gained from visualising *intra-class* knowledge within a CNN. They propose

that a neuron is able to represent a feature from within a class in differing ways. For example, they find that a filter that activates highly for oranges can produce synthesised images of both cut oranges and untouched oranges. This suggests that the network is capable of representing the same concept using different features within the same filter.

Nguyen *et al.* [78] argued that a limitation of the previous activation maximisation techniques is that they assume that when maximising a neuron, there is only one or two types of features that activate it. They build on the work of Wei *et al.* [89] and suggest that all neurons are multi-faceted. That is, a neuron can detect multiple types of features. They propose a technique to discover images that activate each of these facets by discovering similar clusters of images with a single class of the training data. As with Karpathy's CNN codes, they pass an entire classes of images to a network and extract the hidden codes from a fully connected layer. These are reduced to a 2D embedding using t-SNE and clusters of similar activations found using k-means clustering. The average image from each of these clusters is computed and then activation maximisation is run to modify the image towards one that maximally activates a selected neuron. Performing this technique, the authors find that they create images that both better describes a neuron and its many facets, but are also more realistic in their colouring and scaling.

2.2.4.2 Filter Importance and Labelling

An interesting body of work that began prior to the recent deep learning boom was the understanding of how important the neurons within a neural network are to the network itself. This body of work has multiple applications and seems to have advanced alongside network pruning literature where the two have a common desire to understand the importance and impact of individual neurons.

An early example of understanding neuron importance was 'Optimal Brain Damage' (OBD) by Le Cun *et al.* [90]. In OBD, individual neurons are scored using the second derivative of the neuron and ranked, with the weakest ones being subsequently removed from the network. While not generating an explanation explicitly, it is the beginnings of understanding how more complex networks represent training data.

While these previous methods have been able to score neurons to show how important they are to the network or individual prediction, there has been limited attempts to apply a concrete explanation on what is activating the neuron. Early attempts were based on a visual analysis of what the individual neurons were activating on as discussed earlier in Section 2.2.2.3. By visualising the activations from the neurons, viewers could attempt to determine which features in the image they aligned with. This has the problem of potentially introducing confirmation bias to an explanation. Following this, Zhou *et al.* [91] showed how the layers of neurons in a network become object detectors. They did this by passing unseen images through the network and observing how the layer activates (as defined by the sum of the average neuron activation). The authors were then able to display the images that most activated a certain layer and find the visual similarities within them e.g. Layer 5's top 3 images were all dogs. A concept called *network dissection* was subsequently introduced by Bau *et al.* [21]. This is a technique that allows individual neurons to be given a label from a set of visual concepts. This is achieved by combining the previously discussed strand of work involving scoring neurons, along with a novel dataset called the Broadly and Densely Labelled Dataset (Broden). Broden consists of several pre-existing datasets, unified into a single set of images labelled with visual concepts. The dataset contains images labelled with concepts such as colours, textures, objects, scenes etc. in a variety of contexts. The individual neurons are labelled with a concept by passing the Broden dataset through the network and observing how strongly each neuron reacts to each visual concept. The scoring is achieved by collecting every activation map produced by each neuron and building a distribution of activations. The activation maps are then resized to the input size and threshold based on the top quantile level of the distribution. This produces a binary mask for each concept and neuron pair. For each pair, the intersection over union (IoU) is calculated. Each neuron is then labelled with its highest scoring concept. While network dissection aims to align a concept to an individual neuron, Fong and Vedaldi [92] argue that individual filters alone do not represent a single concept. Rather, they propose that semantic concepts are mapped to vector embeddings. Using this, they show that individual filters are not concept specific, but are often able to encode multiple concepts.

Kim *et al.* [35] argue that the above methods for aligning neurons with attribution labels is useful, but understanding how a network reacts to human concepts would be more beneficial. To achieve this, they introduce Quantitative Testing with Concept Activation Vectors (TCAV). TCAV is a framework that allows a models internal representations to be understood in human-friendly concepts. To begin, a dataset of images is assembled in a similar manner to the previous Broden dataset. These images contain concepts such as stripes, dotted, male, female, etc. Alongside this is a set of random images made up of anything that is not the target concept. A concept dataset is passed to the network and the activations from a desired layer are stored. This is repeated with the random dataset. These activations are then used to train a binary linear classifier to distinguish between the two datasets. Taking the normal of the hyperplane that separates the two concepts gives us the Concept Activation Vector (CAV). Directional derivatives are generated using the gradients backpropagated from the target classes softmax to the chosen activation layer. A conceptual sensitivity is generated by taking the dot product of the CAV and the gradients. A score is then generated simply by taking the ratio of the images being tested against the number of images where a concept was important. For example, zebra images would have a high ratio of images reacting positively to the striped concept.

The idea of exploring concepts was expanded by Ghorbani *et al.* [93] through the use of Automatic Concept-based Explanations (ACE). ACE builds upon the previous TCAV work. Rather than have pre-defined concepts, as with TCAV, concepts are automatically extracted from the images themselves. ACE segments images from the same class into varying sizes of superpixels. Superpixels are turned off (by setting them to 0) so that a new dataset is created containing images with only one superpixel region turned on. Patches containing only the superpixel are then passed to the network and the activations from the final layer are stored. These are then clustered to find similar concepts within the activations space. The score from TCAV is then calculated using these automatically generated concepts.

2.2.4.3 Prototypes and Criticisms

An interesting strand of work that has arisen is the use of prototypes and criticisms. Previous network-centric techniques such as those by Zeiler and

Fergus [17], Zhou *et al.* [91], and Yosinski *et al.* [66] use example images to explain the model's inner workings. However, Kim *et al.* [20] argue that unless the dataset from which the example is drawn from is 'clean' (containing only images that are prototypical of the class) then it will be hard to determine how the network is representing the data. To this end, they introduce a novel method called MMD-Critic (MMD: maximum mean discrepancy). Rather than having a method for finding an image that maximally activates some element of the network as with Zhou *et al.* [91], MMD-critic allows the discovery of both prototypes and criticisms from within a dataset. In this context, a prototype is an example image that is strongly representative of the data, while a criticism is an example image that *is not* representative of the data. These examples are created using MMD, a measure of the difference between two distributions. To find prototypes, the desired number of prototypes is specified, and then images from the dataset are greedily selected using a radial basis function (RBF) kernel. These two distribution (prototypes and the data) are measure using the MMD. Finding criticisms is done in a similar way, but instead of finding a set of prototypes that are most similar to the dataset, we search for a set of images that deviate the most.

This method was improved by Gurumoorthy *et al.* [94] through the introduction of a faster set of algorithms for discovering prototypes. This involved the use of non-negative weights to inform the prototypes. For this they trained a support vector machine (SVM) and then only took prototypes that had non-negative weights. Doing this they found that they achieved superior results compared to the previous work, finding prototypes that were more indicative of the dataset.

Of importance to note is that both of these techniques extract information about how the network views an image using the same method as Karpathy's CNN codes [86]. They extract values from the final layer of a ResNet for use with their proposed techniques.

2.2.5 Analysis of Network Centric Methods

As CNNs are used more and more in a variety of situations, the desire to be able to explain the predictions becomes paramount. However, the explanation techniques that we have discussed in the previous sections are only useful

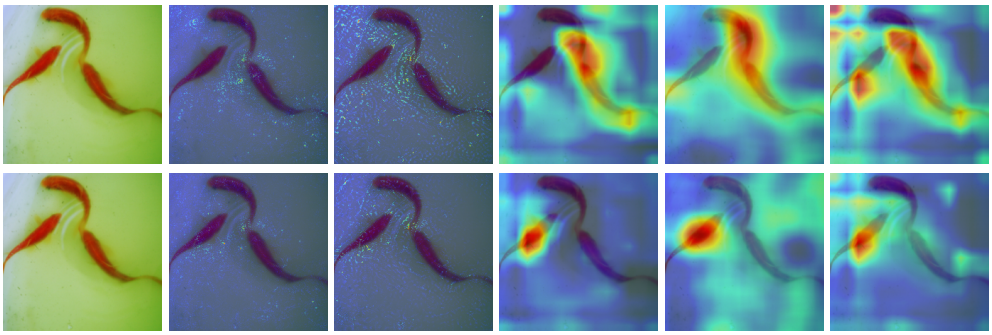

FIGURE 2.7: An image from the 'goldfish' class that is misclassified as a 'roundworm'. Existing visualisation techniques fail to give a clear explanation of the reason for failure. The top row are explanations for the (incorrectly) predicted class, while the bottom row are explanations for the ground truth class. The network used is VGG16. A plausible explanation for the failure is shown later in Chapter 6.

when we are trying to explain how a class was *correctly* predicted. This is of limited use when trying to ascertain why a model may have failed to correctly classify an input. This has the potential to be an important area of research, especially when dealing with vision systems that are integrated into critical applications. In critical applications, such as autonomous driving, surveillance, or medical applications, failure of the system could endanger lives. We have seen previously that when these systems have failed, an investigation is taken to find their failures. For example, the Uber self-driving incident in 2018 [95] where a pedestrian was killed. In such a scenario, where the underlying reasons for failure are being sought, it may be useful to both identify the reasons the model may have failed, and to be able to visualise those reasons on the input image.

We begin by analysing why current explanation methods are not well suited to explaining failure. Perhaps the most crucial aspect is that the previous explanation methods are good at identifying how the network interprets a single image, with little regard to how the model has learnt to represent the class as a whole. A key ability of all of the methods is to be able to produce class specific explanations, either from backpropagation from the classes' softmax score, or in the case of perturbation methods, by observing how the classes' softmax score change. This means that these techniques are perfectly capable of producing explanations for both the ground truth and incorrectly predicted classes. An example of this is shown in Figure 2.7. In this figure, we show an

image from the 'goldfish' class that has been misclassified as a 'roundworm'. The top row of the figure shows explanations produced for the incorrectly predicted class, while the bottom row are the visualisations for the ground truth class. The problem with this is that in both sets of visualisations, the regions of the image highlighted as important to the class are all regions that could realistically be used to define the goldfish class. It is, therefore, not apparent from these visualisations why the network has failed to correctly classify the image.

A possible reason for the failure of these techniques is that the model has learnt to represent a class based on the features found in the training data. However, the techniques in Figure 2.7 have no insight into how the models have learnt to represent a class. They are only able to leverage the features in the image at that time. It therefore make sense to somehow root the explanation for failure in the context of the training data.

In itself this is not a novel view to take, as previously discussed methods have also attempted to interpret the model through the use of the dataset. An example of this is the work by Kim *et al.* [20] and Gurumoorthy *et al.* [94]. In these works, methods were proposed that allowed for the discovery of two categories of images from within the training dataset: prototypes and criticisms. These are images that are strongly typical of the dataset, and images that are strongly atypically respectively. Seeking to understand how the model has learnt to represent the training data is a useful step forward. The prototype images allow a better understanding of what type of image would allow the model to produce a strong prediction, while the criticisms allow us an understanding of why something may fail. However, this again only begins to touch on the reasons for failure. Are the images found by the criticisms atypical because they do not contain features that were expected, or is there some feature present that is acting in an adversarial way, making the network classify it incorrectly?

To begin to discover possible answers to these questions, we believe that further insight into how the network discriminates between classes is required. Again, this in itself is not a unique proposition. Techniques such as Grad-CAM visualise the activations from the final convolution layer by assigning a weight to each of the filters activations. A visualisation produced by Grad-CAM is therefore an insight into the contribution of the final convolution layers filters.

This is important as the individual filters have learnt to represent features found within the training data, that when passed through the classification layers allows a prediction to be made.

2.3 Video Interpretability Techniques

So far we have only looked at input-centric explanation techniques in relation to explaining networks which use images as an input. In this section we discuss how these techniques are expanded to work with video networks.

As interpretability techniques have developed for CNNs, the primary application has been image classification networks. Networks that take a spatio-temporal volume as an input have largely been either overlooked despite often having a substantially different network design that causes image techniques to either be inefficient to compute, or inaccurate due to the addition of the temporal element. In particular, activation based techniques such as CAM suffer as they are are based on visualising the final activation layer and projecting it back to the original image. In action recognition networks, the temporal volume is reduced alongside the spatial dimension. Conversely, black box techniques which perturb the input space (such as LIME) can be effective as they have no need to try and reconstruct the input space from a lower resolution. However, using techniques which perturb the input space of an action recognition network have the potential to be vastly inefficient as the networks contain more parameters compared to image classification networks, and the input volume is larger.

In this section, we will discuss methods that have been introduced and used for action recognition networks. We again break them down into three subsections:

- Activation based methods.
- Perturbation based methods.
- Gradient based methods.

2.3.1 Activation Based Methods

As mentioned previously, CAM techniques such as the original CAM, Grad-CAM, and Grad-CAM++, rely on the final activation layer to form, the core of the visualisation. From here, it is simply a case of weighting the activations

and scaling back to the original input size. However, in action recognition networks that typically use 3D convolutions, there is a temporal element to the activation maps that must also be considered. As the input passes through the network, not only is the spatial component reduced, so is the temporal element. This means that when the activation maps are resized back to the input size, the explanation has to be stretched across frames, causing coarseness in the temporal dimension.

A number of approaches have been taken to aligning the original 2D intent of these techniques with an additional temporal domain. In the simplest case, the CAM can be generated as it would in a 2D network. This produces a CAM with the same dimensions as the final activation layer, for example in C3D this is a 14×14×2 (height×width×depth), while in I3D [96], R(2+1)D [38] or ResNet3D [38], this is 7×7×2. A number of techniques have attempted to be built on top of this simple method. In the Saliency Tube work by Stergiou *et al.* [36], this is very slightly modified so that the the activation maps themselves from the final convolution layer, following batch normalisation, are used to weight the activation maps rather than the gradients.

In Grad-CAM++ [23], the authors discuss the application of their technique for use in action recognition networks. They propose a technique similar to the input⊙gradient technique proposed by Shrikumar *et al.* [32] whereby the resized CAM is combined with the spatio-temporal volume via point-wise multiplication. None of the above techniques, to our knowledge, have been subject to an empirically sound analysis using techniques that measure the accuracy of the generated heatmaps. Without first performing these experiments it is difficult to compare them against each other. In the Grad-CAM++ work, the authors performed an experiment (called faithfulness) where they produce an explanation map based on the Hadamard product of the input image and the generated CAM. This explanation map is then passed back into the model and then a number of comparisons are performed based on the softmax score. However, this is not particularly informative in the context of how accurate the heatmaps are as the aim of Grad-CAM++ is to maximise the heatmap so it covers as much of the target object as possible. This, in turn, produces explanation maps that have more object context than those produced by Grad-CAM and therefore produce better scores in this experiment. We explore this metric further in Chapter 5.

2.3.2 Perturbation Based Methods

Perturbation methods such as LIME or RISE could work perfectly well with spatio-temporal volumes as they treat the network as a black box and are agnostic to input. That is, as they are based on perturbing the input space to discover the relevant regions, they are able to incorporate the temporal element directly into their explanation of the model. A recently introduced technique by Li *et al.* [41] takes the method proposed by Fong and Vedaldi [11] and reconfigures it to work with action recognition networks. This is done through the introduction of a loss function that helps create masks that are smooth in both the spatial and temporal dimensions.

The above perturbation methods require multiple passes through the network to generate a single explanation, therefore these quickly become incredibly inefficient when working with spatio-temporal volumes.

2.3.3 Gradient Based Methods

Gradient based methods, still perform admirably for networks that use a spatio-temporal volume as the gradients are back-propagated back to the original input space. This means that the inherent problem faced by activation based methods is avoided as their is no spatial or temporal resizing required. However, it is still a more difficult task to understand saliency maps based on individual pixel scores compared to more coherent heatmaps such as CAM or those produced using perturbation techniques. This is exacerbated as often there is no cohesion between frames leading to explanations produced this way to appear like noise.

A number of methods have been proposed that modify these gradient-based techniques to make them more suitable to action recognition networks. For example, the work by Hiley *et al.* [97] takes the Deep Taylor Decomposition (DTD) and adapts it to work with action recognition networks. They do this by introducing a discriminative relevance model that is able to split the relevancies assigned to spatial and temporal elements. This allows the regions of the input that are relevant to the motion aspect of the network to be highlighted. Following from this, Hiley *et al.* [42] proposed a method that takes a DTD explanation and uses a 3D Sobel filter to create “edges in time”.

Methods such as these which aim to decompose the spatial and temporal aspects of the network are still relatively rare in the literature.

A further example of this modification of existing image based techniques is the work by Bargal *et al.* [98]. This takes excitation backpropagation [64] and modifies it to work with action recognition networks that contain an RNN element. In particular, they target their technique at CNN-LSTM models such as those by Donahue *et al.* [99].

2.3.4 Analysis of Video Interpretability Techniques

From discussing these methods we see that there is still room for an explanation method such as SWAG to work well in a spatio-temporal environment. The activation based methods (Grad-CAM, Grad-CAM++, and Saliency Tubes) are still limited by their spatial coarseness, but are now further constrained in usefulness by the coarseness of the temporal element. Gradient based methods have the potential to be accurate, but again suffer from being so fine as to limit the user-friendliness of the methods [41]. Perturbation methods, either pre-existing or those that have been proposed specifically for action recognition networks suffer from the same problem as before. That is they require multiple passes through the network before an explanation can be created.

2.4 Post-hoc Explanation Evaluation Metrics

As methods for producing post-hoc explanations have increased, attempts to understand how well a visual explanation truly explains a models prediction have closely followed. The issue with evaluating explanations is that there is no ground truth present to compare explanations against. Every model has the potential to represent the data differently depending on its architecture or even how the starting weights were randomly initialised. This is problematic and requires evaluation metrics that attempt to align an explanation to the model through the use of perturbations, that is removing regions of an image based on the explanation. Other methods can use synthetic datasets to try and understand whether an explanation attributes a known region of an image. Another alternative is to see how well an explanation localises to a known ground-truth bounding box within an image (which is supplied as part of

the dataset). In this section, we will discuss a range of post-hoc evaluation techniques. While we do not propose a novel evaluation metric in this thesis per se, we do use a number of these metrics through the chapters so it is important to outline them up front.

2.4.1 Qualitative Techniques

The first, and perhaps the most obvious, method for evaluating an explanation is to perform a visual inspection. Doing this allows us to compare different methods of explanations to assess how their visual aspects compare. For example: is a method coarse or fine, or is it giving an explanation as expected. However, the major drawback to using this as a means of evaluating the accuracy of a method is that it requires the viewer to impart some pre-formed judgement of which image features *they think* the network should be using. This opens up the possibility of confirmation bias that could impair a judgement on both whether an explanation is correct or incorrect. For example, Ribeiro *et al.* [13] show an example where a network trained to classify dogs, shows that the networks uses snow on the ground as a primary indicator for a breed of dog, rather than any of the features on the dog. If this example where to be shown to a viewer, they could think the explanation is incorrect as it is looking at the dog in the image, without realising it is a fault of the network not conforming to their pre-determined view of how the network should work. Therefore, as has been noted many times in the interpretability literature [9, 100, 101], a human visual inspection of an explanation as a method for ensuring accuracy is discouraged.

2.4.2 Explanation Accuracy

The primary method for ensuring that an explanation has been created that successfully aligns with the networks methods of discrimination is to measure the accuracy (which we will describe in this section). In this case, the accuracy is typically discovered by altering the image in some way based on the recommendation of an explanation, and then seeing how the network reacts.

In this subsection we begin by discussing the difference between local and global accuracy, before exploring examples of both.

2.4.2.1 Local vs Global Accuracy

So far we have mentioned the word accuracy as a singular metric, however, there are two distinct concepts of accuracy found in the literature. These are local accuracy and global accuracy. Local accuracy is the most common method for quantitatively measuring an explanation's relationship to its corresponding network. Local accuracy seeks to quantitatively measure how well an an explanation lines up with a model *at that point in time*. By this we mean it actively reflects how the model interprets the input image. Global accuracy on the other hand seeks to find all regions of the image that *may at some point* to be useful to the model regardless of whether they are or not in that instance. Global accuracy is also harder to measure as well as it requires a series of models to be trained using modified datasets.

In many ways local and global accuracy could be seen to be related to the ideas of soundness and completeness we discussed in the introduction. This depends somewhat on how we choose to align the soundness and completeness concepts with the underlying model. For example, soundness relates to how truthful an explanation is to describing the network. In this context, soundness could be seen to be represented by local accuracy, with its metrics based on understanding how an explanation represents the way the network has learnt to represent a given class. Contrasting this would be completeness, which relates to the extent to which the network is described. This could either be seen as again the local metric, if we are only concerned about the network in its trained state. However, if we care about understanding the networks ability to learn anything that represents a given class, then equating completeness to global accuracy makes more sense. In the following section we explore various methods of measuring local and global accuracy.

2.4.2.2 Local Accuracy

A very early method of quantitatively measuring how well an explanation method performed was proposed by Zeiler and Fergus [17] in tandem with their work on occlusion maps. Here, they took images of dogs and separately obscured (by setting to 0) each eye, the nose and a random point with a square. They are then able to observe how the mean feature vectors change for each layer depending on which area of the image is obscured. While this was only

done over a very small test set (5 images), it is an early example of modifying a region of the image by setting it to 0 (the mean value of the normalised RGB channels) and observing how some measurement of the network changes.

This idea forms the basis of a number of techniques that can be roughly grouped together and are referred to by Hooker *et al.* [102] as modification-based evaluation metrics. The general thrust of this idea is that we begin with an image, and through a series of modifications to that image, we end up with a heavily modified image. At each modification, the image is passed to the network and some metric is stored, classification accuracy, softmax score etc.

Samek *et al.* [103] seek to formalise this idea of modifying an image and observing how the network's predictions are affected with the introduction of a heatmap evaluation framework. This is the introduction of a greedy iterative procedure that removes pixels from an image in an ordered sequence and stores the softmax of the image class. The ordered sequence is derived from a heatmap produced my the explanation techniques whereby the most important regions of the image are defined by the highest values on the heatmap. The authors propose modifying them most important to least important. In the proposed heatmap evaluation framework, the pixel to be removed is located and all pixels in a 9×9 neighbourhood are replaced with randomly drawn values. This gives a set of results, that when charted as the number of modification steps vs the class softmax score should show a steep drop as the important regions of the image are removed. A quantitative value is taken by measuring the area above the curve.

Kindermans *et al.* [104] built upon the method introduced by Samet *et al.* and split the image into non-overlapping 9×9 patches. The pixel values from the heatmap that fall within each patch are summed and then ranked highest to lowest. The patches are then iteratively replaced with their mean per colour channel and the classifiers output stored. As before, a steeper drop in performance indicates a more accurate explanation technique.

A measure of how well a method scores pixels was developed by Fong *et al.* [11] and builds upon the previous techniques. However, a crucial difference is that rather than use the 9×9 patch as a basis for region removal, a mask is produced by thresholding the heatmap. This allows regions of the image to be removed in a more natural way. This mask is produced by slicing the explanation into binary masks by thresholding at threshold α where

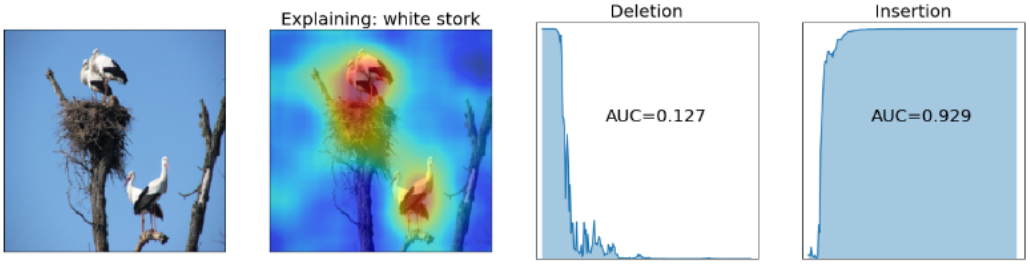

FIGURE 2.8: Overview of the deletion and insertion local accuracy techniques. Taken from [9]

$\alpha \in [0 : 0.05 : 0.95]$. As before, the most relevant pixels are modified first and the value of the softmax for the target class observed. It is expected that unless the pixels relevant for the given class are modified, the softmax score for the target class should remain consistent.

Dabkowski and Gal [10] explore concepts related to this idea of removing the most important regions first and formalise it. They suggest two concepts related to the previous work that should be explored. These are the Smallest Destroying Region (SDR) and the Smallest Sufficient Region (SSR). SDR is the smallest region of the image that can be removed that causes the model to make an *incorrect* prediction. SSR is the opposite, the smallest region of the image left remaining that allows the model to make a *correct* prediction. They introduce a new saliency metric based on SSR that crops the image to contain the salient region as defined by the explanation heatmap. The cropped image is passed to the network and the class softmax recorded. The log of the softmax score is subtracted from the log of the cropped are (as a fraction of the image size). This gives a score that can be used to determine a metric performs for SSR. The lower the score, the better the SSR explanation.

In the work by Petsiuk *et al.* [9], the strands from the previously mentioned ideas are drawn together to give two new explanation metrics. These are deletion and insertion metrics. Petsiuk *et al.* [9] either start with the original image and incrementally remove the most salient images, or start with a blank image and reintroduce the most important pixels. At each stage of insertion or deletion the altered image is fed to the network and the softmax score for the target class stored. However, as opposed to the previous work, Petsiuk *et al.* produce a probability curve using the stored scores as a function of the removed pixels and then measure the area under the curve (AUC).

For deletion of pixels, a sharp drop towards the beginning of removal (and therefore a low AUC) indicates a good explanation as it suggests the method being evaluated has correctly identified the most important pixels used by the network. Conversely, for inserting pixels, a high AUC indicates that the evaluated method is reintroducing the most important pixels allowing the probability of the prediction to increase rapidly. An example of this technique is shown in Figure 2.8. This is the method that we will primarily use for measuring local accuracy in this thesis.

Performance Information Curves (PICs) were introduced alongside the XRAI [65] method (discussed previously in the gradient-based methods) as a way of measuring how well an explanation ranks the importance of regions of the image against the information content in the image. This use of the information content is distinctly different to the previous metrics discussed which measures scoring accuracy against the amount of pixels altered (typically re-introduced or removed). In the XRAI paper [65], the decision to use the information content is suggested as the usual method of measuring the amount of pixels altered tends to give better results for explanation methods that are very fine. That is, they can highlight single pixels as being important rather than large contiguous regions such as with Grad-CAM or LIME. Using the information content gives results that perform the other way, penalising methods that do not produce cohesive explanations, despite this being potentially more accurate.

PICs are proposed in two varieties: Accuracy Information Curves (AIC) and Softmax Information Curves (SIC). For AIC the y-axis is given as the accuracy calculated for all the images at each information content level. For SIC the y-axis is the median of the softmax scores at an information level compared to the original image. For both AIC and SIC the y-axis values are normalised between 0 and 1. A single value is extracted from both AIC and SIC using the same AUC technique as the RISE modification-based evaluation metric.

A question that needs to be asked about the above evaluation metrics is what it means to modify a region of an image. Fong and Vedaldi [11] suggest that their are three common ways to delete an image region. These are:

1. **Constant Value**: Here we set each pixel within a region to a constant value. This is typically 0 due to the normalising of input images reulting

in 0 being the mean value each channel. Alternatively, some metrics set RGB channels separately to replace the pixels with the average colour of the input image.

2. **Blurring:** Rather than using a constant value, the use of blurring replaces a pixel with its counterpart from a blurred version of the input image. This method of removing pixels is used as there is concern that introducing hard edges as with a constant value may create an image that is out of distribution. It has been argued that introducing blurred counterpart pixels instead creates a more realistic image [65].
3. **Noise:** Randomly assigning a value to a pixel within the range of the image values. Fong and Vedaldi [11] sample them from a Gaussian distribution. As with using a constant value, the use of noise has been criticised for creating artificially strong boundaries within the image [11].

An alternative to these deletion methods called pixel flipping was introduced by Bach *et al.* [55]. As input images to CNNs are typical normalised to be centred around 0, multiplying the value to delete by -1 will cause it to 'flip'. Whilst this technique has been used subsequently [71], it seems to have not been used as commonly as the three mentioned above.

2.4.2.3 Global Accuracy

The previously discussed local accuracy metrics have all aimed to measure the accuracy of an explanation technique, i.e. how well an explanation can determine which features are required for a trained networks prediction. However, concerns have been raised about these techniques as the removal of features form the input image (either in the form or zeroing or blurring) may lead to the image now falling out of the distribution of the training data [102]. Remove and Retrain (ROAR) is a metric introduced by Hooker *et al.* [102] that aims to tackle this problem by continuously retraining the network to ensure that images with removed features do not fall out of distribution. This is done in an incremental way. To begin, a network is trained as usual and an explanation produced for each image of both the training and validation set. New training and validation sets are generated by removing (setting to 0) the most important $t\%$ of the image features. These new datasets are then used to retrain the network and and store the validation accuracy. This is repeated

for $t = [0, 10, 30, 50, 70, 90]$. To ensure that the comparisons are fair and that the randomness introduced during the retraining does not skew results, the process is repeated 5 times. This results in every t step requiring 5 models to be retrained, therefore for a single explanation method to be tested on a single model, 30 models are required. For datasets with a large number of images (i.e ImageNet) or computationally inefficient explanation techniques, this metric can take an inordinate amount of time to compute [105].

2.5 Gap Analysis Summary

The analysis of existing techniques discussed in this Chapter has highlighted a number of gaps in the existing body of explainability work. In this section we highlight these gaps, and relate them to the following chapters in this Thesis.

With regards to input centric methods we identified the inability for current methods to produce explanations that are neither too fine or too coarse in an efficient way. Existing techniques that are able to have control over the coarseness of the explanations produced are inefficient to run, due to the underlying reliance on network perturbation. We address this issue using two approaches. The first is addressed in Chapter 3 through the introduction of a technique which allows for explanation of arbitrary coarseness to be created in a single forward and backward pass. The second approach is addressed in Chapter 5 and uses existing CAM methods to produce multiple explanations that can be recombined to create an explanation with a higher spatial resolution.

Having discussed input centric methods for images, we also investigated those designed for use with video as an input. We identified both a lack of explanation techniques for use with video inputs, but also the same gap in the research as explanations for images. However, due to the temporal element of a network using video as an input, this need was greatly increased due to the reduction of both spatial and temporal resolution. We found that existing techniques were often inappropriate for coping with this decreased temporal resolution when resizing. This is addressed in Chapter 4 where we discuss how we are able to create explanations that are able to identify regions important to the network in both the spatial and temporal dimensions.

Finally, we identified a gap in both existing input and network centric methods for explaining why an input might have failed to be classified correctly.

Input centric methods are almost exclusively designed to identify reasons why a network has made a successful prediction. With network centric methods, this is not the case, for example we saw how prototypes and criticisms were used to identify how a network may have learnt to represent a given class in a dataset. However, a gap exists in the literature for connecting the two techniques together. To the best of our knowledge, no technique is able to highlight regions of an image that contributed to a networks failure to predict a class. We address this in 6 where we show how taking inspiration from network centric techniques allows us to explore individual failed predictions in more detail.

2.6 Chapter Summary

In this chapter, we have outlined the common methods for creating both network- and input-centric explanations. We have also discussed various methodologies for measuring how well these explanation methods perform. These will be useful going forward when we begin to compare our proposed methods with existent explanations methods. Of particular importance are the areas we discussed that we believe are fertile grounds for research. In particular for image-centric explanations we identified the lack of mid-grained explanations that can be computed in an efficient way. This research will form the basis of Chapters 3, 4, and 5.

From network-centric methods, we highlighted that the concepts of using prototypes could be linked up with the idea of explaining networks through an understanding of filter activations. This research is reported in Chapter 6.

CHAPTER

3 SWAG: Superpixels Weighted by Average Gradients

3.1 Introduction

In the literature review, we highlighted the idea that explanations can have varying degrees of coarseness. Here, coarseness refers to the size of the region that is assigned a relevance score. Fine-grained explanations assign a score to every pixel while coarse-grained explanations assign scores to large regions of the image. Typically, there is some constraint placed on the coarseness of an explanation. For example, activation based methods are constrained to the size of the activation maps used; perturbation based methods are constrained by the acceptable time budget when passing images through the network. We investigate if we are able to find a middle ground between these fine grained and coarse grained explanations and if this can be achieved in a computationally efficient way. To our knowledge, this approach has not been taken before, or considered in other contexts such as weakly supervised learning.

In this chapter, we therefore propose two novel complementary techniques that when combined produce an explanation that has a number of desirable properties when compared to other similar techniques. The first of the techniques we propose is Superpixels Weighted by Average Gradients (SWAG). This is a method that uses superpixels to segment an input image into discrete regions, before weighting each superpixel using the backpropagated gradient. This is a distinct to semantic segmentation which requires a ground truth label for each segment being labelled. SWAG uses superpixels that are created

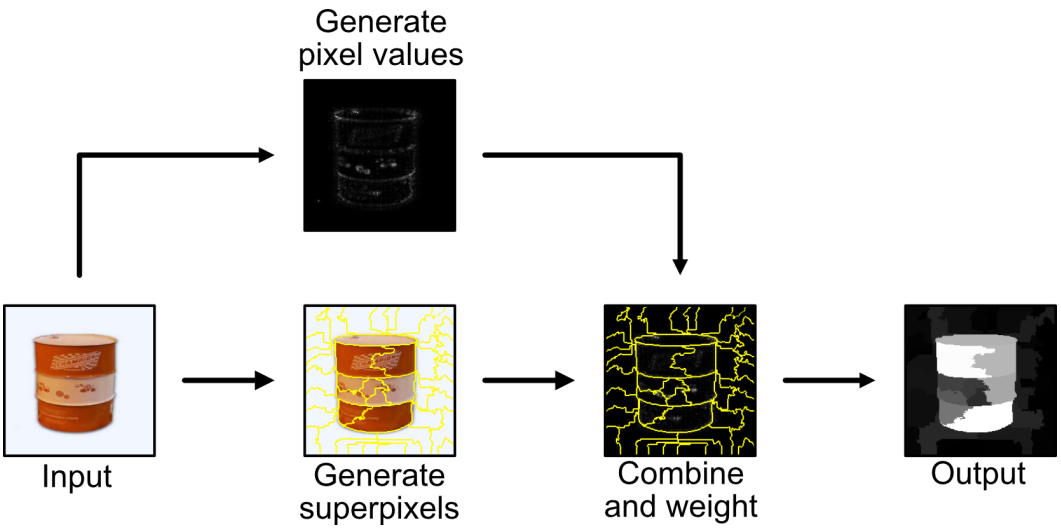

FIGURE 3.1: A high level overview of how the SWAG method works.

based on the properties of an image. As such the superpixels do not represent semantic concepts as they would in semantic segmentation. An overview of SWAG is shown in Figure 3.1.

The complementary technique we propose in this chapter, is the use of superpixels created especially to achieve accurate explanations. We explore how the same backpropagated gradients used for SWAG can also be integrated into the superpixel creation process. We begin by providing some motivation for why these techniques may prove useful to a wider audience. We then perform a number of experiments that seek to understand both the local and global accuracy of our technique, its weak-localisation ability, and its computational efficiency. We also show how our proposed superpixel generation method can be used as a drop in replacement for use in existing explanation techniques such as LIME. This chapter contains work we have presented in the following papers [44, 45].

3.2 Motivation

In this section, we outline the motivation behind the work in this chapter. In particular we explore and discuss the gap that exists between explanation methods with regards to how fine or coarse the produced visualisation is. We discussed in the previous chapter (Section 2.2.2) three of the primary methods for creating explanations: gradient based, activation map based,

FIGURE 3.2: Comparison showing how coarse activation-based methods are in reality, with the smoothness of their appearance being primarily produced by bilinear interpolation during resizing. Left: original image. Centre: Grad-CAM without interpolation. Right: Grad-CAM. Both explanations created using ResNet50.

and perturbation based. The three techniques all produce explanations with varying degrees of coarseness, that is, the size of the region within an image they assign a score to. Fine-grained techniques such as gradient based methods assign a single value to every pixel within an image. At the other end of the scale are the coarse grained explanations produced by the activation map based methods. Explanations produced by these are limited by the size of the activation map on which they are based. For example, an explanation produced for a ResNet50 [68] model using Grad-CAM will be using a 7×7 activation map. This means that only 49 scores (representing importance to the networks prediction)) are assigned to all the pixels present in an image (50,176 for a 224×224 input). The smooth appearance of explanations produced by CAM based methods [15, 23, 34, 72] and RISE [9] is a result of bilinear interpolation used when the initial explanation is resized to match the size of the input image. Resizing without the interpolation shows how the scores based on the activation map are assigned in reality. An example of this is shown in Figure 3.2. This figure highlights just how coarse this method of explanation is. Despite the coarseness of these explanations, there must be some benefit to using them or they would not have thrived within the computer vision community? The answer to this comes in their ease of interpretability. Yang and Kim [76] suggest that a highly interpretable explanation is one that has a high accuracy of feature attribution. That is, an explanation assigns high scores only to those features that are important to the model. This is distinct to the concept of saliency, as the regions a human may find useful for discriminating a given class, may not be those useful to the model. Using feature

attribution effectively removes the human from the process of determining what makes a good explanation. In their work, Benchmarking Attribution Methods (BAM), Yang and Kim show that coarser methods such as Grad-CAM perform significantly better at only assigning high scores to regions considered important by the model. Conversely, fine-grained methods such as vanilla gradients and guided backpropagation perform poorly in this respect.

However, accuracy of feature attribution is not the same as the accuracy of an explanation technique. Features tend to be a collection of pixels that make up some object. For example, in a cat detection network, a human may see a feature as being the ear of a cat. However, the network is unlikely to be using that feature in the same way as a human may label it, instead using a subset of pixels from within the feature to actually make a prediction. Intuitively this makes sense as it has been shown repeatedly through adversarial examples that, by changing only a small selection of pixels, the accuracy of the model can be compromised [106]. This suggests that explanations may score a region as being important, whereas in reality only a sub-region within the high scoring region is important to the networks prediction. In extreme cases, only a single pixel is required to be modified in order to alter a networks prediction [107]. Important to note is that the single pixel attacks are not randomly chosen pixels, but are discovered using a perturbation based technique. However, it is still illuminating that the discover of a single pixel, when altered, is able to adversely effect the network. As discussed in the background chapter, local accuracy is a good measure of the ability of an explanation technique to correctly score the regions of an image most important to a model. It is important to note here that the local accuracy technique is able to assign a score to fine explanations (where every pixel has a score) and to coarse explanations. This is beneficial as we have discussed with the adversarial examples that coarse regions do not necessarily reflect how the network represents features. Having a technique that can work for all explanations can give us a good insight into how accurate a range of explanation granularities are. Using the deletion technique introduced by Petsiuk *et al.* [9] (discussed further in Section 3.7.2.1), we compute local accuracy scores for a range of fine and coarse grained methods and plot them alongside their score to pixel ratio. The score to pixel ratio is a measure of how many pixels are represented by a unique score. For example, gradient based methods assign a score to every pixel so have a 1:1 ratio whereas CAM based

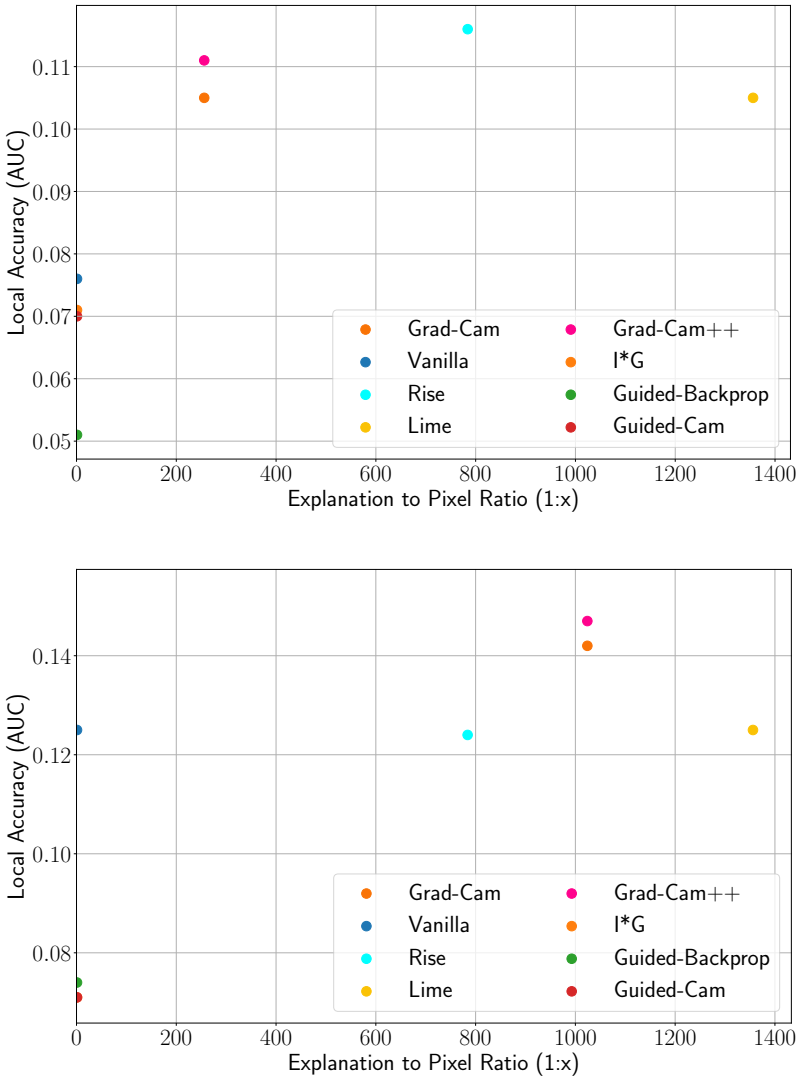

FIGURE 3.3: A comparison using ImageNet between the local accuracy score (lower is better) and the score to pixel ratio. Top: VGG16. Bottom: ResNet50.

techniques use the coarse activation map. For ResNet (7×7) this would give a ratio of 1:1024, for VGG16 (14×14), this would be 1:256. It should be noted that the use of a 7×7 final convolution is much more common now than the larger version [68, 108–110]. This analysis is shown in Figure 3.3. In particular, note the difference in local accuracy between the fine-grained methods. With the exception of vanilla gradients for ResNet50, they are all more accurate than the coarse methods.

Perturbation techniques such as LIME or RISE have the ability to alter the size of the regions that they assign a score to, however this comes at the

cost of increasing computational requirements. For example, LIME defaults to 1,000 perturbations for approximately 30 regions per image (using ImageNet). In a perfect scenario LIME would be able to test every combination of regions being set to 0 or not. However, for every region k that is added, the potential number of combinations doubles (yielding 2^k total). This results in potentially degraded performance unless the number of perturbations increases accordingly. Increasing the number of perturbations allows a deeper insight into how removing different regions of the input image effects the networks prediction. The more combinations of input image that can be passed through then network, the more robust the understanding of which regions are important to the network. A similar constraint affects RISE, where a large number of randomly generated 8×8 boolean grids are required. However, increasing the size of these would require an increased number of passes through the network in order to maintain consistent results. As RISE currently suggests 4,000 and 8,000 passes for VGG16 and ResNet50 respectively, this could require a large increase if a larger grid is used. Another concern raised about perturbation techniques is how the features are selected as a basis for visualisation. Petsiuk *et al.* [9] suggest that the superpixels used by LIME (which are arbitrarily sized and adhere to dominant boundaries within the image) may not capture regions suitable for the explanation. Intuitively this may be true as the superpixels used by LIME are generated based upon the RGB values of the image and therefore align to the boundaries within the image. It may be that a feature that is important to a networks prediction is located within the same superpixel as one that is not. However, any explanation score would be assigned to a region containing both which may be misleading. The simple solution is to increase the number of superpixels used within the explanation, but as we have discussed, this leads to a large increase in computational requirements. To account for this, the XRAI method which uses superpixels as a basis for it's explanations, seeks to adjust the superpixels to incorporate certain regions of the image. In this case, the XRAI authors noted that edges in an image are often highlighted as being important by gradient-based methods. To account for this and ensure their superpixels do not sit on these edges, they manually dilate all the superpixels by 5 pixels to incorporate these edges. This does not necessarily solve the problem of how best to segment the image for the creation of an explanation. Rather, it is a heavy handed approach, expanding

superpixels to incorporate regions that may not be beneficial to the underlying explanation.

It, therefore, seems that their is a need for an explanation technique that is able to bridge the divide between fine and coarse grained visualisation techniques. Doing so would allow us to achieve the higher interpretability rates of coarse grained methods, with the improved local accuracy results for fine-grained techniques. At this stage in the thesis we are only exploring techniques which find regions which contribute to a networks classification ability. In the penultimate chapter we explore techniques which are designed to discover regions of an image that can contribute to failure. In this chapter, we suggest two novel approaches to this problem:

- The first novel approach is Superpixels Weighted by Average Gradients (SWAG), a method for marrying the region based approach of techniques such as LIME with the gradient based approach.
- The second novel approach is to generate superpixels using gradient based explanations, allowing us to create superpixels that better reflect the regions deemed important by the model. This is a distinct but complementary approach to the first one, and has applications for explanation techniques besides SWAG which use superpixels.

3.3 Superpixels Weighted by Average Gradients (SWAG)

The first technique that is introduced in this chapter is Superpixels Weighted by Average Gradients (SWAG). This is a method that allows us to create regions of any size and subsequently score how important the contents of that region are to the network. The score is normalised by the size of the region to ensure larger regions do not dominate. The core idea of our technique is to join the accuracy of pixel-based methods with the interpretability of region-based methods. To achieve this, we take gradient values backpropagated to the input, and pool them into discrete regions. With SWAG we therefore have two distinct components that must be generated and combined. The first is the method of generating the gradients. Any pixel-based method should be able to offer an accuracy improvement over existing region-based techniques if used correctly. For this work, we chose to use guided-backpropagation as the method for

generating pixel-scores. Here, a larger gradient value indicates that the corresponding pixel of the input image is more important to the networks prediction. We opted for this method primarily because it was both simple to implement and offered improved accuracy over vanilla backpropagation. To briefly recap, guided backpropagation follows the same procedure as vanilla backpropagation, but during the backward pass, only gradients whose corresponding activations were positive on the forward pass are allowed to propagate.

Guided-backpropagation produces an image $M \in \mathbb{R}^{224\times224\times3}$. As per [30], each pixel (i, j) in the gradient (back-propagated to the input layer in a non-cumulative way) is obtained by taking the maximum of the absolute values: $M_{i,j} = \max_c |M_{i,j,c}|$, where c is the colour channel. With a method in place for generating individual scores, we now require a set of regions that can be weighted.

The choice of region within an image that is assigned a score is typically a characteristic of an explanation that is uncontrollable. For example, gradient based techniques give a score to every pixel while CAM based methods are constrained to using the activation map. It is interesting then to look at the decision taken by techniques that are able to control the regions that they assign scores to. These primarily fall into two categories, the use of a rigid grid [9, 17] or the use of superpixels [13, 65]. The use of a grid is often used for its simplicity, the RISE technique [9] uses it to create simple regions that can be continuously perturbed and then combined with one another. The authors of LIME and XRAI chose superpixels as they claim that they correspond to the human intuition of what constitutes a meaningful region within an image. However, to our knowledge no human studies have been undertaken to confirm this claim. As noted previously, perturbation techniques struggle with computational inefficiency when tasked with using large amounts of superpixels as a basis for an explanation. It is important to note that the perturbations of the networks used by these techniques are not used to define the superpixels segments themselves as with Markov Chain Monte Carlo (MCMC) [111, 112]. Rather, these perturbation are used to explore how the prediction of the network changes as combinations of regions are masked or unmasked. As our proposed technique will not use perturbations as a basis creating scores, we believe that superpixels are an appropriate choice.

Now we have a method for generating individual pixels scores, and a method for creating regions to weight, we must have a method of combining the two. In the context of this chapter, combining is the weighting of independently produced superpixels with pixel-wise scores obtained from back-propagated gradients. The intention behind this is that the accuracy of pixel-wise scoring technique can be joined into regions that humans find more understandable.

The most similar technique to ours is XRAI. However, XRAI uses multiple sets of superpixels created at differing scales using the Felzenszwalb superpixel method [113]. XRAI then uses integrated gradients as a method for creating the individual pixel scores and combines with the superpixels using an iterative method. This iterative method builds an explanation by selecting from the multiple sets of superpixels those regions which have the highest attribution score within their boundaries. This is repeated until there are no superpixels left to add or the explanation is complete.

While the authors found this method produced good explanations, it is very computational inefficient as it requires multiple passes through the superpixels to allocate scores to regions of the image. Instead we propose to simply pool the superpixels found within a superpixel region and assign that pooled score to the region. This is a much more efficient method of creating an explanation, and found the pooling of scores to be robust in producing explanations that scored well in available metrics. We produce an explanation $E_i \in \mathbb{R}$ by weighting each superpixel region R_i (where R_i is the set of pixels belonging to the i^{th} superpixel), with the mean values of M found within that superpixel:

$$E_i = \frac{1}{|R_i|} \sum M \cap R_i. \tag{3.1}$$

We experimented with multiple methods of pooling the pixel scores into the superpixel but found that taking the mean was the optimal method. The justification and discussion of the other methods tried are given in Section 3.6.5.

3.4 Superpixels Designed for Explanations

Now that we have a method in place for the combining of the pixel scores superpixel regions, a question that should be asked is 'how well do these regions align to those deemed important by the model?'. Previous techniques

have either simply used off-the-shelf superpixels [13], or combined multiple superpixels created at varying scales [65] to find regions important to the model. However, to the best of our knowledge no one has previously tried to create superpixels that are designed to group regions by both appearance, and their importance to the networks prediction.

In this section, we propose the use of superpixels created solely for the purpose of giving accurate explanations. We explore how incorporating the pixel scores into the superpixel creation process can be achieved and the benefits of this. We begin by discussing the method of superpixel generation used and outlining how our proposed methods operate.

3.4.1 Definition of Superpixels

Many different methods for creating superpixels exist [114], each with their own advantages and disadvantages. For this work, we choose to use Simple Linear Iterative Clustering (SLIC) [115]. SLIC is a fast method that accurately adheres to object boundaries [114]. Experiments justifying SLIC over other superpixel methods (Felzenszwalb and Quickshift) can be found in Section 3.6.1.

The benefits of SLIC for use with our method is that it is easy to adjust the algorithm in order to incorporate features other than RGB values. For example SLIC has been adapted to use texture information [116] and depth information [117]. For our method we propose two ways of incorporating pixels scored from guided backpropagation. The first is to simply use the guided-backpropagated gradient by itself as an alternative to using the RGB image. These superpixels are simple to generate as SLIC treats the guided backpropagated gradients as a greyscale image. The second is to use a combination of the RGB image and the guided-backpropagated gradient to inform the creation of the superpixel. This is a more complicated proposition as it requires the SLIC algorithm to be adapted to combine both the RGB image and gradient distances. Throughout this thesis we will use the following method of referring to our proposed methods:

- **SWAG$_\mathbf{I}$:** SWAG using SLIC superpixels generated using only the values in the image.
- **SWAG$_\mathbf{G}$:** SWAG using SLIC superpixels generated using only the pixel scores generated using guided-backpropagation.

- **SWAG$_{\mathbf{I+G}}$:** SWAG using SLIC superpixels generated using both RGB values from the image and the pixel scores from guided-backpropagation.

3.4.1.1 Gradient-Based Superpixels

In this section, we introduce the SLIC algorithm and propose changes to it that allow it to better create superpixels that line up with both discriminative regions in the image, and those used by the network. We first begin by examining how SLIC works using a colour image. SLIC generates superpixels by clustering pixels in both colour and co-ordinate space: $[l_i, a_i, b_i, x_i, y_i]$, where l, a and b represents the CIELAB colour space [118], and x, y are pixel co-ordinates. SLIC proceeds to cluster these to produce cluster centres C_i. Superpixels are allowed to expand or contract within a limited range, in the original SLIC algorithm this is fixed at $2w_s$ from the cluster centre point. Here, where $w_s = \sqrt{N/K}$, N is the number of pixels in the image, and K is the desired number of superpixels. To determine whether a pixel (position j) belongs to a given superpixel, its distance to the spatial position or RGB value of the superpixel (at position i) is measured. Here, distance is defined as a combination of both colour distance d_c, and spatial distance d_s:

$$d_c = \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2}, \tag{3.2}$$

$$d_s = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}. \tag{3.3}$$

These distances are then combined to give a single distance value D' for each pixel within a superpixel:

$$D' = \sqrt{\left(\frac{d_c}{w_c}\right)^2 + \left(\frac{d_s}{w_s}\right)^2}. \tag{3.4}$$

Due to the differing scales of d_c and d_s, (determined by the maximum values of the CIELAB colour space and superpixel size respectively) a scaling component is used for each. For scaling colour distances, a value w_c is used. When this is large, priority is given to the spatial component, and when it is small, priority is given to the colour distance. The SLIC paper [115] uses a w_c value of 10. The authors chose this value as they found using a constant value in the range 0 to 40 made the superpixel creation process more controllable, using 10 as

their default. Spatial distance is scaled by w_s which seeks to maintain the grid like structure of the superpixels. Clustering proceeds iteratively as in k-means clustering.

Superpixels are designed to adhere to boundaries within an image which makes them useful as a starting point for CNN explainability methods [13]. However, by confining a superpixel method to only taking into account the colour space and distance when generating superpixels, we are potentially creating superpixels in a way that does not lead to the production of the most accurate explanations. For example, this process could be splitting an important region of an image across superpixels, when it may be beneficial to have it represented by a single one. We, therefore, propose a method of incorporating a gradient component into the SLIC algorithm. To begin with, we introduce a gradient component g to the initial superpixel description vector: $C_i = [l_i, a_i, b_i, x_i, y_i, g_i]^T$. Here g is a pixel within our gradient-based explanation M that provides a single score for each pixel. Here, M is scaled between $[0, 100]$ to match the range of LAB values. To compute the distance between pixels and the superpixel centre d_g, as with the spatial and colour distances, we calculate the Euclidean distance: $d_g = \sqrt{(g_j - g_i)^2}$. Following this, we alter the distance function D' to incorporate d_g:

$$D' = \sqrt{\left(\frac{d_c}{w_c}\right)^2 + \left(\frac{d_s}{w_s}\right)^2 + \left(\frac{d_g}{w_g}\right)^2}. \tag{3.5}$$

We also introduce a new parameter w_g that allows us to control the weighting of the newly introduced gradient element. These give superpixels that are created by combining both the image and gradient, by removing the d_c component (i.e. the image input) we are able to produce superpixels using only the gradient:

$$D' = \sqrt{\left(\frac{d_s}{w_s}\right)^2 + \left(\frac{d_g}{w_g}\right)^2}. \tag{3.6}$$

3.5 Metric Implementation

As discussed in the literature review, a number of metrics have been proposed that allow us to measure how well an explanation is performing. In particular we will use metrics that explore the following areas: local accuracy,

global accuracy, weak-localisations, interpretability, and efficiency. In this section, we discuss how these metrics are implemented throughout this chapter.

3.5.1 Local Accuracy

Measuring the local accuracy is an important, and common, way of understanding how well an explanation is performing. The local accuracy shows us how accurate the explanation is at highlighting regions of an input image that are most discriminative to the model. The better an explanation, the better the regions of the image that scored highly will align to the regions used by the model.

As we outlined in the literature review, there exist multiple methodologies for calculating the local accuracy. For this chapter, we chose the deletion metric introduced by Petsiuk *et al.* [9] metric for two reasons. The first is that it has been shown to be a popular technique amongst the community [72, 119], and the second is that it has a number of common traits with the global accuracy metric we use (discussed further in the following section). In this chapter, we will not perform insertion experiments, instead using the BAM metric as a measure of interpretability.

In the deletion experiment, every pixel within the input image is assigned a score from the explanation. For techniques such as LIME or XRAI that use superpixels, all pixels within a superpixel are assigned the same value. Pixels are then iteratively removed from the original image in order of importance, most important first. Here, removing a pixel from the image means setting it to the 0 (the mean colour value of the channel). As pixels are removed from the image, it is returned to the model for evaluation and the softmax score of the initially predicted class recorded. At each new evaluation it is expected that the removed pixels will damage the models ability to correctly classify the image. Striking the correct balance between how many pixels to remove for each step is an important decision. If too few are removed at each step then the process will be hugely inefficient. If too many then the results will be too coarse to properly interpret. We follow the number of steps taken by the original authors in their released code [120]. Here, the authors use 28 steps giving a removal amount of 1,792 pixels at each iteration.

When the softmax scores are plotted against each iteration taken, we get a chart showing how the softmax of the target class decreases as pixels are

removed. Ideally the better an explanation method, the faster the softmax score should drop. To obtain a single score, the Area Under Curve (AUC) value is taken. A lower AUC value indicates that the explanation is better at assigning scores to pixels.

We obtain results for a dataset by passing through the entire validation set and taking the mean of all the AUC values.

3.5.2 Global Accuracy

The second aspect that we aim to understand is the global accuracy of an explanation. Where local accuracy seeks to determine how well an explanation can score the regions of an image important to a models prediction, the global accuracy is a measure of how well the explanation finds all regions of the image that contains elements of the object that the model could learn from.

To measure the global accuracy, we need to be able to measure how well an explanation can find all areas of the image that may be useful to the model for describing the class. To do this we use the metric proposed by Hooker *et al.* [102], called Remove and Retrain (ROAR). This is a method of measuring global accuracy by retraining a model once a set amount of pixels have been removed from all images in a dataset. This is achieved by having multiple versions of a dataset with differing amount of pixels removed, each with a correspondingly trained model.

To create these datasets, we begin with a trained model. Using a chosen explanation technique, we create explanations for every image in both training and validations sets. Using these explanations, pixels are removed at the corresponding percentages: [0, 10, 30, 50, 70, 90]. At each removal step, a new training and validation set is created. This training data is then used to train a new model and the model accuracy is assessed using the new validation set. The better an explanation is at highlighting regions of an image that contain features of the object, the quicker the accuracy should drop.

Due to the stochastic nature of training models, this process is repeated 5 times for every explanation method tested. This means that for every explanation technique tested, 30 models are trained, each requiring a separate dataset of training and validation images with the correct percentage of pixels removed.

ROAR shares some similar properties with the previous local accuracy technique. They both remove regions of importance first, and the method of removal in both cases is to set the regions to 0. When plotted, they both produce similar looking graphs where the aim is to cause the softmax score or model accuracy to drop as quickly as possible. In the original implementation of the ROAR technique, the authors did not assign a single value to each set of results, relying on visual inspection instead. We, therefore, follow the local accuracy method and assign an AUC value to each set of results allowing us to better compare results between techniques.

3.5.3 Weak-Localisation

Localisation metrics aim to identify how well an explanation technique is able to weakly localise an object. That is, how well can an explanation localise an object without implicit supervision at training time. The weak-localisation task takes an explanation and sweeps through a range of thresholds in such a way that a bounding box can be drawn that incorporates the explanation left after thresholding. As per Fong and Vedaldi [11], we report results using three simple thresholding methods. First, we threshold the pixel value (scaled to between 0 and 1) in steps $[0 : 0.05 : 0.95]$. Second, we threshold using the mean intensity of the heatmap (μ_I) by taking our threshold $\alpha \in [0 : 0.5 : 10]$ and multiplying by the mean to form $\alpha\mu_I$. Finally, we threshold using the energy of the heatmap in such a way that the most salient region of the heatmap covers a defined percentage of the energy, where $\alpha \in [0 : 5 : 95]$. An example of thresholding using the pixel value is shown in Figure 3.4. Localisation error is calculated using the Intersection over Union (IOU), where an overlap greater than 50% is counted as correct. We only use ImageNet to perform this experiment as it contains ground truth bounding boxes that are vital for this technique to work correctly.

3.5.4 Attribution Accuracy

While methods of generating explanations for CNNs has been an active area of research for a number of years, only recently have attempts been made to understand how well the attributions assigned by an explanation align

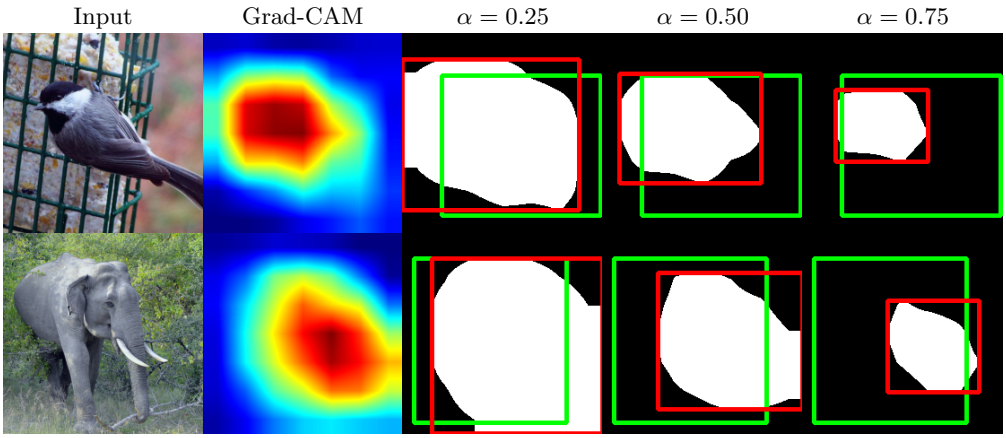

FIGURE 3.4: Examples of Grad-CAM explanations thresholded using the pixel value. The explanation is scaled between 0 and 1. Green is the ground truth bounding box, whilst red is the bounding box based on the thresholded explanation.

against a known ground truth. This is separate to accuracy, which aims to evaluate the correctness of the explanation, instead aiming to capture the mismatch between an explanation and the features known to be important to the model. Yang and Kim [76] introduced Benchmarking Attribution Methods (BAM), a dataset and associated methods used to discover an explanations false positives. These are the features of an image that are attributed high importance by an explanation, but actually have no importance to the model. BAM does this by introducing a composite dataset consisting of two separate datasets. These are scene images from MiniPlaces [121], and objects extracted from MS COCO [122]. The dataset uses 10 object classes from MS COCO (backpack, bird, dog, elephant, kite, pizza, stop sign, toilet, truck, zebra) and 10 scenes from MiniPlaces (bamboo forest, bedroom, bowling alley, bus interior, cockpit, corn field, laundromat, runway, ski slope, track/outdoor). These extracted objects are layered onto the scene images to create the new dataset. Examples from this dataset are shown in Figure 3.5. By compositing an object into a scene, it allows us to know for certain where the pixels are that the network should be using to predict the object, and the regions it should not.

Alongside the BAM dataset, the authors also define a method of computing the average attribution that an explanation gives to a specific region of an image. In this case, as seen in the BAM dataset, these regions, c, are either

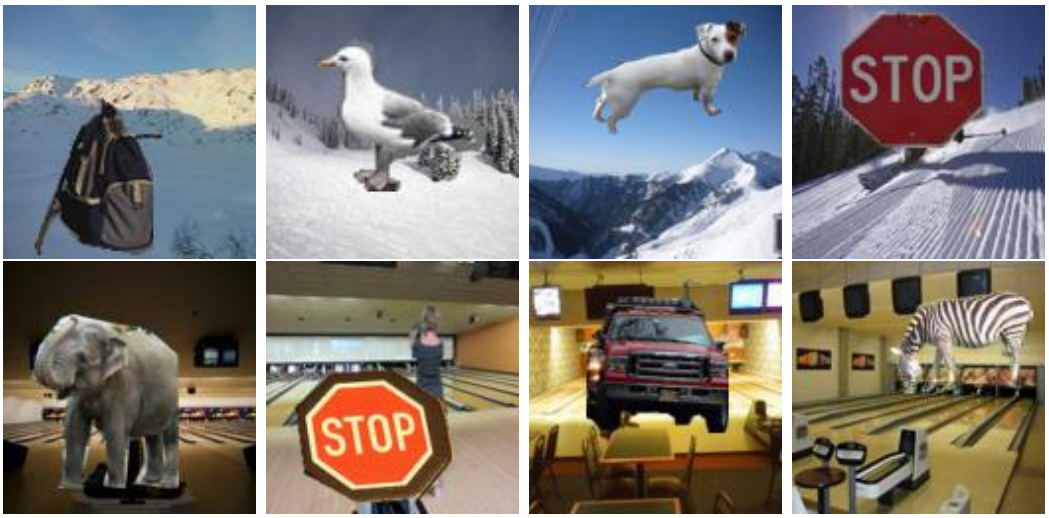

FIGURE 3.5: Examples of the BAM dataset. The top row is the ski slope scene, while the bottom row is the bowling alley scene. Both scenes have objects pasted in the foreground.

objects or scenes. The average attribution of pixels within a regions is given as:

$$g_c(f, x) = \frac{1}{\sum I_c} \sum e(f, x) \bigodot I_c. \tag{3.7}$$

Here, f is a model, x is an input image, and e is the explanation for image x using model f, whose values have been normalised to $[0, 1]$. I_c is a binary mask where pixels inside a region c have a value of 1 and those outside 0. Where g_c is applicable for single images at a time, concept attribution is further defined as the average of g_c over those inputs which are classified correctly. This is called G_c:

$$G_c(f, X) = \frac{1}{X_{corr}} \sum_{x \in X_{corr}} g_c(f, x). \tag{3.8}$$

X is a dataset of images and X_{corr} is the set of images from the dataset that are correctly classified.

The model contrast score (MCS) is a method proposed by Yang and Kim using the BAM dataset. MCS allows us to measure how well an explanation method provides attribution to regions of an image that are known to be used for prediction of an objects presence. This is done using two models, each trained on either the objects in the BAM dataset, or the background scenes.

The MCS is therefore the difference in attributions given to either objects or scenes by each model. More concretely:

$$MCS = G_c\left(f_o, X_{corr}\right) - G_c\left(f_s, X_{corr}\right) \tag{3.9}$$

Here f_o and f_s correspond to the model trained for objects and scenes respectively.

3.5.5 Efficiency

A question raised by Doshi-Velez and Kim [123], when discussing methods of generating explanations, is the time constraints that surround them. As explanations can be used in a variety of different environments the need to understand the computational costs of generating an explanation is important. For example, the requirement for a doctor sat with a patient needing an explanation in a timely manner is very different to a researcher with less strict time constraints.

In the literature, there are two predominant methods of measuring the computational efficiency of a deep learning method, we can either measure the required floating point operations (FLOPS), or the "wall clock" time, that is the number of seconds taken to compute an explanation. This is averaged over a large number of explanations. Of the two, measuring the "wall clock" time is much more common when discussing explanations. To run this experiment we take the first 1,000 images from the validation set and generate explanations for each. We measure the time taken in seconds to compute the explanation. We then take the mean time taken to create the explanation as our indicator of the efficiency of the method.

3.6 Superpixel Optimisation

With our two complementary methods, defined above, there are a number of important choices that have to be made in order to allow optimal performance. In this section, we begin by justifying our use of SLIC as our choice of superpixel method. We then discuss the choice of gradient-based pixel scoring method that is used as the underlying weighting method for SWAG. These gradients are the used when we create our gradient based superpixels.

Having determined these crucial elements, we then look to determine the optimal choice of colour and gradient weights (w_c and w_g respectively) for use with the combined gradient and image superpixel method $\text{SWAG}_{\text{I+G}}$. Finally, we discuss the choice of how many superpixels we determine to be optimal for use in SWAG.

3.6.1 Justification of Superpixel Method Choice

The choice of method that generates the superpixels is a core element of SWAG. In this section, we compare three techniques: Simple Linear Iterative Clustering (SLIC), Quickshift [124], and Felzenszwalb [113].

We choose these techniques to compare against as they are currently used in alternative explanation methods. Felzenszwalb is used as the underlying superpixel generation technique in XRAI [65], while Quickshift is the default method for LIME [13]. For all techniques, we use scikit-image to generate the superpixels. As the methods all have varying parameters that affect the way superpixels are generated, we only change one parameter for each technique, and keep the remaining parameters as the defaults. For SLIC, we change the number of segments; for Quickshift, the kernel size; and for Felzenszwalb the scale. These parameters are the ones that have the most impact on the number of superpixels generated.

We look at three aspects of the superpixel generation techniques. The first is the ability for each technique to form superpixels that naturally align well to regions that are useful for explanations. We do this using the local accuracy AUC metric and our SWAG technique. The second aspect we investigate is how consistent the superpixels are in terms of superpixel sizes. Ideally we would like to have control over the superpixels generated, and the ability to create consistently sized superpixels for explanation may be beneficial. In contrast we wish to avoid techniques which could produce very small or very large superpixels within the same set of superpixels. Finally, we observe the computational costs for each method as it is important to not pick a technique that is unable to scale when multiple images require explanations.

3.6.1.1 Natural Alignment

We measure the natural ability for a technique to align with explainable regions. Although it is difficult to compare methods directly, as they all

TABLE 3.1: AUC results for the local accuracy experiment. Lower is better.

Method	Mean Superpix	VGG16	ResNet50
Felzenszwalb(50)	342	0.090	0.121
Felzenszwalb(100)	215	0.096	0.127
Felzenszwalb(200)	122	0.103	0.134
Quickshift(4)	246	0.111	0.142
Quickshift(3)	352	0.106	0.137
SLIC(200)	153	0.099	0.128
SLIC(300)	235	0.092	0.119
SLIC(400)	328	0.086	0.113

produce superpixels with differing shapes and sizes, we compute the mean number of superpixels and display it alongside the local accuracy results. These results are shown in Table 3.1. Quickshift seems to perform poorly across the board, while SLIC performs well when compared to similar sized superpixels (i.e SLIC(400) vs FZ(50)).

3.6.1.2 Superpixel Consistency

To allow for comparison between the ways the differing methods size the superpixels, we record the mean number of superpixels, along with the minimum and maximum number of superpixels. The range in the number of superpixels is striking with both Felzenszwalb and Quickshift having a large range. SLIC on the other hand is much more constrained in its range as they are controlled by the initial seeding of superpixel start points. SLIC begins by initialising a grid, and growing the superpixels from that point.

The number of superpixels produced by SLIC is, therefore, capped by the starting grid which allows for some consistency and control when compared to the Felzenszwalb and Quickshift techniques.

3.6.1.3 Computational Efficiency

The computational efficiency of a method is an important factor to consider when generating explanations [5]. It is particularly important for tasks which take a long time to compute such as action recognition. We run each of the

TABLE 3.2: Showing how the choice of superpixel methods and parameters effects the number of superpixels produced.

Method	Min Superpix	Mean Superpix	Max Superpix
Felzenszwalb(50)	16	342	1025
Felzenszwalb(100)	7	215	920
Felzenszwalb(200)	4	122	718
Quickshift(4)	40	246	3688
Quickshift(3)	64	352	4165
SLIC(200)	5	153	199
SLIC(300)	10	235	289
SLIC(400)	28	328	401

TABLE 3.3: A comparison of the computational efficiency for each method. This is the average time to computer a set of superpixels for a 224×224 image

Method	Time Taken (seconds)
Felzenszwalb(50)	0.05
Felzenszwalb(100)	0.05
Felzenszwalb(200)	0.05
Quickshift(4)	0.78
Quickshift(3)	0.47
SLIC(200)	0.06
SLIC(300)	0.06
SLIC(400)	0.06

superpixel methods 1,000 times and compute the mean time taken to generate a set of superpixels. These results are shown in Table 3.3. From these results we see that Felzenszwalb and SLIC and take fairly similar times to compute. Quickshift, however, takes a much longer time per image to compute a set of superpixels.

3.6.1.4 Conclusion

Each of the three techniques we have tested have their own strengths and weaknesses. Of the three Quickshift is the slowest to compute, has by far the

largest range in terms of superpixels per image, and seems to have the worst natural alignment. SLIC and Felzenszwalb are similar in the time taken to compute and their natural alignment to explainable regions. However, SLIC is much more predictable with regards to the number of superpixels being created. SLIC has an upper bound on the number of superpixels able to be created due to the use of a grid as a starting point. Felzenszwalb does not have this constraint and therefore produces superpixels with widely varying counts between images.

Based on these observations, we feel justified in our use of the SLIC method as a means of creating the underlying superpixels for our method.

3.6.2 Choice of Superpixel Count

At the heart of our technique is the use of SLIC superpixels to create a foundation with which to build explanations upon. One of our prime motivations is to create a technique that bridges the gap between fine and coarse explanations, therefore the number of superpixels that we use is important. In practice, the number of superpixels presents a trade-off between the desired granularity of the explanation and the spatial accuracy (large superpixels can extend beyond the boundaries of a significant object, whereas small superpixels cause explanations to become less human-interpretable). We have shown previously that when scores are assigned to individual pixels they produce explanations with better accuracies. It makes sense intuitively that as we increase the number of superpixels used, the local accuracy should also improve. If we continued to a theoretical point where every superpixel corresponded to a single pixel, our explanation would be identical to the gradient method we have used to weight the superpixels.

We found that using an approximate starting value of 300 pixels offered a good balance between fine and coarse explanations. For clarity, we chose this value based on visual assessment rather than running local accuracy metrics on multiple different superpixel values. In later chapters we investigate a method of finding the optimal superpixel count value. Using 300 superpixels for our SWAG method, we show how our proposed method inserts into the chart from Figure 3.3. We show this in Figure 3.6

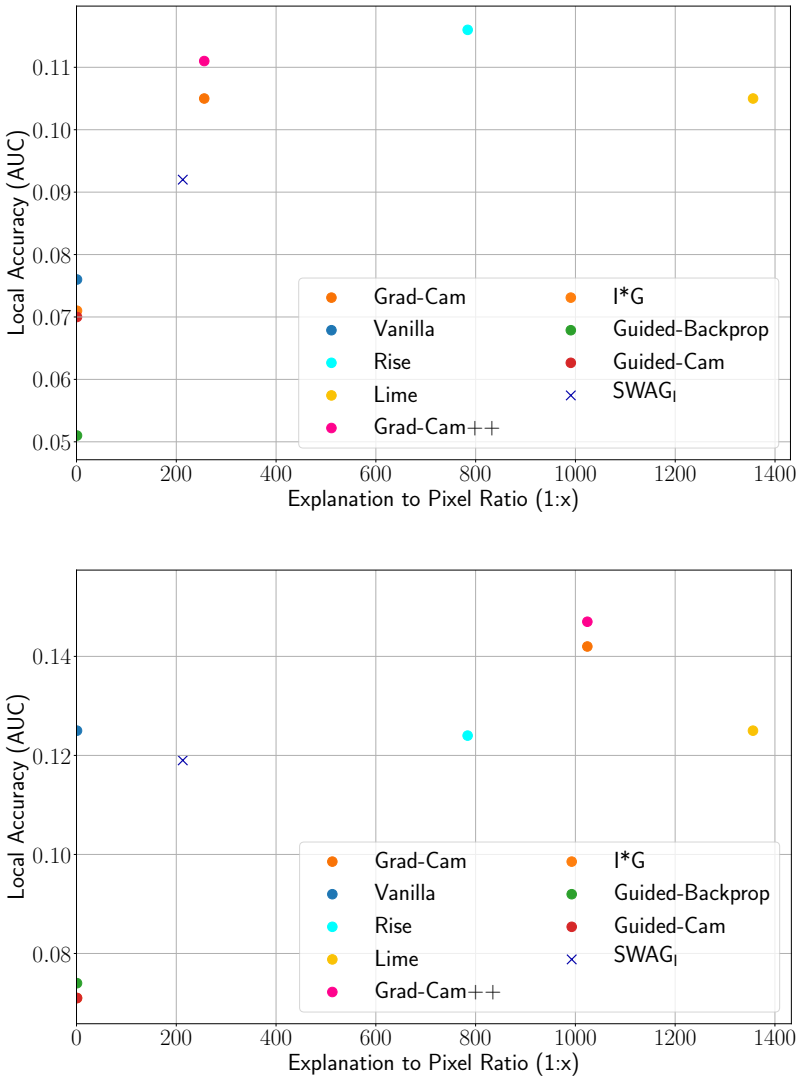

FIGURE 3.6: A comparison between the local accuracy score (lower is better) and the score to pixel ratio using ImageNet. Top: VGG16 Bottom: ResNet50.

3.6.3 Choice of Attribution Method

Finding the optimal method of creating gradients used by our techniques is of paramount importance to the correct generation of explanations. As discussed in literature review (Chapter 2), there are a large range of methods for the generation of back-propagated gradients. We constrain ourselves to only selecting from methods that are simple to implement and computationally efficient to run. With this criteria, we investigate the use of: vanilla-backprop, guided-backprop and input$\odot$gradients. As noted earlier XRAI uses integrated

gradients as it's attribution method. While this method is simple to implement, we feel that this does not meet our criteria for being efficient as it requires multiple calculations of the gradient to create. For example XRAI, uses 100 passes through the network to create it's pixel scores for every image. We provide an outline of each method we tested:

- **Vanilla-Backprop**: Gradients are backpropagated from the target classes softmax score back to the input image. The maximum of the absolute values are taken over the three channels to produce a single 224×224 image.
- **Input⊙Gradients**: Gradients are produced as with the previous vanilla technique but prior to taking the maximum of the absolute values the gradients are multiplied with the RGB values in the image.
- **Guided-Backprop**:When the input image is passed through the network the activations are stored and during the backward pass only gradients that correspond to a positive activation value are allowed through.

As a sanity check, we also test using both uniformly random noise and Sobel to weight the superpixels. In order to understand how well each of these methods perform with SWAG, we use the local-accuracy experiment devised by Petsiuk *et al.* [9]. The results are shown in Table 3.4. These results give us confidence in our choice of attribution method for two reasons. The first is that guided-backpropagation beats the alternative methods of vanilla gradients and input⊙gradients. This suggests that this is an appropriate method that gives useful attributions to our superpixels. Secondly we are able to beat the baselines. This is important, as Hooker *et al.* [102] showed, in its pixel form guided backpropagation was unable to perform better than the baselines for global accuracy.

3.6.4 Gradient Sanity Check

Previous work has discussed how guided-backpropogation does not pass sanity checks such as randomisation of weights [101]. By only allowing positive gradients to backpropagate through regions of the image that had a corresponding ReLU activation, a visual similarity occurs when the weights are randomised. This is likely due to the strong involvement of the positive-only activation map component which forces the explanation to look at the edges of an image. This

TABLE 3.4: AUC results for the local accuracy experiment. Lower is better.

Attribution Method	VGG16	ResNet50
Random Noise	0.170	0.210
Sobel	0.154	0.188
Vanilla-Backprop	0.134	0.190
Guided-Backprop	**0.092**	**0.119**
Input⊙Vanilla-Backprop	0.118	0.165

TABLE 3.5: AUC results for our sanity check for guided backpropagation for use with SWAG. Lower is better.

Attribution Method	VGG16	ResNet50
Guided Backprop (Predicted Class)	**0.092**	**0.119**
Guided Backprop (Random Class)	0.104	0.131

is also what gives guided backpropagation it's distinctive and clean look. However, we've shown that guided backpropogation works well as an attribution method for our technique. To ensure that we actually explaining the reasons behind how the network is discriminating between classes, we propose a sanity check. Guided-backpropogation works by backpropogating from the predicted classes softmax score to the input image. By randomly assigning the target class instead of using the predicted one, we would like to see a drop in local accuracy as the pixel scores no longer align to the regions used by the network for discrimination. We show the results for this in Table 3.5, and an example of the explanations produced in Figure 3.7. It is noticeable that even when the class has been randomised, the results are still good compared to our baselines and other techniques from the previous experiment. This is again likely due to the activation map component which will always force the areas of the image that caused an activation to score highly, regardless of the gradient component.

3.6.5 Choice of Pooling Method

In this section, we quantitatively justify the method chosen for pooling the gradients into superpixels. In Equation 3.1, we proposed taking the mean of the

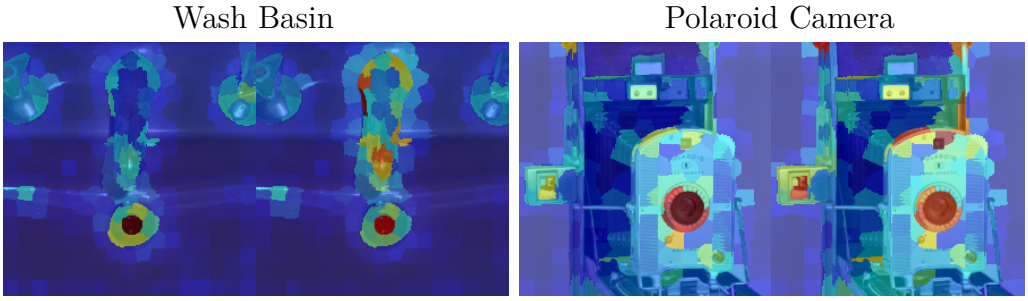

FIGURE 3.7: Examples of SWAG using guided backpropagation with both the predicted and random class. Note that the superpixel regions are identical for the predicted and random targets.

gradients within a superpixel as the importance score for that region. However, there are multiple other methods that could have been chosen. Therefore, we run the local accuracy experiment using guided backpropagation as SWAG's gradient method, with a number of substitutions for scoring using the mean value. These alternatives are:

- Minimum.
- Maximum.
- $\ell 1$-norm.
- $\ell 2$-norm.
- Variance.
- Standard deviation.
- Sum of the gradients.

The results from the local accuracy experiment for each of these methods can be found in Table 3.6. Here we see that taking the mean of the gradients gives the best local accuracy performance overall. However, it is interesting to note that taking the minimum value performs equally well for ResNet50. A potential oversight in these results was investigating the use of the median value. Although it is likely it would've performed poorly as a characteristic of the guided-gradients we used is that they are typically sparse with only a few high scoring pixels. This may not have allowed the median value to work well.

TABLE 3.6: AUC results for the local accuracy experiment. Lower is better. Here we see that of the pooling methods tried, taking the mean works the best.

Method	VGG16	ResNet50
ℓ1-norm	0.099	0.124
ℓ2-norm	0.096	0.122
Maximum	0.100	0.125
Mean	**0.092**	**0.119**
Minimum	0.095	**0.119**
Standard Deviation	0.097	0.124
Sum	0.099	0.124
Variance	0.097	0.124

3.6.6 Choice of Weights for SWAG$_{\text{I+G}}$

Our SWAG$_{\text{I+G}}$ method introduces a new weight (w_g) that allows us to control the influence that the gradients have upon the generation of superpixels. In addition to the gradient weight, there is also a spatial weight (w_s) and a colour space weight (w_c) that we must optimise for. As with the original SLIC algorithm, we leave $w_s = \sqrt{N/K}$, where N is the number of pixels in the image, and K is the desired desired number of superpixels. This controls how far the superpixel can spread from its initial centre point. This allows us to maintain superpixels that have a degree of compactness to them rather than stretching out over the image. However, w_c and w_g require further investigation as they must be balanced carefully so as to allow the superpixel to capture both the salient region of the colour space as well the regions of the image deemed important to the network by the gradients. We, again, use the local-accuracy metric and perform a grid search over the weight values. We constrain our search space to between the values of $w_g \in [4, 20]$ and $w_c \in [4, 20]$, incrementing in steps of 2. We chose these values as they are centred around the default (in the original SLIC implementation) w_c value of 10. As we have scaled the gradients to have approximately the same range as the colour space values we should be able to apply the same rage of values to each weight. Here, values of 5 and 20 represents a doubling or halving of the channels influence respectively. We choose not to use values that would diminish the influence of the colour channel beyond half. Doing so could begin to allow the superpixels to detach

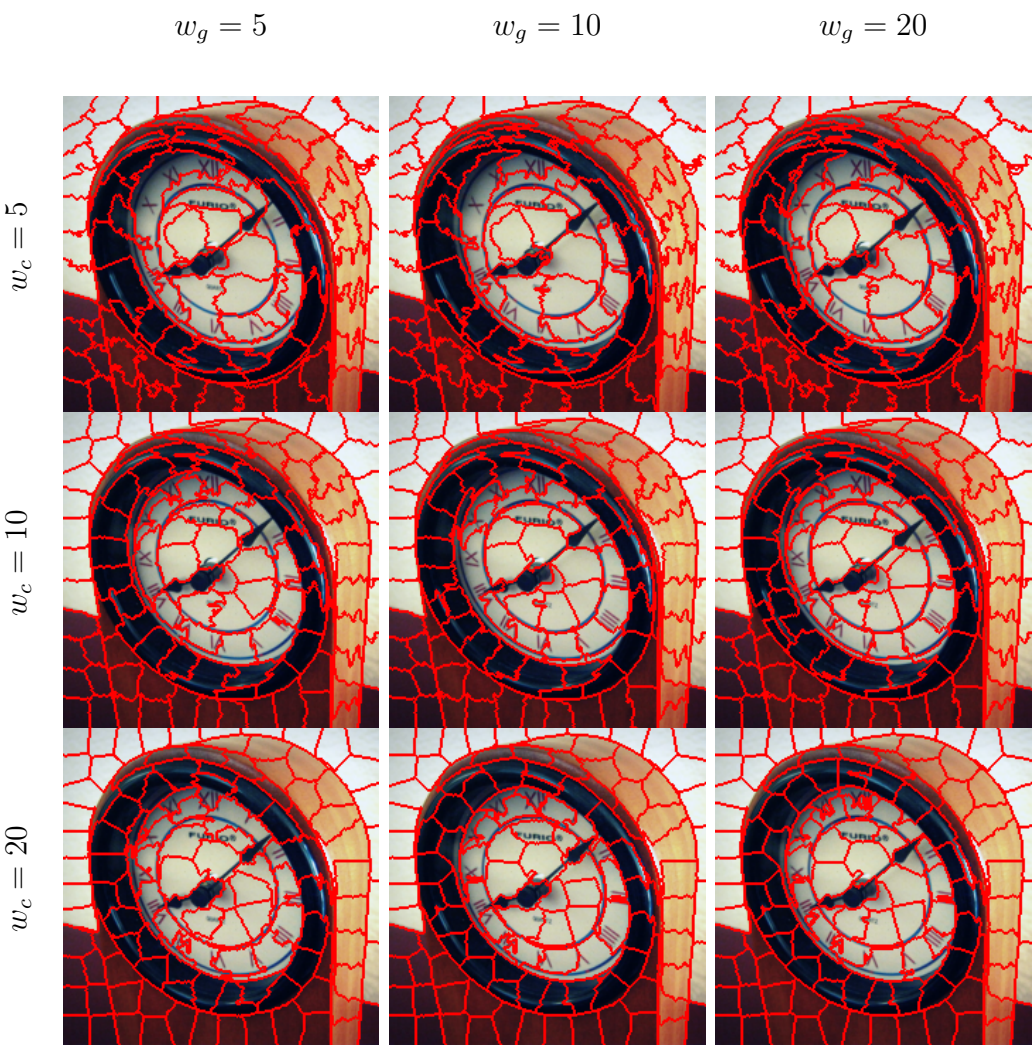

FIGURE 3.8: Comparison showing how altering the balance of w_g and w_c effects the generation of superpixels. For clarity we show approximately 150 superpixels.

themselves from the underlying image which would remove one of the benefits of using superpixels, namely that they root themselves to human interpretable regions of an image (i.e. the beak of a bird, or the ball in a sports game). An example of the superpixels created using these search values can be seen in Figure 3.8.

From these qualitative results we can see that when both w_c and w_g approach 20, the superpixels begin to become squarer as the influence of the spatial component is increased, however note that the colour boundaries are still adhered to. In contrast, as w_c is decreased the influence of the colour space becomes more apparent. For example, note the right side of the clock, as

the colour becomes more important the superpixels begin to differentiate the colour gradient up the wooden side. As w_g decreases, we begin to see a looser adherence to the colour components within the image and a greater ability to form around areas with high gradient value.

When we produce quantitative results using the local-accuracy technique, we begin to gain a more concrete understanding of how the choice of weights effects our explanation method. We again use the local accuracy metric to understand how altering the weights effects the ability of the superpixel boundaries to adhere to features important to the network. In Figure 3.9, we show the interplay between the two values when the other is fixed. It is interesting to note that the local accuracy continues to improve as the influence of the colour space component is diminished. For SWAG $_{\mathrm{I+G}}$, we aimed to maintain a balance between the superpixels created for the image ($\mathrm{SWAG_I}$), and those solely using the gradients ($\mathrm{SWAG_G}$). Allowing the influence of the colour component to diminish further would therefore encroach upon the explanations created using $\mathrm{SWAG_G}$.

3.6.7 Final Parameter Choices

For clarity, we present the final parameter choices for our technique:

- **Superpixel Method**: We use the *SLIC* method for generating superpixels as we found it to be both efficient to compute, and had a more reliable range of superpixel sizes compared to other methods.
- **Attribution Method**: *Guided backpropagation* was chosen as the attribution method of choice as we showed it to outperform other methods for the local deletion metric.
- **Pooling Method**: The *Mean* value of the attribution values falling into a superpixel was determined to the optimal method.
- **Initial Superpixel Count**: Using a count of *300* superpixels was determined to produce an explanation that lay between the fine and coarse explanations produced by existing techniques.

Using these parameters we will now being to run experiments using our SWAG method.

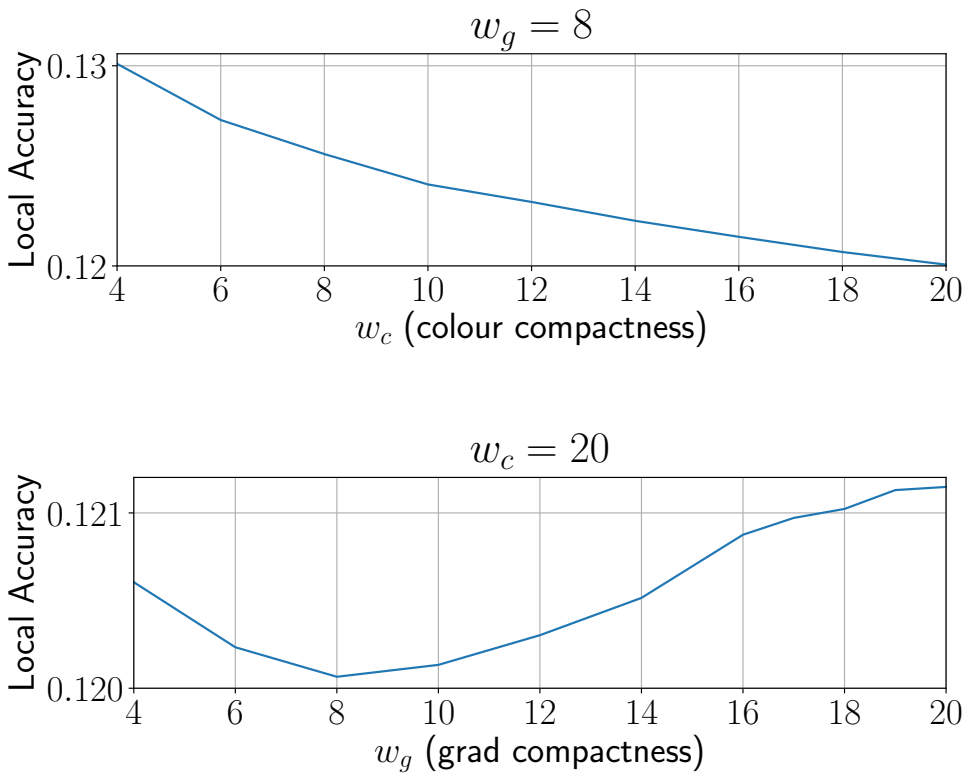

FIGURE 3.9: Grid search results showing the interplay between each parameter when we fix each at their best values.

3.7 Experiment Results

In this section, we present the results of the experiments that were outlined in Section 3.5. To recap, these are the local and global accuracy, weak-localisation, attribution accuracy, efficiency, and LIME superpixel substitution. We conduct the experiments to ascertain how well our SWAG_{I}, $\text{SWAG}_{\text{I+G}}$, and SWAG_{G} methods perform compared to a number of well known explanation techniques. In particular we compare against the following techniques: Grad-CAM, Grad-CAM++, LIME, RISE, and XRAI. Note that these are all region based techniques. We also show results for guided-backpropagation as this underpins our technique. To ensure fairness we run all experiments using the authors original implementations for LIME [125], XRAI [126], and RISE [120]. The only changes made are to utilise a PyTorch backend instead of a TensorFlow one when required.

FIGURE 3.10: An example of the Euclidean baselines used. Top: Centre point. Bottom: Random point

Baselines are often used to evaluate how well a technique performs, for example in the work by Hooker *et al.* [102], random noise and Sobel edge detection [127] are used as baselines to compare against various saliency map techniques. However, as we are explicitly comparing against region-based explanation approaches, we instead use baselines based on the Euclidean distance from a specific pixel. We use both a centre point Euclidean distance map (referred to as centre), as well as the Euclidean distance to a uniformly randomly chosen pixel (referred to as random). Examples of these are shown in Figure 3.10. We report results across multiple datasets: ImageNet [128], Caltech-UCSD Birds 200(CUB200) [129], Stanford Dogs [130], and Oxford Flowers 102 [131]. Excepting ImageNet, these are all fine-grained datasets consisting only of visually similar objects. This presents an additional challenge to existing explainability methods where discriminative features may occupy a small region of the image. All work is conducted with PyTorch [132] using pre-trained VGG16 [67] and ResNet50 [68] networks for ImageNet. These models were fine-tuned for the fine-grained datasets for 50 epochs with a learning rate of 0.001 for both VGG16 and ResNet50. Top-1 validation accuracies for VGG16 and ResNet50 respectively are: ImageNet (71.59%, 76.15%), CUB200 (82.22%, 85.42%), Stanford Dogs (79.60%, 85.09%), and Oxford Flowers (94.95%, 92.24%).

We hypothesise that for the metrics we have outlined in Section 3.5 our proposed SWAG method should outperform other comparable techniques. We

suggest this may be the outcome due to SWAGs ability to create finer explanations. For both the local and global accuracy metrics this should have the effect of allowing explanations to more precisely locate regions of importance. For the weak-localisation and BAM metric this is a less certain as activation based techniques such as Grad-CAM or Grad-CAM++ may benefit in their coarseness. This would allow the explanation to highlight the object completely which is beneficial to these techniques.

3.7.1 Qualitative Inspection of Results

In this section we show qualitative examples to allow for visual inspection. We select these images by generating an explanation every 50 images and then selecting representative examples. To ensure that we have not simply cherry picked examples, we also include further examples in the appendix. By only allowing selection from a smaller set of non hand-picked explanations, we hope that this gives some faith that these examples are representative. We will follow this selection process for all qualitative inspections in the following chapters. In Figure 3.11 and Figure 3.12, our method is compared to a number of other well known and comparable methods. These examples are computed using both the VGG16 and ResNet50 networks with images from the ImageNet validation set. From these images we see that both SWAG_{I}, $\text{SWAG}_{\text{I+G}}$, and SWAG_{G} provide much more concise explanations than other methods. In particular we see how coarse the two CAM methods are, especially for the ResNet50 architecture. For example, note the camera in Figure 3.11 where the lens seems to be the most important feature. Here, the CAM based methods give very coarse explanations, essentially highlighting the entire front of the camera as important. LIME also fairs poorly in this example with the superpixels being too large to accurately represent the important regions. RISE gives visually more concise explanations as it has the benefit of having more regions to work with than LIME, but also has the additional power of being able to pass multiple variations through the model. XRAI is interesting as it manages to be both precise by highlighting small regions of the image, but also noisy in its visual appearance. The latter attribute is likely due to the use of integrated gradients which aims to assign non-zero values to the majority of the image. This in turn likely leads to the visual noise. This visual noise may cause

difficulty in a viewer discerning between important and not important image regions.

Compared to these methods, the precision of SWAG based methods can be seen, for example note the precision surrounding the body of the chainsaw or the plughole of the washbasin with only minimal highlighting of other regions of the image. Where there is visual noise in our method, for example the VGG16 version of the camera, it seems that other methods also highlight the noisy regions.

Further examples using images from the other datasets experimented with can be found in Appendix A.

3.7.2 Explanation Accuracy

In this section, we present the results for what are perhaps the most important metrics for an explanation technique, their accuracy. As we have previously outlined, this is divided into two differing ways of measuring the accuracy: local and global. Using the deletion metric by Petsiuk *et al.* [9] and ROAR by Hooker *et al.* [102] to measure the local and global accuracies respectively, we present the results for each.

3.7.2.1 Local Accuracy

The local accuracy experiment attempts to ascertain how well an explanation attributes importance to regions of the image. This is achieved by iteratively removing regions of the image as determined by each explanation technique. These are removed from most important region to least. In this metric, the intuition is that removing the regions important to the network's decision will force the network to alter its decision. Therefore, as the important regions of the image are removed, the softmax score will drop accordingly. A sharper drop will indicate that pixels important to the network are being removed more quickly, suggesting a better explanation. This local metric measures the area under the curve (AUC) as features are removed from the input image. A smaller AUC values indicates that its corresponding method is better able to attribute importance to the regions of the image most discriminative to the network. The results for all datasets and networks used can be found in Table 3.7. The corresponding plots for VGG16 and ResNet50 are shown in

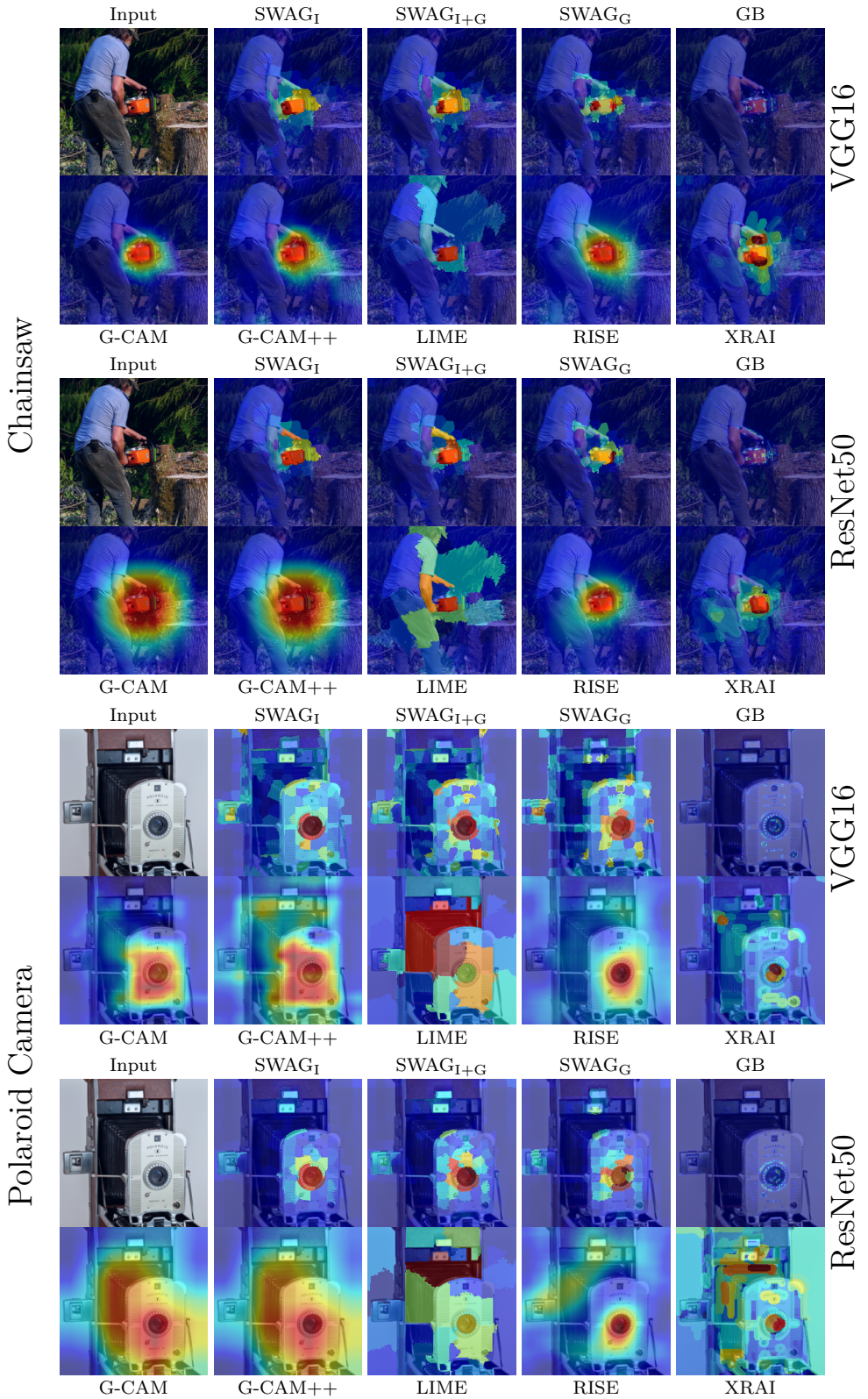

FIGURE 3.11: Qualitative comparison between methods using examples from ImageNet with ResNet50 and VGG16. Our SWAG methods provide finer explanations than comparable activation, and perturbation based methods.

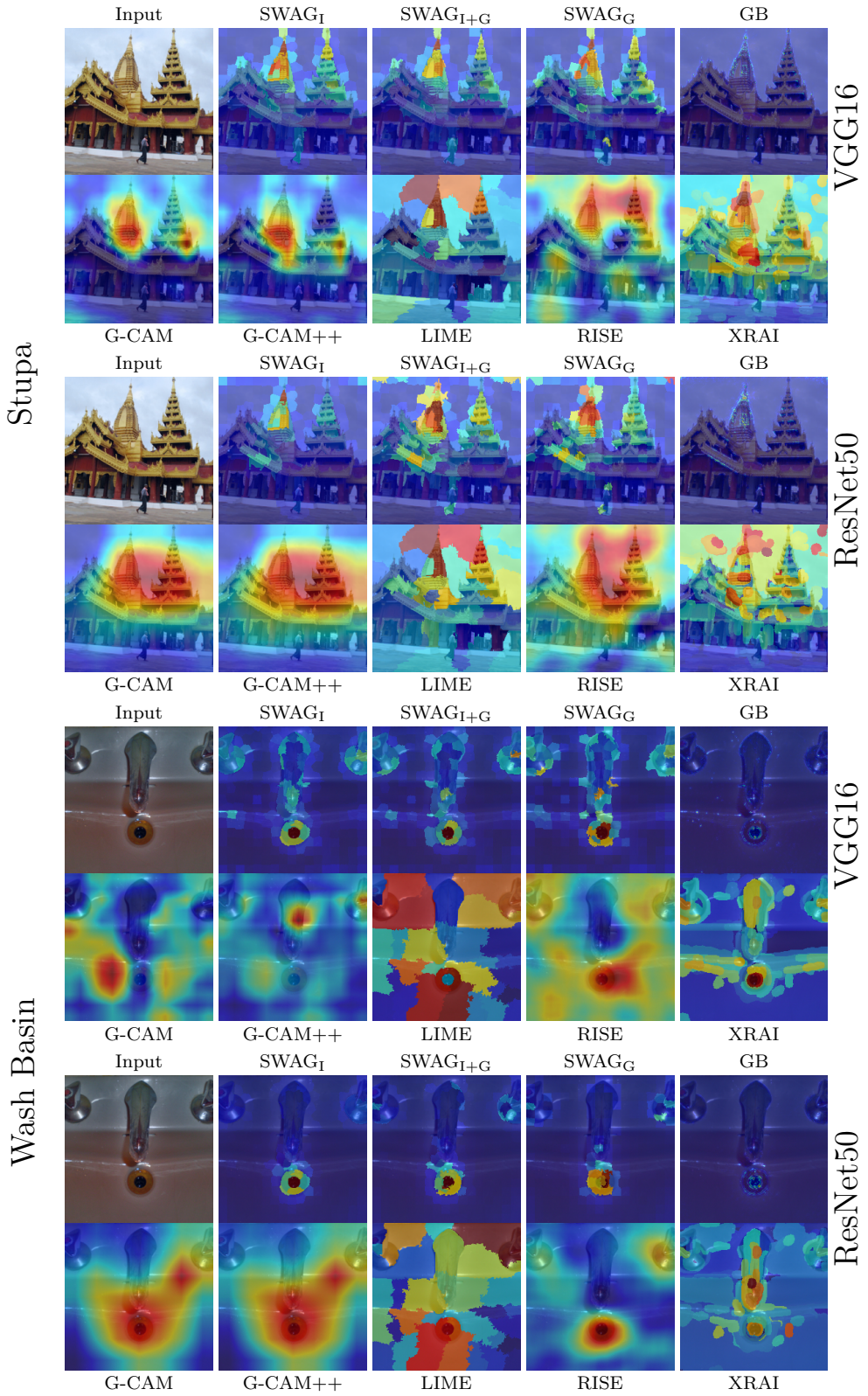

FIGURE 3.12: Qualitative comparison between methods using examples from ImageNet with ResNet50 and VGG16.

Figure 3.13 and Figure 3.14 respectively. Each figure contains standard views of the results for each dataset alongside charts showing a closer view of the bottom left corner.

From these results, we can see a number trends that are important to highlight. The first is the performance of the baselines. As expected, the random baseline performs poorly across all datasets and models. The centre point baseline is more interesting. On the whole it performs poorly too, outperforming only the random baseline. It is notable however that with the Oxford Flowers dataset it beats both Grad-CAM and RISE. This is likely due to the very central nature of the images within the dataset, where the centres of the flowers align to the centre of the image. This explains the centre baseline success as it simply removes the flower from the image during the earliest removal steps.

LIME performs admirably compared to the other perturbation technique, RISE. This is despite LIME being able to score far fewer regions ($\sim$ 37 to 64). LIME also outperforms all techniques for Stanford Dogs using the ResNet50 architecture. This seems to be because the size of the superpixels used by LIME match with the features present in the images of the dogs. Typically, LIME attributes the dog's head as the most important regions and assigns it to a single superpixel meaning. This results in the dogs head being removed in the first step of pixel removal, which in turn causes the models prediction abilities to decrease significantly.

It should also be pointed out how well guided backpropagation performs compared to all other techniques. While this is not surprising, as it is a pixel-based technique, the results are still strikingly better than all of the region based techniques. As the technique is able to apply scores at the pixel level, it can very quickly identify the important pixels and cause the network to crash. This is seen in Figures 3.13 and 3.14 showing how quickly the guided-backpropagation curve drops.

Finally, we move on to discussing our three SWAG methods. SWAG_I performs well, beating all of the previous region-based techniques apart from the previously mentioned LIME and Stanford Dogs combination. Adding a gradient element to the generation of the superpixels further improves upon these results with $\text{SWAG}_{\text{I+G}}$ surpassing SWAG_I. However, using the gradient alone to inform the superpixel process gives the best results of all. This suggests

that using superpixels built upon the gradient allows regions to be formed that are better suited to explaining how the image is used by the network to discriminate between classes.

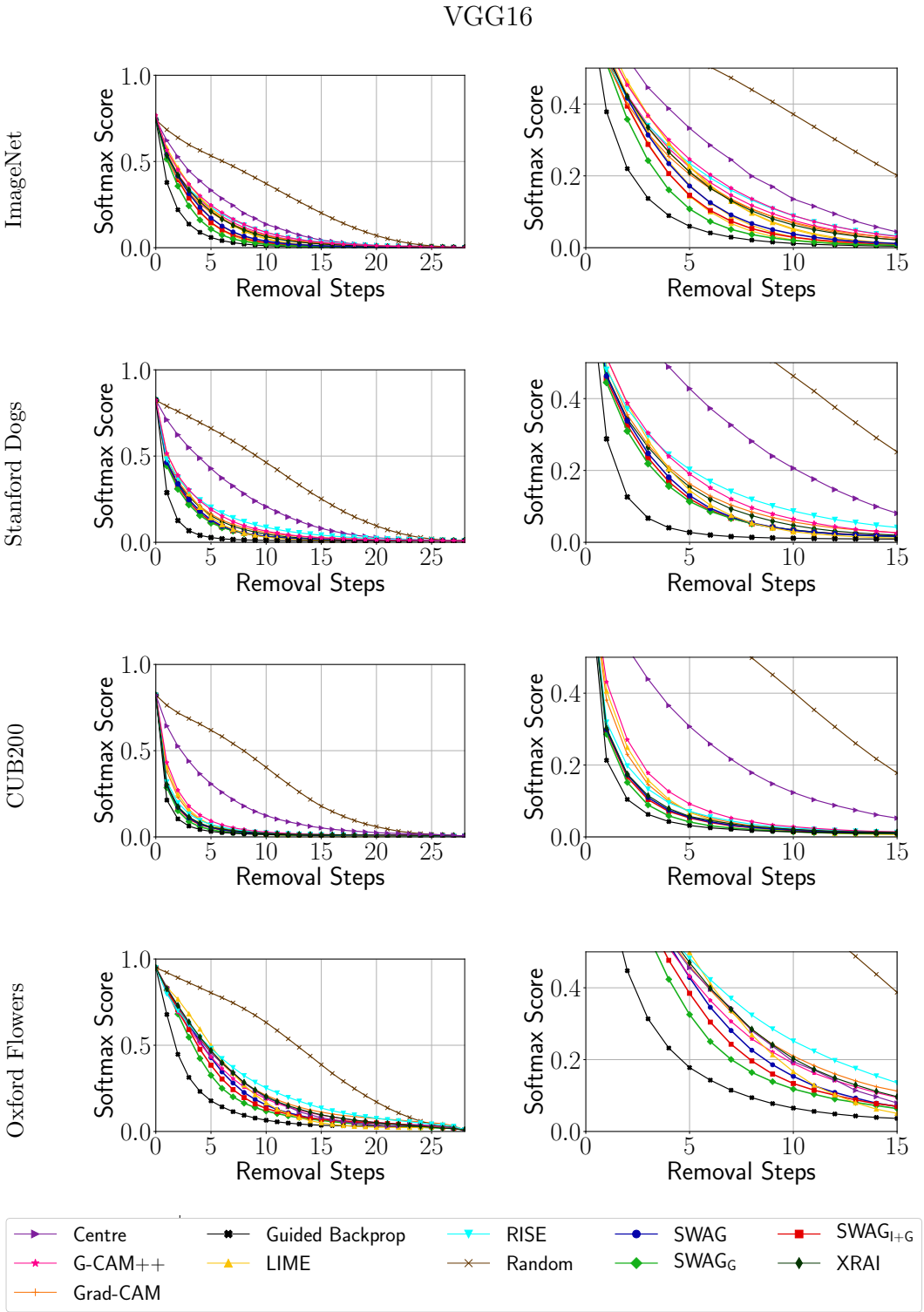

FIGURE 3.13: Local accuracy AUC charts for VGG16. Left column is the regular view, right is a zoomed in view of the bottom left corner.

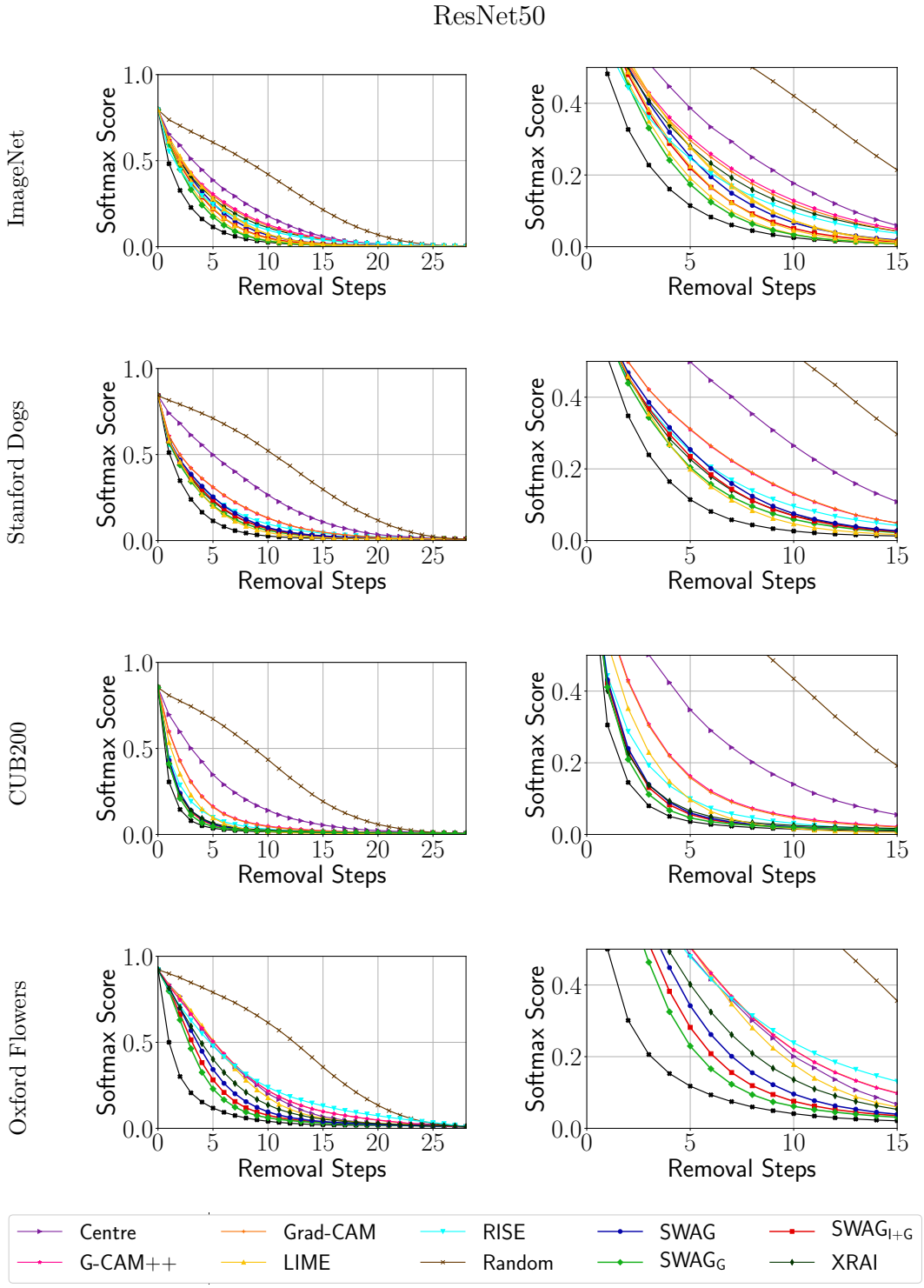

FIGURE 3.14: Local accuracy AUC charts for ResNet50. Left column is the regular view, right is a zoomed in view of the bottom left corner.

Table 3.7: Area under the curve for the local accuracy experiment. Lower is better.

	ImageNet		CUB200		Stanford Dogs		Flowers 102	
Method	VGG16	ResNet50	VGG16	ResNet50	VGG16	ResNet50	VGG16	ResNet50
Random	0.274	0.303	0.296	0.317	0.337	0.371	0.446	0.425
Centre	0.153	0.177	0.153	0.168	0.200	0.233	0.221	0.223
Grad-CAM	0.105	0.142	0.060	0.099	0.097	0.150	0.237	0.235
Grad-CAM ++	0.111	0.147	0.069	0.101	0.104	0.149	0.217	0.234
XGrad-CAM	0.105	0.142	0.058	0.099	0.099	0.150	0.239	0.235
LIME	0.105	0.125	0.059	0.074	0.087	**0.107**	0.214	0.218
RISE	0.116	0.124	0.057	0.072	0.113	0.129	0.250	0.244
XRAI	0.105	0.137	0.053	0.063	0.090	0.117	0.227	0.188
$SWAG_I$	0.092	0.119	0.051	0.062	0.083	0.123	0.206	0.168
$SWAG_{I+G}$	0.084	0.109	0.050	0.060	0.080	0.118	0.195	0.151
$SWAG_G$	**0.073**	**0.095**	**0.046**	**0.057**	**0.077**	0.110	**0.177**	**0.137**
Guided-Backprop	0.051	0.074	0.040	0.046	0.042	0.080	0.122	0.086

3.7.2.2 Global Accuracy

Having obtained results for the local accuracy, we now turn our attention to understanding how well our technique performs for the global accuracy metric. Where the local accuracy metric sought to discover how well an explanation was at finding regions of the image important to a specific decision by the network, the global accuracy metric seeks to know how well an explanation can find all features representative of the class.

Using the ROAR technique outlined earlier is the primary way of finding the global accuracy. The downside to using this technique is that it is very inefficient as it requires multiple models and datasets to compute. To this end, we were only able to calculate results for two datasets (CUB200 and Stanford Dogs) using ResNet50. We also found that the combination of a computationally inefficient test metric combined with an inefficient explanation method caused problems. In the end we were unable to test RISE and XRAI due to the sheer amount of explanations needing to be created for each dataset. With these caveats, the results are shown in Table 3.8 and Figure 3.15

As with local accuracy we see that the baselines perform poorly compared to the other region-based techniques tested. However, it is interesting to see that the difference between our baselines and the non-baseline methods is not as great as when measuring the local accuracy. Indeed, the centre baseline performs better than the guided-backpropagation for the Stanford Dogs dataset. This suggests that a certain amount of coarseness is desirable in order to achieve a good score with this metric. This is shown by the Grad-CAM results which both perform almost identically to one another and give good global accuracy scores across both datasets. LIME is interesting as it performs well with the CUB200 dataset, but poorly with Stanford Dogs. Looking at the plots in Figure 3.15 suggests that initially LIME has a sharper drop than the CAM methods but then performs poorly during the later stages of feature removal. The good initial performance is likely due to the use of superpixels which closely allign to the object regions ion the image. This is useful for obtaining a good global accuracy as ideally, assuming all the class features are in the object, we would be able to remove the entire object without removing the background. The use of superpixels goes some way to allowing this. It is likely that the superpixels used by LIME are too coarse for this.

The use of superpixels is the reason we believe that all the SWAG variants perform so well in the global accuracy experiment. Unlike LIME, SWAG uses a much larger number of superpixels allowing for better alignment with the boundaries of the object. Within the object, the increased number of superpixels are able to more accurately identify the important regions, rather than having them combined into a single large superpixel. Unlike in the local accuracy experiment, SWAG_G performs worst of the SWAG variants, although still better than the other methods tested. This is likely due to the expansion of the superpixels away from the object boundaries. As has been noted previously by Kapishnikov *et al.* [65], the areas surrounding the boundaries of the object are important to the model. This is the reason XRAI artificially expands their superpixels regions. However, with the global accuracy, this is not desirable as it means we are more likely to remove background pixels rather than object pixels. This is likely what leads to the inferior score for SWAG_G. SWAG_I performs well as the superpixels will align closely to the object boundaries, however $\text{SWAG}_{\text{I+G}}$ performs the best of all methods tested. $\text{SWAG}_{\text{I+G}}$ gives the best global accuracy results because it finds a happy medium between SWAG_I and SWAG_G. By incorporating enough of the image, the superpixels stay better aligned to the object boundaries, but by incorporating a certain amount of gradient when making the superpixels, features within the object are more accurately scored. It is possible that the global accuracy score achieved with $\text{SWAG}_{\text{I+G}}$ could be improved as we chose the w_c and w_g to work best for the local accuracy metric. However, due to the computational requirements of the ROAR technique, it is impractical to perform hyper parameter optimisation.

3.7.3 Superpixel Replacement for LIME

Superpixels form the underlying foundation for techniques such as LIME and XRAI. Here the superpixels are generated, then either perturbed or weighted in some way. However, both of these examples use "off-the-shelf" superpixels, Quickshift for LIME and Felzenszwalb for XRAI. In this experiment we demonstrate the usefulness of using our proposed SLIC variants as a drop in replacement for those used in LIME. We use the local accuracy experiment and generate superpixels using off the shelf SLIC as well as our combined image and gradient, and gradient only variants. These are referred to as LIME_I,

TABLE 3.8: ROAR AUC results. Lower is better.

Method	CUB200	Dogs
Centre	0.284	0.393
Grad-CAM	0.219	0.344
G-CAM++	0.218	0.342
LIME	0.210	0.364
$SWAG_I$	0.173	0.320
$SWAG_{I+G}$	**0.172**	**0.319**
$SWAG_G$	0.179	0.324
Guided Backprop	0.231	0.435

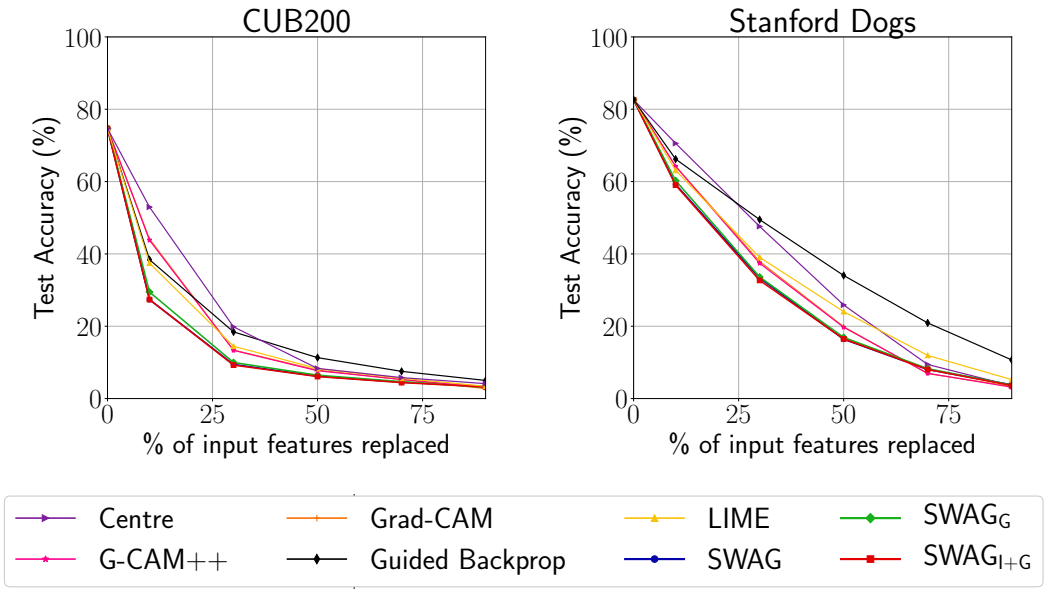

FIGURE 3.15: ROAR results using the ResNet50 architecture. Faster accuracy drop indicates a better global explanation.

LIME $_{I+G}$, and $LIME_G$ respectively. In the initial LIME implementation using Quickshift, approximately 37 superpixels are created. To mimic this we use SLIC with the approximate number of superpixels to generate set to 50. The results for this experiment using ImageNet are shown in Table 3.9. From these results we see that simply using our SLIC variants as a drop in replacement

TABLE 3.9: Local accuracy results for LIME. Changing Quick Shift to SLIC and our variants (I+G and G). Lower is better.

Method	VGG16	ResNet50
LIME_{I}	0.107	0.126
$\text{LIME}_{\text{I+G}}$	0.090	0.108
LIME_{G}	0.082	0.099

allows for local accuracy to be dramatically improved for both VGG16 and ResNet50. Indeed such is the improvement that with only approximately 50 superpixels used for the explanation with LIME, the results are close to SWAG using 300 superpixels. This is potentially very useful to users of LIME as it would allow only a modest increase in the number of superpixels using our technique to achieve superior results. Without our superpixel technique many more superpixels may be required, which would in turn requires a large number of additional perturbations.

3.7.4 Weak Localisation

A common experiment undertaken to assess an explanation method is to investigate its ability to locate a salient object within an image. We used the well-established method [11,64,133] outlined in Section 3.5.3. Achieving a good score across the weak localisation experiment indicates that an explanation is able to both identify where in the image the object is, and also be cohesive enough to score a large number of pixels within the bounding box. While a useful proxy to get insight into the cohesiveness of an explanation, weak localisation experiments do not directly measure the accuracy or quality of an explanation, Petsiuk *et al.* [9].

The results for the weak-localisation experiment using ImageNet are shown in Table 3.10. For the VGG16 network we are able to obtain a better localisation score for two of the three metrics than Grad-CAM. Interestingly for VGG16, all of our SWAG methods perform better than the others when thresholding using the mean or the energy. However, it seems that our method performs poorly when thresholding by pixel value, most likely due to the uneven distribution of

Table 3.10: Weak-localisation results as % of localisation error. Lower is better.

	VGG16			ResNet50		
	Val	Mea	Eng	Val	Mea	Eng
Random	57.43	58.96	57.39	57.43	58.96	57.39
Centre	47.57	48.18	47.68	47.57	48.18	47.68
Grad-CAM	52.06	49.76	51.80	45.94	45.89	44.35
Grad-CAM ++	**47.32**	47.25	46.08	**45.76**	**45.83**	**43.85**
LIME	54.82	52.40	52.82	53.08	50.77	51.19
RISE	55.01	57.94	49.68	52.73	53.82	50.53
$SWAG_I$	55.06	46.10	45.01	56.73	52.50	50.65
$SWAG_{I+G}$	54.57	46.44	44.95	56.33	52.10	50.70
$SWAG_G$	54.16	**45.95**	**44.86**	56.69	52.07	52.02
Guided Backprop	55.28	46.32	49.63	56.44	51.53	52.35

values between superpixels. For ResNet50, we note that our method does not perform well when compared to Grad-CAM and Grad-CAM++. Of particular interest in these results, is how well simply pointing to the centre of the image performs. For VGG16 this baseline, performs better than the majority of methods apart from Grad-CAM++, and in ResNet50 is only approximately +2.5% from the best performing methods. This, again, highlights that weak-localisation is perhaps not a good measure of accuracy or quality.

Our method mirrors and in some cases outperforms guided backpropagation. As guided backpropagation is used to weight, and in some cases define our superpixels, it is likely that using an alternative method could yield better results using our method.

3.7.5 Efficiency

To measure the efficiency of an explanation technique we measure the average time to compute a single explanation for each of first 5,000 images of the ImageNet validation set, each cropped to 224×224. These results were computed using a single NVidia Titan X GPU. We show the average time taken to compute each explanation in Table 3.11 and Figure 3.16.

Table 3.11: Mean computation time in seconds

Method	VGG16	ResNet50
Grad-CAM	0.03	0.03
G-CAM++	0.03	0.03
LIME	5.80	4.76
RISE	13.19	17.48
XRAI	31.10	30.57
$SWAG_I$	0.12	0.18
$SWAG_{I+G}$	0.12	0.18
$SWAG_G$	0.07	0.10

From these results, we see that our method produces explanations that are efficient to produce. When compared with CAM based methods, we see that SWAG based methods are marginally slower, however we are able to obtain a much better local and global accuracy for only a relatively small increase in computational cost. The additional overhead incurred by the SWAG methods is due to the creation and weighting of the superpixels. Notably $SWAG_G$ is quicker to compute that the other SWAG methods as it does not have to deal with the colour channels and so only creates superpixels using a single grey scale channel. Compared to LIME (the other technique using superpixels in a similar way to ours) we see that our technique is much faster to compute due to only requiring one pass through the network. RISE is also an inefficient method to generate explanations due to the number of images required to be passed through the network. However, XRAI is the most inefficient method by far for two reason. The first is the use of integrated gradients as the gradient method underlying their explanations. In the code the authors released, they use 2 baselines, each with 100 passes through the network. The second is the use of 6 sets of superpixels that have to be iterated through and weighted. Overall, this leads to a very inefficient explanation method.

3.7.6 Attribution Accuracy

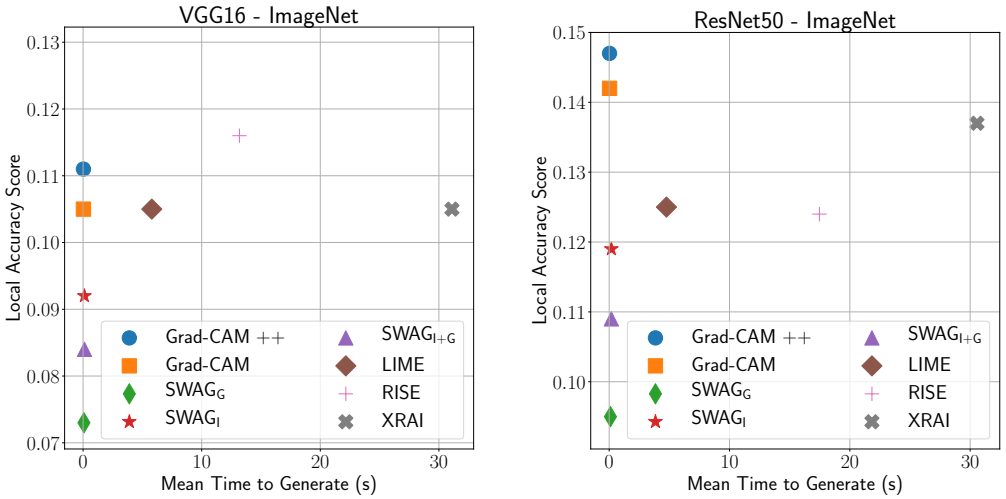

FIGURE 3.16: Computation efficiency Vs local accuracy AUC score. A lower AUC is better.

The final results that we present are for the attribution accuracy metric discussed earlier in this chapter. This is a measure of much an object with known boundaries is highlighted by the explanation. If an explanation covers an object without highlighting the background it will score well.The results for this experiment using ResNet50 are shown in Table 3.12. Here, we see that the coarser explanations achieve higher scores using this metric. For example, Grad-CAM++ which achieved the worst local accuracy score (excepting baselines) using ResNet50, here achieves the highest score for this metric. Indeed it seems that the better a method is at the local accuracy metric, the worse it scores with this metric. This seems to happen because the model contrast score (MCS) gives higher rewards for an explanation completely covering the desired object. However, we have seen in the qualitative examples (Figures 3.11 and 3.12) that the explanation methods that score well with the local accuracy metric are typically more focused and precise than those which do not. This would then result in less of the object being covered by the explanation as an it is able to better differentiate between regions of the object that are important or not.

Table 3.12: MCS scores using the BAM dataset. Higher is better.

Method	ResNet50
Grad-CAM	0.312
G-CAM++	0.326
XGrad-CAM	0.312
LIME	0.210
RISE	0.202
XRAI	0.243
$SWAG_I$	0.194
$SWAG_{I+G}$	0.200
$SWAG_G$	0.194
Guided Backprop	0.044

3.8 Chapter Summary

In this chapter, we have introduced two complementary techniques. The first technique introduced is SWAG, a method for efficiently creating accurate explanations through the use of backpropagated gradients and superpixels. The second is a technique for incorporating gradients into the superpixel formation process using either the gradients alone, or the combined image and gradient.

We have shown through experimentation that our SWAG methods offer improved local and global accuracy compared to a range of comparable techniques. This is achieved in a computationally efficient manner. The trade-off is a drop in interpretability which favours coarser, less accurate techniques. Of the techniques proposed ($SWAG_I$, $SWAG_{I+G}$, and SWAG $_G$) $SWAG_G$ performs the best in local accuracy and weak-localisation, while $SWAG_{I+G}$ performs the best in global accuracy. Both $SWAG_{I+G}$ and $SWAG_G$ perform better than $SWAG_I$. However, interestingly $SWAG_G$ performs poorly in global accuracy suggesting, that $SWAG_{I+G}$ is the best all-round technique of the three.

In this chapter we also experienced limitations in the range of metrics we were able to fully test. This was primarily due to the computationally inefficient manner of these metrics. In particular, we found that the global

accuracy metric ROAR was particularly inefficient. A decision was made to run the global accuracy metric instead of the insertion portion of the local accuracy metric. Going forward, due to the inability to run ROAR in an efficient manner, we will drop the global accuracy, and instead focus on running both insertion and deletion local accuracy metrics.

CHAPTER

4 SWAG-V: Explanations for Action Recognition

4.1 Introduction

In this chapter, we extend the SWAG technique from the previous chapter to work with networks that use video as an input, specifically action recognition networks. These have a number of different requirements compared to image networks. We begin by presenting a brief literature review of the architectures used for action recognition, before moving on to discuss how current techniques are used to create explanations. We then show the motivation behind introducing SWAG-V (SWAG for Video) and discuss how it is implemented. A number of experiments are then conducted comparing our proposed technique to alternative methods and baselines. The key contributions of this chapter are two-fold. The first is that we introduce a robust video explanation technique to the video explainability field, an area of research that is often lacking. The second is that we introduce a method for finding the optimal trade-off point between insertion and deletion metrics. This allows explanations to be created in such a way that we find a middle ground between being too coarse and too fine.

4.2 Action Recognition Review

Action recognition is the task of being able to detect or classify a person's actions. We will first discuss deep learning approaches to action recognition,

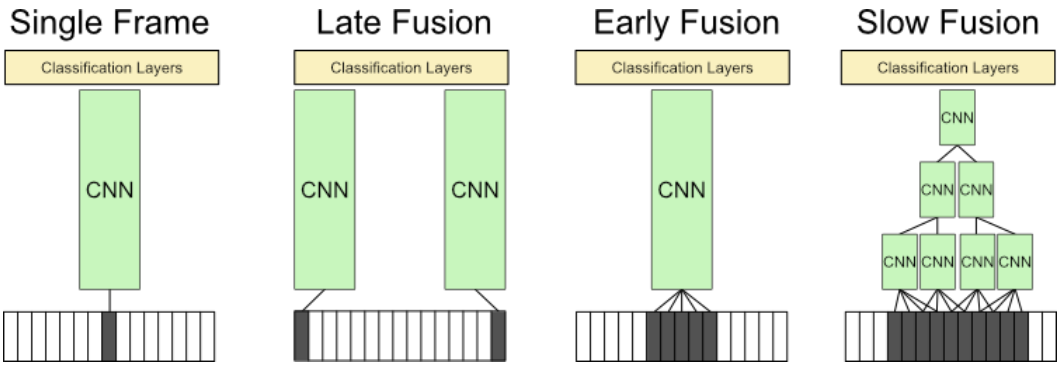

FIGURE 4.1: An overview of the different fusion methods. The 16 frame clip is shown at the bottom, with the dark gray frames being those used as an input.

highlighting the differences from image networks that create problems when generating explanations. Then, we briefly discuss the datasets used in this chapter.

4.2.1 Deep Learning Approaches For Action Recognition

Deep learning was invigorated by the ImageNet challenge [128]. This was a challenge to correctly classify and detect objects in 2D images. While successes in 2D images [134] were the catalyst for the shift towards the use of deep learning methods, the use of deep learning with videos has proven more difficult due to the videos combined temporal-spatial nature. As such, a number of differing techniques for handling video data in Deep Learning has emerged. The majority of techniques initially followed two differing architectures:

1. Two stream approaches that separate spatial and temporal components.
2. 3D spatio-temporal convolutions that combine spatial and temporal components.

However, more recent approaches have combined these two styles of architecture to give improved results.

4.2.1.1 Two Stream Approaches

One of the first attempts at using Deep Learning for action recognition in videos was attempted by Karpathy *et al.* [135]. This work was important in two ways. The first was that it introduced Sports-1M, the first large-scale video dataset

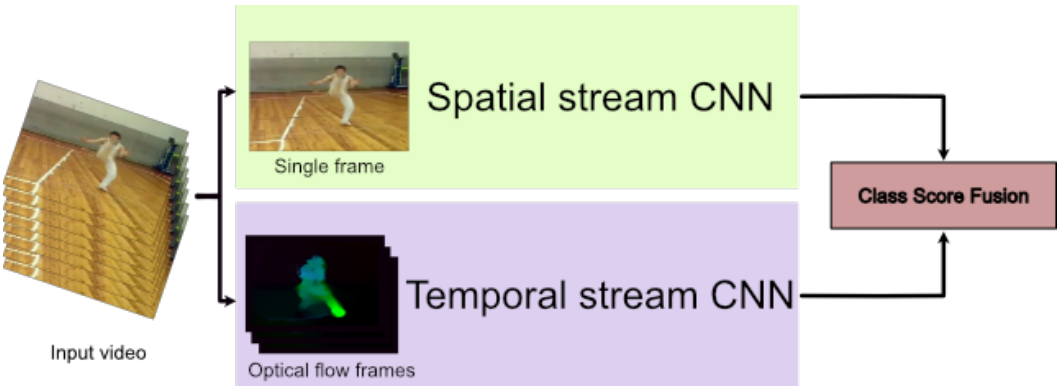

FIGURE 4.2: An overview of the two stream approach showing one stream taking the RGB frames, and a second taking the optical flow representation.

to the community. Until the introduction of later datasets this was commonly used to pre-train action recognition networks in the same way ImageNet is for image networks. The second contribution was the exploration of how to use the frames of a video as an input. At this point Karpathy was still using 2D CNNs so relied on the concept of giving each CNN a frame, and then fusing the multiple 2D CNNs together. Aside from using a single frame as an input to a 2D CNN as one might with image classification, three fusion methods were proposed: late fusion, early fusion, and slow fusion (see Figure 4.1). Slow fusion, which was found to work best, uses blocks of four frames (of a 10 frame clip input) with an overlap of two have their own one-layer convolution network, these are then fused into two separate networks for a layer before being fused into a single network. Using the slow fusion technique preserves the temporal information until the third convolution layer. An interesting aspect to note though was that these fusion methods only offered small increases over using a single frame as input to the CNN.

The two concepts of preserving a combined spatio-temporal input for as long as possible in the network, and having two separate networks that merge late on are important for the architectures that followed.

4.2.1.2 Two Stream Approaches

The first of these newer architectures, a two stream approach, was proposed by Simonyan and Zisserman [136]. This used two separate networks as per

the late fusion method seen earlier, but had one network accepting a single frame, and the second network accepting a stack of optical flow frames (see Figure 4.2). Both networks use 2D convolutions, this means that the stack of optical flow frames containing the temporal information is convolved over in its entirety at the first convolution layer. This results in the network being unable to learn from the changes in the temporal information at lower layers. The two networks are not joined until after their respective softmax layers. The authors found that using a Support Vector Machine (SVM) [137] proved to be the best method of joining the softmaxes rather than averaging them.

The two stream approach decouples the spatio-temporal information which allows the spatial network to be pre-trained on a large body of data, in this case ImageNet [128].

This architecture has been modified in various ways since its inception, with more complex fusion techniques [138], improved appearance and motion representation [139], and action localization [140]. Feichtenhofer *et al.* [40], introduced a two stream architecture but with ResNets with temporal to spatial links in place of the original ones. ResNets [68] allow networks to be trained for more layers with a reduced risk of overfitting, this allows more complex representations to be formed.

4.2.1.3 Combined Spatio-Temporal Approaches

The second main architecture used for action recognition in videos is a combined spatio-temporal approach. Although the first example of 3D Convolution Networks were introduced by Ji *et al.* [141] and Baccouche *et al.* [142], they were not used for large-scale supervised training until the C3D network introduced by Tran *et al.* [37]. C3D networks allow the temporal element of the video to be propagated through the network for longer, as opposed to losing the information at the first layer as was the case with the two stream network. The key difference is that rather than using 2D convolution filters, that work over the entire temporal volume, 3D convolution filters are used instead (see Figure 4.3). This allows blocks of temporal information to be processed independently, allowing the temporal information to persist further into the network. The C3D network has eight 3D convolution layers, with the temporal information lasting until the final layer. The authors found that a 3D convolution filter of

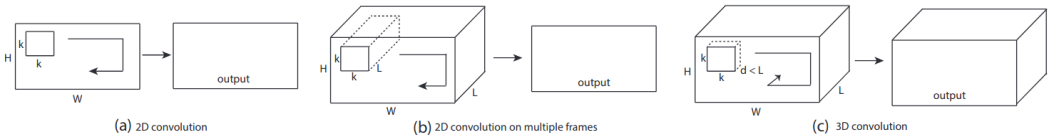

FIGURE 4.3: The difference between a 2D convolution (a), a 2D convolution over multiple frames as used in the two-stream network (b), and the 3D convolution used in the C3D network (c). Note the output of the 3D convolution contains an additional dimension. Taken from Tran *et al.* [37].

size 3×3×3 gave the best performance when working with combined spatio-temporal features. Their experiments showed that using a 3D CNN gives more suitable spatio-temporal features compared to using a 2D CNN.

4.2.1.4 Joining the Two Techniques

As Girdhar *et al.* [143] noted, two stream networks often outperform other architectures as they are flexible enough to allow for the latest architectures to be swapped in for the existing networks. However, the combined spatio-temporal nature of C3D showed that these architectures were able to replace 2D networks.

An architecture proposed by Varol *et al.* [139] combined aspects of both the two-stream and spatio-temporal approaches. This technique was called Long-term Temporal Convolutions (LTC). From the two-stream approach it takes the two-stream architecture with use of separate optical flow and appearance data. From the spatio-temporal approach it takes the 3D convolution network and inserts it into the two networks. The authors also explore the expansion of the networks temporal depth input, from C3D's original 16 frames, up to 100 frames, and show that the increased depth results in an improved performance. This result is corroborated by Ng *et al.* [144] who show that optimal classification performance is given by 120 frame inputs for their network. While the LTC work gave better results, a limitation of this approach became the performance of C3D compared to more modern 2D architectures such as Inception [145] or ResNet [68]. To combat this problem, Carreira and Zisserman [96] proposed the idea of taking more current 2D architectures, and 'inflating' them to produce 3D networks. This technique is called Two-Stream Inflated 3D ConvNets (I3D).

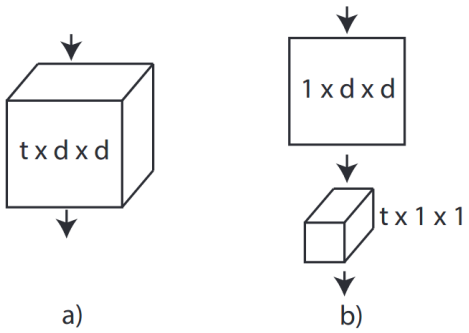

FIGURE 4.4: An overview of the difference between 3D convolutions (a) and the separated variation used for R(2+1)D (b). The separated variation shows how the spatial and temporal elements are decomposed into two separate convolutions. Taken from Tran *et al.* [38].

I3D was based around taking the inception architecture and inflating the 2D convolutions to become 3D convolutions, by adding a temporal element. These inflated networks were then used in a two stream approach. This gave state of the art performance for a number of action recognition datasets. Following this work, Tran *et al.* [38] introduced ResNet3D and R(2+1)D. ResNet3D was created in a similar way to I3D, a ResNet model was inflated to use 3D Convolutions. R(2+1)D takes a different approach, and is based on work done by Qiu *et al.* [146]. Here, the 3D convolutions are split into a combination 2D and 1D convolutions. This is shown in Figure 4.4. The 2D component performs a spatial convolution followed by a 1D convolution over the temporal extent. The benefit of this is to double the number of non-linearities, allowing more complex functions to be represented. In the paper, the authors show that R(2+1)D performs better than ResNet3D and only marginally worse than I3D.

4.2.2 Datasets

A number of datasets exist for action recognition applications. However, not all are suitable for deep learning tasks due to their limitations in both quantity of samples and action classes. However, a small subset are repeatedly experimented upon within the deep learning literature, notably UCF101 [147], Sports1M [135], ActivityNet [148], THUMOS [149], and Kinetics [150]. In particular, Kinetics is now the standard dataset to pre-train and test networks on. We will give a brief description of it here as we use it throughout the

chapter. In addition we also discuss UCF101 as this has a subset of the dataset annotated with ground truth bounding boxes that localise the action. A more thorough review of other action recognition datasets can be found in Chaquet *et al.* [151].

4.2.2.1 Kinetics

The Kinetics 400 dataset was initially introduced in 2017 by Kay *et al.* [150] and developed in parallel with the I3D network [96]. Kinetics 400 is a dataset consisting of 400 classes of human actions, each with over 400 clips per class. The motivation for introducing Kinetics 400 was two fold. The first was to provide more classes and data with which to train on, the second was to increase the variation within the classes. This has been shown to be a problem with datasets such as UCF101 where a lack of variation allows a network to learn the background rather than the action [152]. For example, in UCF101 all tennis playing takes place on a tennis court, simply seeing a tennis court indicates the action taking place. Kinetics 400 attempts to diversify this by having actions take place in a variety of environments. Kinetics 600 [153] was introduced as an extension to Kinetics 400 with Kinetics 700 [154] following as a subsequent extension.

4.2.2.2 UCF101

UCF101 was introduced in 2012 by Soomro *et al.* [147] and incorporates and builds upon previous implementations of UCF datasets, UCF11 [155] and UCF50 [156].

The dataset was introduced as a way to remedy the perceived problems in action recognition datasets of the time, notably, small numbers of classes and videos that were not real (i.e. they were staged or acted). UCF101 is made of 13,320 videos split into 101 classes. Each of the videos is obtained from YouTube and irrelevant clips were removed at the point of creation, meaning the dataset consists of videos with a single action in and accurate labels. Of potential interest, when further analysing the datasets inclusion in the project, is that the 101 classes are also grouped into five high-level groups: Human-Object Interaction, Body-Motion Only, Human-Human Interaction, Playing Musical Instruments, and Sports. For deep learning purposes the

dataset is often seen as being small so is not often used to train a network from scratch due to over fitting [37]. Of importance to this chapter is the subset of UCF101 that was introduced as part of the THUMOS challenge [149]. It contains bounding boxes for 24 of the 101 classes, giving the location of the action in each frame of the video.

4.3 SWAG for Video: SWAG-V

We propose that the technique discussed in the previous chapter, SWAG, is extremely well suited for use in action recognition networks where the input is a spatio-temporal volume. Action recognition networks typically take an input that consists of multiple 2D images (video frames) stacked into a single volume. This stacked frames are temporally cohesive. To represent this in the form of an explanation we again use SLIC, but instead create 3D superpixels that posses both a spatial and temporal element. As with SWAG, the gradients are backpropagated to the input and used to weight the 3D superpixels. As every individual frame has an associated set of gradients and superpixels, the technique for weighting each frames' superpixels is identical to the 2D version of SWAG.

In this chapter, we will again show how superpixels can be generated using both the image and the gradients individually as well as combining them. We will refer to these as SWAG-V_{I}, $_{G}$, and $_{I+G}$ respectively.

Another key difference between SWAG and SWAG-V will be how we optimise. As SWAG was image based, it is considerably more efficient to compute results for. To this end we were able to generate results for both local and global accuracy metrics. However, using video data makes the generation of global accuracy results particularly inefficient and therefore prohibitively lengthy. Instead of showing results for local and global accuracy, we instead double down on local accuracy as the means of showing explanation accuracy. What this means in practice is that we will generate results for both the insertion and deletion metrics for the RISE deletion metric. Previously, for SWAG, we only dealt with the deletion metric as we felt it to be more important. It is important to note that the metrics available for evaluating explanations for action recognition networks are much more limited than those proposed for image based networks. The majority of these metrics are therefore applied in

the same manner as they would be for an image based-network. Where metrics that incorporate the temporal element do appear, they are typically part of localisation metrics [41, 98].

4.4 Optimisation

While the technique is only altered from SWAG through the use of 3D superpixels rather than 2D ones, a number of changes must be considered to adapt it to use for action recognition networks. The primary considerations are the attribution method (i.e. which gradient method to use), the number of superpixels to create, and how best to combine superpixels and gradients to create SWAG-V $_{\text{I+G}}$. We explore the attribution method first, as while we found that guided backpropagation worked well for 2D images, there is no certainty this performance will transfer to video. From SWAG we keep the use of SLIC as the method of generating superpixels and mean pooling as the method for weighting each superpixel with the gradients. In this section we therefore present the rationale behind the choices for the primary considerations we have just discussed.

Throughout this section we show results using Kinetics 400 with the R(2+1)D architecture. We start with an initial value of 100 superpixels, and then update this when we determine the optimal value.

4.4.1 Attribution Method

The attribution method is key to the success of SWAG-V. Determining which gradients produce the best initial results is important to discover early on before further decision are made. In this section we show results for two baselines (random noise and Sobel edges), and three methods for generating gradients (vanilla backpropagation, guided backpropagation, and Input$\odot$Gradients). We again chose these baselines as, although we are now using video as an input, the baselines can be generated for each frame in an input clip. This results in a similar appearance to gradient based methods. These results are shown in Table 4.1. From these results we observe that the baselines perform poorly for both deletion and insertion metrics. Vanilla backpropagation performs almost as poorly as random noise for the deletion metric, but performs much better for the insertion metric. With Input$\odot$Gradients this is reversed with

TABLE 4.1: Determining the optimal method of attribution. Results are shown for both insertion and deletion local accuracy metrics. For deletion, lower is better. For insertion, higher is better.

Method	Deletion R(2+1)D	Insertion R(2+1)D
Random Noise	0.194	0.192
Sobel	0.162	0.298
Vanilla Backprop	0.188	0.302
Guided Backprop	**0.113**	**0.371**
Input⊙Gradients	0.156	0.185

the deletion metric performing much better than the insertion metric, which performs worse than random noise. Of all the techniques tested though, it is clear to see that, as with SWAG for 2D images, guided backpropagation performs the best. Easily beating all the other techniques in both deletion and insertion results. We will again use guided backpropagation going forward.

4.4.2 Choice of Weights for SWAG-V_{I+G}

As with SWAG, it is not initially obvious that when creating a superpixels using a combination of the image and gradients, what is the best way to combine them. In this section we determine the optimal way of combining the images and gradients in the superpixel creation process. To do this we modify the weights for each input as we did in the previous chapter. The weights are labelled w_c and w_g for the image and gradients respectively. Each weight modifies how important the input is in the generation of the superpixel. A low weight score means a high importance, while a high weight score gives lower importance.

For both deletion and insertion metrics, we begin by performing a coarse grid search using w_c and w_g values of $[5, 10, 20]$. Here, 10 is the default value used by SLIC, while 5 and 20 represent a doubling and halving of an inputs influence respectively. As with SWAG, to ensure that some aspect of the image is always taken into account, we only increase the image weight to a maximum of 20. Based on these coarse results we then performed a finer grid search in

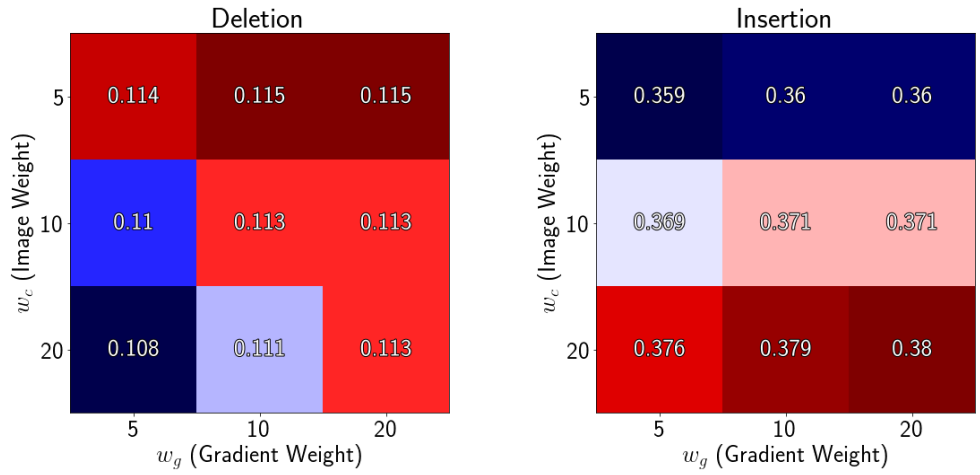

FIGURE 4.5: Coarse search for optimal values for w_c and w_g. Left is deletion scores. Lower (dark blue) is better. Right is insertion scores. Higher (dark red) is better.

the best performing region for each metric. The initial coarse results can be found in Figure 4.5.

From these initial coarse results, we can see that both deletion and insertion results tend to favour a high w_c value and a low w_g value. This suggests that both metrics find the addition of the gradients to be beneficial to creating superpixels that better align with the explainable regions.

Based on this, we narrow on in the best performing values and subsequently produce a finer grid search using the combined scores of insertion and deletion to find the optimal value. We combined the scores as so: (deletion + (1−insertion)). Subtracting the insertion score from 1 means that as both scores approach 0, the better they are. We compute the combined scores for $w_c = [18, 20]$ and $w_g = [8, 9, 10, 11, 12]$. These results are shown in Figure 4.6. From these results we see that the lowest score occurs at $w_g = 9$ and $w_c = 20$. We will use these values for SWAG-V_{I+G} going forward.

4.4.3 Initial Superpixel Count

With SWAG we determined that an initial count of 300 was appropriate for creating explanations for 2D networks which typically take a 224×224 image as an input. However, action recognition networks typically take an input with

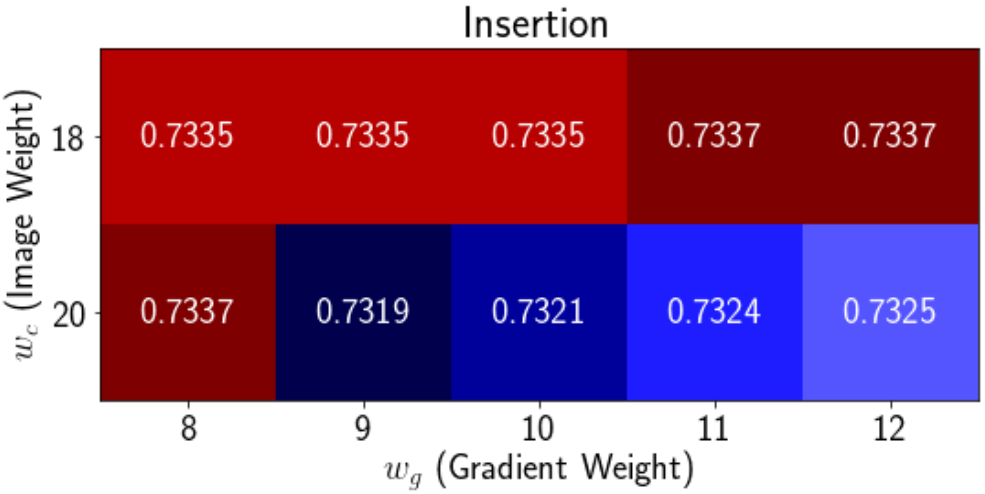

FIGURE 4.6: Fine search for optimal values for w_c and w_g. Here we show the combined insertion and deletion score. Lower (dark blue) is better.

a spatial dimension of 112×112. Simply using the same number of superpixels as SWAG may result in explanations that are therefore too fine grained. In this section we explore the optimal number of superpixels to generate. We again use both the insertion and deletion metrics. We sweep through a range of superpixel counts and store the results. To find the optimal value, we add the deletion scores to 1−insertion scores. In this way both sets of scores are better as they approach 0. The results from the sweep of superpixels counts and this combined result is shown in Figure 4.7. From the deletion and insertion results we see that they are the opposite of each other. Deletion (where a low score is better) improves the more superpixels there are in the image. The insertion metric (where a high score is better) performs much better with a very low number of superpixels and degrades as more are added. Again, looking at the combined values (deletion + (1−insertion)), where a low value is better, we see that after an initial rapid improvement in performance, the scores plateau. When we analysis this we find that the lowest score occurs at a superpixel count of 120. This is marked with the red cross in the figure. Going forward we will use a superpixel count of 120.

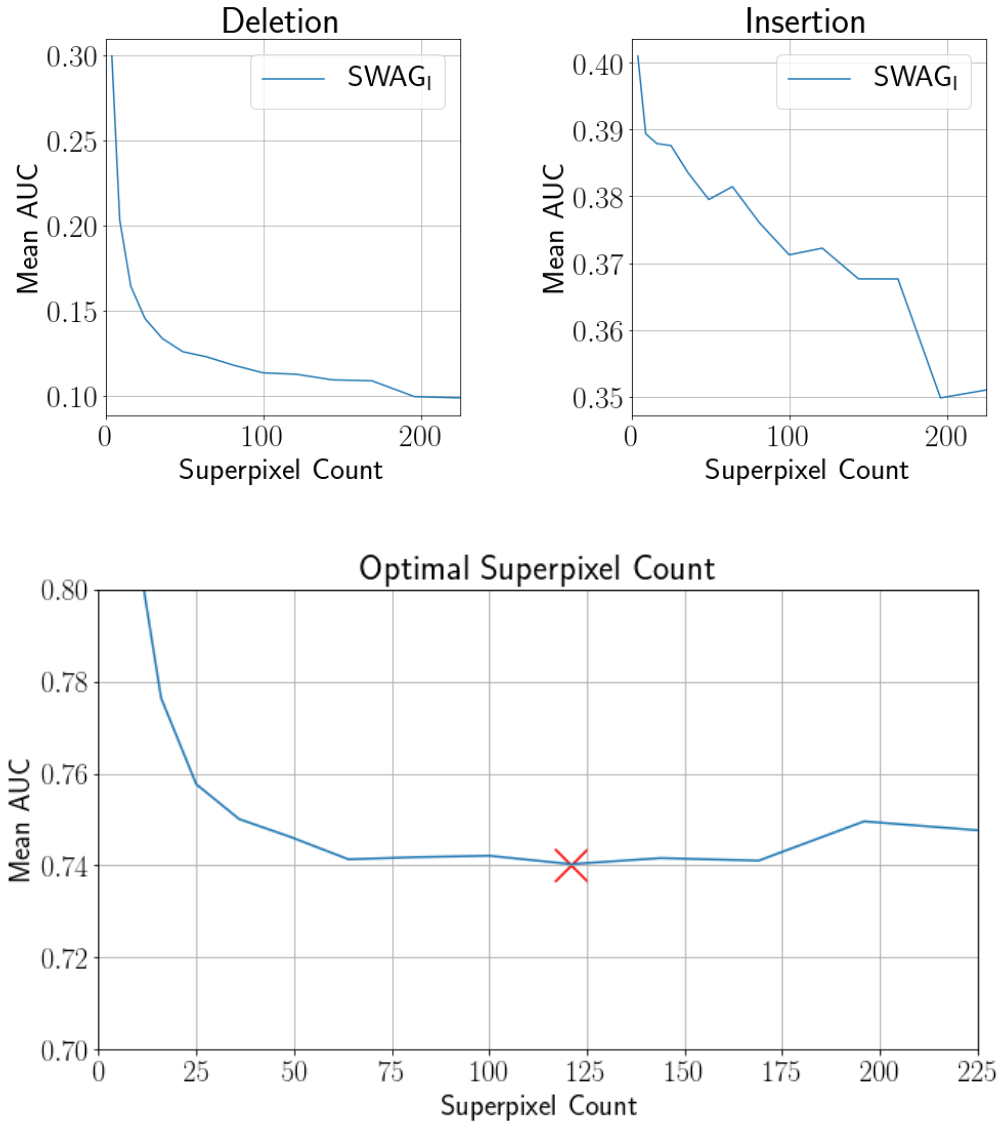

FIGURE 4.7: Top: Showing how the superpixel count affects the insertion and deletion scores. Bottom: Showing the optimal superpixel size.

4.4.4 Final Parameter Choices

For clarity we present the final parameter choices for SWAG-V determined in this section.

- **Attribution Method**: We chose *guided backpropagation* as our attribution method. As with SWAG, we found it to outperform the other methods tests using the local accuracy metric.

- **Initial Superpixel Count**: Using the combined deletion and insertion results, we found that *120* was the optimal superpixel count to start the explanation with.
- **SWAG-V$_{\mathbf{I+G}}$ weights**: We showed, using the combined deletions and insertion results, that $w_g = 9$ and $w_c = 20$ were the optimal values for use with SWAG-V$_{\mathrm{I+G}}$.

Using these values, we now begin to conduct our experiments.

4.5 Experiments

In this section, we present our results based on the application of Gradient Weighted Superpixels to action recognition networks. As discussed previously, we find our technique well suited to explaining networks that use spatio-temporal volumes. In this section we explore this both qualitatively, through the use of example images, and quantitatively. For the quantitative metrics we use local accuracy, but in this chapter we use both the insertion and deletion metrics. Due to the computational requirements, we do not run any experiments for global accuracy. For weak-localisation, we use a variation of the metric used for SWAG, although tailored to work with the video data available to us.

We hypothesise that we will achieve similar results to those of SWAG. That is, we expect SWAG-V to perform well in the local accuracy deletion metric. With the addition of the insertion metric, we expect this not to perform as well as methods such as Grad-CAM, but rather that SWAG-V will perform consistently well across both insertion and deletion. Unlike SWAG, we expect SWAG-V to perform better than existing techniques for weak-localisation tasks as it should be able to better localise action both spatially and temporally.

4.5.1 Qualitative Inspection of Results

We begin by presenting an example explanations created using Kinetics 400 and R(2+1)D. Examples for the classes capoeira (a Brazilian martial art) and shearing sheep are shown in Figures 4.8 and Figures 4.9 respectivly. In the figures we show individual frames from a 16-frame clip. We show our results alongside the initial input frames as well as the activation map based methods. From this example we can see how much more precise all variants of SWAG-V

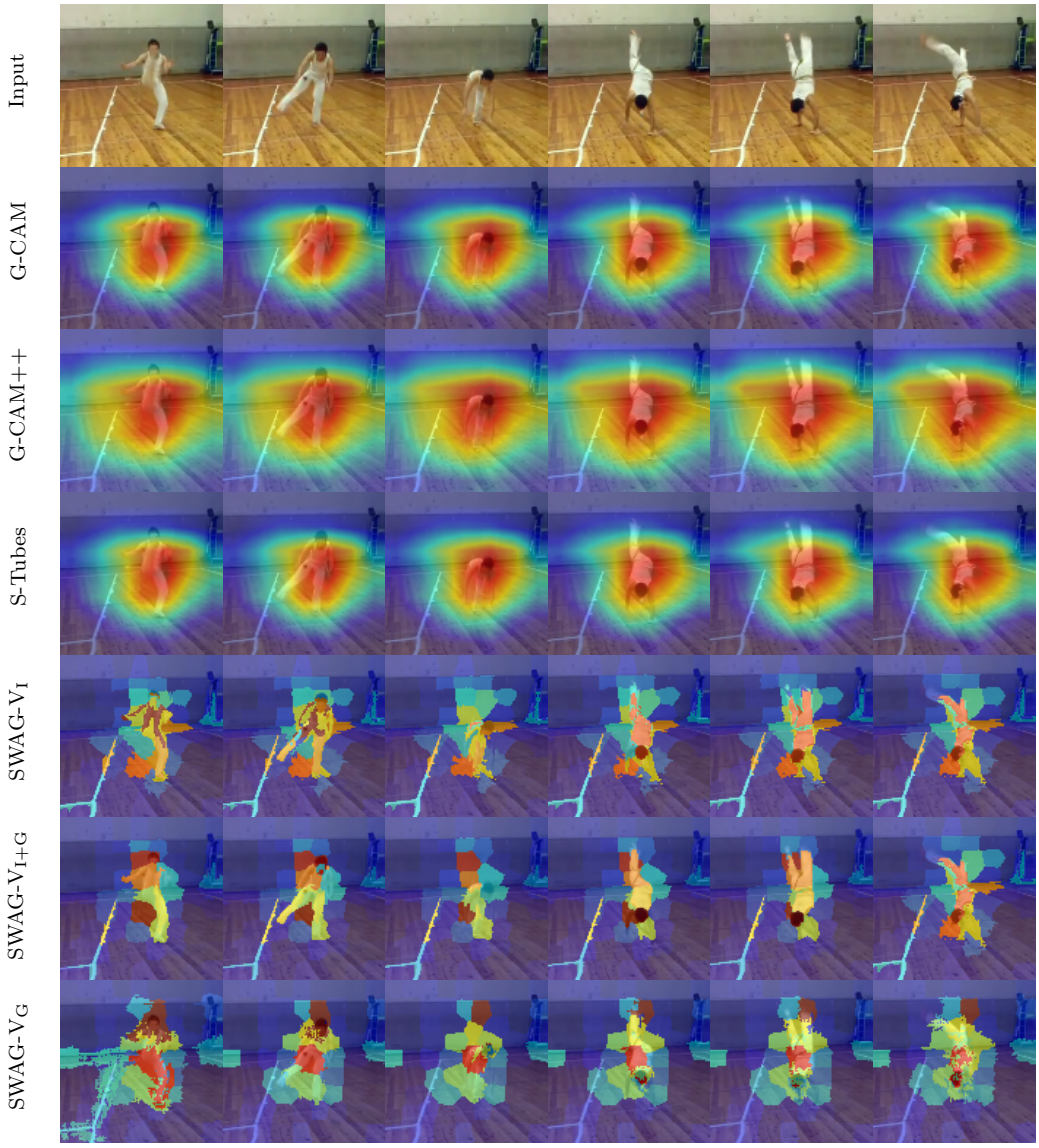

FIGURE 4.8: An explanation for the Kinetics 400 class 'capoeira' using R(2+1)D. Note the more precise explanations generated using SWAG-V compared to previous methods.

are when compared to the activation map based methods. These methods are shown to be coarse in both the spatial and temporal dimensions. For example, note that the activation map based methods are unable to precisely follow the motion of the person in the capoeira action class. Instead the explanations produced by the activation based methods seem to highlight the centre of the frame and highlight all of the action in that region. In comparison, we see that our proposed methods are much able to better track the location of limbs of the person as they move. Interestingly, SWAG-V$_G$ appears to be much noisier, with

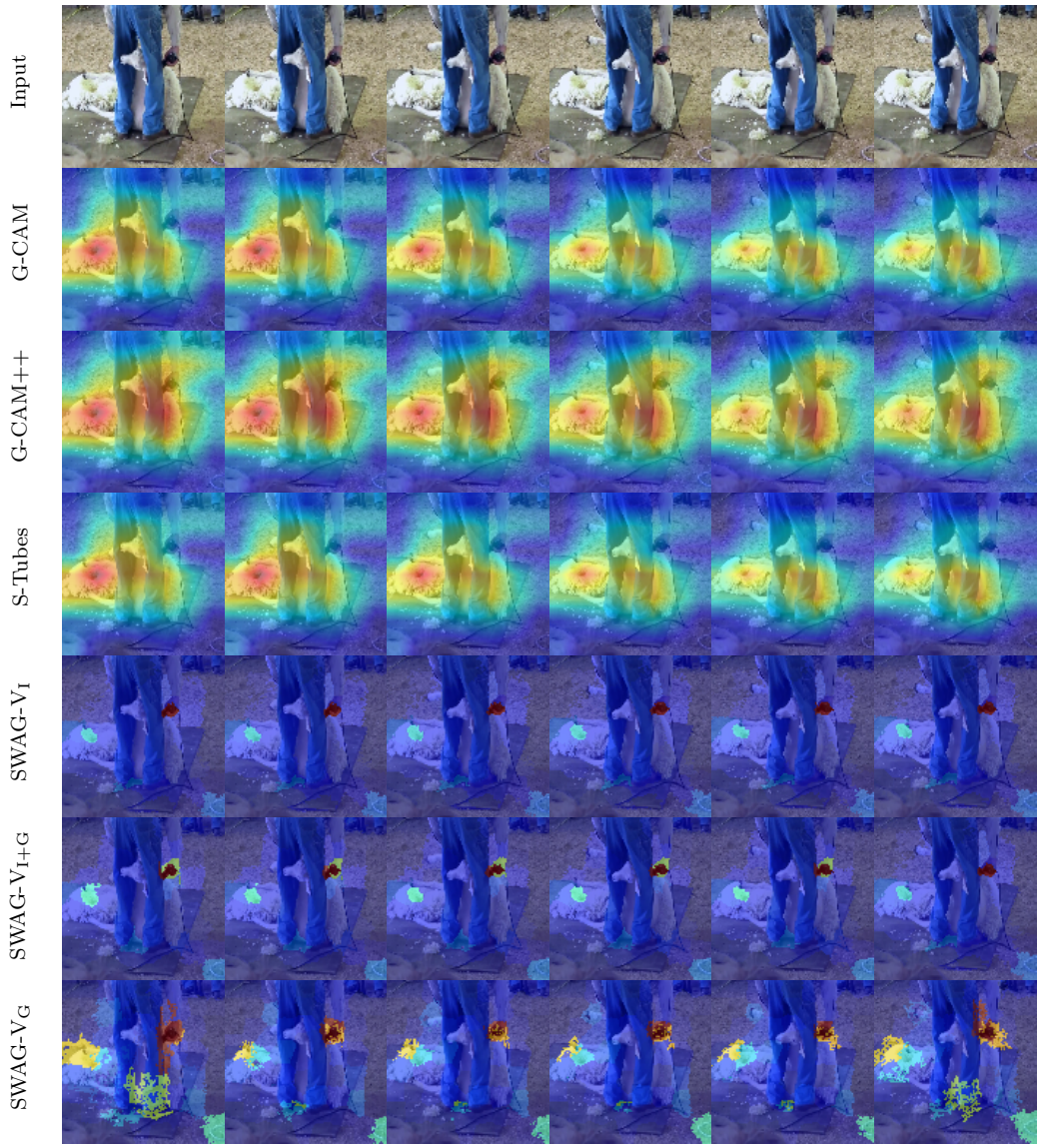

FIGURE 4.9: An explanation for the Kinetics 400 class shearing sheep using R(2+1)D. Note the SWAG-V methods better highlight the action of the clippers.

superpixels mixing into each other, compared to the other SWAG-V methods. SWAG-V$_{I}$ and SWAG-V$_{I+G}$ seem to have smoother superpixels that follow the motion more accurately. We see smilar results in the sheep shearing clip. Here our SWAG-V methods highlight both the clippers and a portion of the sheep. However, the SWAG-V $_{G}$ method produces superpixels that are, again, noisy in their shape. Here the superpixels appear quite diffuse compare to SWAG-V$_{I}$ and SWAG-V$_{I+G}$. Additional examples can be found in Appendix B.

4.5.2 Local Accuracy

Now that we have qualitatively shown that SWAG-V produces explanations that are better able to highlight an action both spatially and temporally, we now perform quantitative experiments. As with SWAG, we begin by exploring the accuracy of the explanation. Unlike SWAG however, we only show results for local accuracy. This is due to the inefficient method for measuring the global accuracy, that is the constant requirement to retrain the network. While this was computationally expensive for use with image networks, the additional temporal element and the size of the video datasets makes it infeasible for this thesis. As discussed previously, we therefore use the time that would have been spent running global accuracy tests, and instead run both insertion and deletion tests.

While we covered the deletion metric in the previous chapter, the insertion metric is similar in many ways, but seeks to highlight a different property of an explanation. Ostensibly both measure how well an explanation aligns to the regions of the image used by the network to inform the predicted class. Deletion measures how well an explanation can locate the pixels that are very important to the network. By removing these pixels, in order from most important to least, we would expect to see that the more accurately an explanation can locate the important pixels, the faster the network is unable to predict the class. The insertion metric works slightly differently by reversing the order in which the pixels are altered. Starting from a blank image (all pixels set to 0), the pixels are iteratively reintroduced most important first. In many ways this insertion metric is similar to a concept introduced by Dabkowski and Gal [10], called the smallest sufficient region (SSR). They used this to help build explanations that were compact and cohesive in nature. Indeed, in the qualitative examples the authors show, the SSR is a cohesive region, rather than a scatter of pixels the network finds important. In the metric introduced by Kapishnikov *et al.* [65] they modify the insertion metric so that rather than the percentage of pixels removed, they report the compressed size of the image. The intent being that the more the image can be compressed, the more cohesive the region being reintroduced.

This then sets up a conflict between the insertion metric, and the deletion metric. While the deletion metric favours the precision of an explanation to

accurately locate individual pixels that are important, the insertion metric rewards finding cohesive regions that are important. It therefore becomes a balance between how well a method is at the insertion vs deletion metric. It is likely that as a technique become more precise it will give better deletion results, while a technique that becomes more cohesive should give better insertion scores.

4.5.2.1 Implementation

To generate our insertion and deletion scores we first create an explanation for an input using one of our chosen methods. Using these explanations, we rank every pixel based on its importance to the network. For deletion, we begin with the video and remove pixels from most import to least. For insertion, we begin with a video containing only zeros and reintroduce the pixels from the original input, most important first. We introduce or remove pixels over 28 iterations. This equates to 7,168 pixels being introduced or removed at a time. While this is more pixels than introduced or deleted for the SWAG experiments (1,792 pixels at each iteration), it is necessary to ensure that the experiments are able to be run in an efficient and timely manner.

To generate local accuracy results, we use two architectures and the Kinetics 400 dataset. Weights for the models were imported from those released by Tran *et al.* as part of the R(2+1)D paper [38]. As is standard, we perform experiments on the spatial stream. We do not perform any experiments on motion streams. From each of the Kinetics 400 validation videos, we extract the centre 16 frames and use these to obtain our scores. This gives us 18,362 clips.

We compare our method against a number of activation based methods, namely Grad-CAM, Grad-CAM++ and Saliency Tubes. Saliency Tubes requires a network architecture where there is only one linear layer in order to generate the weights for the activations. As such, we do not test produced results for C3D using Saliency Tubes. Also, we compare against guided backpropagation to see how it compares against SWAG-V which uses guided backpropagation for weighting.

4.5.2.2 Results

We show the results for the insertion and deletion metrics in Table 4.2. From these results we can begin to discern a number of trends. The first is that the random baseline, as expected, performs poorly. It is the worst in the deletion metrics and the second worst in insertion. Interestingly, both baselines perform better than guided backpropagation in the insertion metric but much worse in the deletion. This again reinforces our belief that every explanation is a trade-off between precision explanations and cohesive ones. For C3D simply pointing at the centre of the clip produces better deletion results than Grad-CAM, although Grad-CAM produces better insertion results. This suggests that the methods based on activation maps sacrifice precision for a more cohesive explanation. This seems to be the case with these methods performing the best in insertion, while only beating the baseline methods for deletion.

When we look at the results for our proposed method we see that all three methods perform much better at the deletion metric than the activation based methods. However, for the insertion based metrics, SWAG-V does not perform as well. Indeed, for C3D, both baseline methods outperform SWAG-V_G with the insertion metric. This suggests that SWAG-V in general is more precise than the activation based methods, but not as cohesive. Grad-CAM and saliency tubes perform identically suggesting there is little difference between the explanations. Interestingly Grad-CAM++ outperforms Grad-CAM on everything apart from insertion with the R(2+1)D architecture. This is contrast to the previous chapter were Grad-CAM outperformed Grad-CAM++ on the local accuracy metric.

4.5.3 Weak-Localisation

As discussed in Chapter 3, there is a body of work devoted to weak-localisation of objects through the use of model interpretation techniques. As such, it has become common when introducing a novel interpretability technique to have a section of experiments that discuss how well the generated saliency map locates the given object within an image. In networks that deal with image classification this primarily takes the form of extracting some portion of an explanation and seeing how it aligns spatially with a bounding box. However, for video inputs we need to localise in both the spatial and temporal

TABLE 4.2: Deletion and insertion results for both C3D and R(2+1)D using Kinetics 400

	Deletion		Insertion	
Method	C3D	R(2+1)D	C3D	R(2+1)D
Centre	0.153	0.167	0.264	0.304
Random	0.222	0.265	0.272	0.322
Grad-CAM	0.157	0.118	0.313	**0.447**
Grad-CAM++	0.136	0.125	**0.315**	0.422
Saliency Tubes	–	0.118	–	**0.447**
SWAG-V_I	0.091	0.113	0.312	0.372
SWAG-V_{I+G}	0.087	0.111	0.314	0.381
SWAG-V_G	**0.068**	**0.080**	0.262	0.343
Guided Backprop	0.031	0.035	0.184	0.287

dimensions. To do this, we extend the weak-localisation method used in the previous chapter (established in [11, 64, 133]) for use with video inputs.

To begin to adapt this experiment for use with spatio-temporal inputs, a suitable dataset with corresponding bounding boxes is required. For this experiment, we chose UCF101. This is both a commonly available dataset, and crucially contains localisation annotations for 24 of the classes. It is these classes that we will use to measure the localisation abilities of the interpret ability techniques. This is also in-line with techniques for specifically generating action localisations in spatio-temporal inputs (i.e. bounding boxes through time) [157].

For simplicity, we filter out any videos from the validation set (containing 914 videos) that have more than one action to localise per frame, and any clips that have fewer than 16 contiguous bounding boxes present. This reduces the validation set to 697 videos. We crop videos into 16 frame clips, starting a new clip every 8 frames. If the 16 frame clip does not contain a ground truth bounding box for every frame, we ignore it. This generates a total of 10,704 clips for evaluation. To evaluate, we generate a bounding box for each frame in the clip, and get an IOU score for each frame. We then average these over the 16 frame clip. If the average IOU is greater than 0.5 we classify it as correct.

TABLE 4.3: Weak localisation results as % localisation error, where a lower score is more desirable. For each of the three thresholds tested, we present the average score as an indication of the overall ability of the method to weakly localise action.

	C3D			R(2+1)D		
	Val	Mea	Ene	Val	Mea	Ene
Centre	87.91	88.26	87.66	87.91	88.26	87.66
Random	90.51	90.63	90.34	90.51	90.63	90.40
Guided Backprop	90.62	90.69	100.00	90.68	87.58	85.64
Grad-CAM	90.57	90.46	90.53	85.40	87.08	85.32
Grad-CAM ++	90.21	89.91	89.92	85.40	87.08	85.32
Saliency Tubes	–	–	–	85.58	85.86	85.64
SWAG-V_I	74.00	74.46	74.26	71.76	72.61	72.58
SWAG-V_{I+G}	**73.00**	**72.52**	**72.99**	70.02	**69.94**	71.25
SWAG-V_G	75.22	74.25	74.09	**69.88**	70.80	**70.59**

4.5.3.1 Results

We show the results for this experiment in Table 4.3. From these results we can pick out some interesting observations. The first is that guided backpropagation performs poorly for both models, in some cases even being outperformed by the baselines. The second observation is how close the results are for the baselines compared to the CAM results. In particular, using the C3D architecture, the centre baseline is beaten only by our proposed SWAG-V methods, with even the random baseline outperforming them in some instances. The R(2+1)D network provides slightly different results, with the CAM techniques and saliency tubes slightly outperforming the baselines.

Given these results, our SWAG-V methods outperform all others tested by a large margin, often by around 15% to 20%. Of the three SWAG-V methods, SWAG-V_I performs the worst, with SWAG-V_{I+G} performing best with C3D, and SWAG-V_G the best with R(2+1)D. Compared to the difference with the other methods though, there is little difference between the SWAG-V methods, with around 2% variation.

4.5.4 Efficiency

We show the mean time taken to generate an explanation in Table 4.4. These are averaged over 1,000 explanations. From these results, we can see that

Table 4.4: Mean computation time in seconds

Method	C3D	R(2+1)D
Grad-CAM	0.08	0.15
G-CAM++	0.08	0.15
Saliency Tubes	–	0.05
SWAG-V_I	0.41	0.68
SWAG-V_{I+G}	0.46	0.74
SWAG-V_G	0.26	0.40

SWAG-V adds an additional computational overhead when compared to the alternative methods. As with the results for SWAG in the previous chapter (Section 3.7.5, we see that SWAG $_G$ is the fastest of the SWAG-V methods due to the simplicity of making the superpixels. Again, $SWAG_{I+G}$ is the slowest due to the overhead involved in combining the gradients and images. Despite the additional computational cost, we believe the improvement in the other metrics justifies the increase.

4.6 Future Work

We have seen that SWAG-V performs well with the architectures experimented on in this chapter. However, in future examination of SWAG-V we would like to experiment with two-stream architectures. This would involve the creation of explanations for both the RGB stream, as well as one based on dense optical flow. Doing so would allow us to create individual explanations for both the appearance (the RGB stream), and perhaps more interestingly, the motion. This is likely to provide additional insights into how an action recognition models uses the input to inform its prediction.

4.7 Chapter Summary

In this chapter, we have proposed SWAG-V, an extension of SWAG for the production of explanations for action recognition networks. Through experimentation we have shown that SWAG-V performs better in both deletion and weak-localisation metrics compared to other comparable explanation methods.

However, this comes at the cost of performance in the insertion metric. As with SWAG, this suggests that a trade-off between cohesion and understandability is necessary for improved accuracy. Using SWAG-V allows us to achieve this trade off for action-recognition networks.

There are a number of areas within this chapter that would have been beneficial to explore further given appropriate time and resources. The first would have been to explore how our methods performed in a global accuracy metric. This is something that would have been impossible without drastically increasing the resources or time available to us. Secondly since completion of this work, a weak-localisation metric that aimed at localising temporal regions has become more popular [41]. Running this metric would have allowed us to further understand how well SWAG-V is able to perform in weak localisation tasks.

In this and the previous chapter, we have focused on creating explanations based on combining superpixels and gradients. However, activation-based methods such as CAM and its variants remain popular and particularity effective at weakly-localising an object. In the following chapter, we introduce a method that allows us to improve upon the existing CAM techniques, allowing us to create more accurate explanations, that retain the interpretable quality of CAM methods.

CHAPTER

5 Jitter-CAM: Improving the Spatial Resolution of CAM Based Explanations

5.1 Introduction

In the previous chapters, we have explored explanations produced through combining superpixels and gradients. However, Class Activation Map (CAM) based methods remain popular for both explanations and tasks such as weak-localisation. SWAG in particular performed poorly at weak-localisation. Indeed, CAM methods are popular enough to have spawned a number of modifications, notably Grad- CAM [15], Grad-CAM++ [23], XGrad-CAM [70], Ablation-CAM [24], and Score-CAM [72]. The common advancement through all of these methods is in accurately weighting the activation maps prior to combining. However, as we have discussed in previous chapters, these activation maps are typically very coarse. There can only be so much progress with weighting the activation map before the inherent accuracies present become a limiting factor. based explanations. It is important to discuss briefly at this point the difference between activation maps and attention maps as the terms are often used interchangeable. At their most basic they are the output of a convolution layer. For example, Jetley *et al.* [158] suggest that attention maps are used either in post hoc analysis (as we are doing), or in trainable attention mechanisms. However, as the majority of CAM papers use the term activation maps [23, 24, 34, 72], we will also use this terminology.

In this chapter the key contribution is a method for generating explanations with an increased spatial resolution compared to existing CAM based methods.

This is achieved through rescaling the input image, and creating multiple CAM explanations from this larger image. These are then recombined into a high resolution explanation.

5.2 Related Works and Motivation

As we have discussed previously, a number of CAM based methods have been proposed that aim to produce explanations that better indicate the reason behind a networks prediction. Since the original CAM method [34] was restricted to network architectures that possessed a global average pooling layer at prior to the classification layers, this restricted the techniques adoption. However, Grad-CAM generalised this by backpropagating the gradients from a specific class output to the activation map. Weights for the activations were produced by taking the mean value of the gradients in such a way that each activation map had a corresponding mean gradient value. The intuition here is that the mean of the gradient will be indicative as to how useful the network finds a particular activation map to the output. Grad-CAM, Grad-CAM++, and XGrad-CAM all make use of the gradient component to create the weights for the activations.

Recently, there has been a trend towards a more perturbation based approach. This is seen in the Ablation-CAM, and Score-CAM techniques. In both these techniques, the network is perturbed and the output prediction score for the desired class used as the activation maps weight. With Ablation-CAM, the activation maps are extracted and iteratively 'turned off' by setting them to 0. This is similar to dropout in the zeroing of filters, but carried out in a more controlled and structured manner. When these activations are passed to the classification layers, it gives an indication as to how useful it was to the networks prediction based on the amount the classification score drops. Score-CAM is much more similar to RISE in its approach. It extracts the activation maps from the final convolution layer and iteratively multiplies the input image with each activation map so as to mask out regions not activated in the activation map. These are then passed to the network and the prediction score used as the weight for that activation map. These methods require as many passes through the network as there are activation maps to weight. In the case of VGG-16, this is 512 passes (the number of filters in the final convolution

layer). For more modern networks, the number of filters in the final layer has often significantly increased, for example ResNet [68] uses 2,048 filters in the final layer, while ShuffleNet [109] and MobileNet [110] use 1,024 and 1,280 respectively. The increase in filters subsequently increases the computational requirements needed to produce an explanation. In addition, it is notable that in Ablation-CAM [24], Desai and Ramaswamy suggest that their technique does not produce explanations as accurate as previous techniques when using networks that do not posses fully connected layers. Fully connected layers are now uncommon in newer networks such as ResNet [68], DenseNet [108], ShuffleNet [109], and MobileNet [110].

Despite all of the above advances in generating better methods of weighting the activations, a limiting factor of any explanation produced in this manner is the size of the activations. It is common for newer CNN architectures [68, 108–110] to produce a final activation map with a height and width of 7. VGG16 [67] is a somewhat of an outlier with a final activation map of 14×14. The size of the activation map is function of both the input image size, and the network architecture. The majority of networks use a 224×224 input image, however the architectures contain much more variation. Networks newer than VGG16 are typically able to have many more layers. This often requires the size of the activation maps to be reduced in size the further into the network they travel. This allows for these very deep networks to be run efficiently. When these coarse activation maps are resized to the input image size using bilinear interpolation, imprecision will be inherent in the explanation. If there was a method of creating activation maps with a higher resolution than 7×7 then the resultant explanations would likely be more accurate than before. In the following section, we propose a method of rescaling the input image to facilitate the production of multiple explanations of the network which can be combined to produce a high-resolution CAM explanation.

5.3 Jitter-CAM

As we have highlighted over the previous chapters, the limited resolution of the activation maps is a cause of concern given the prevalence of CAM based techniques. By offering a technique that allows us to increase the resolution of the CAMs, we believe that explanations can be produced that are more

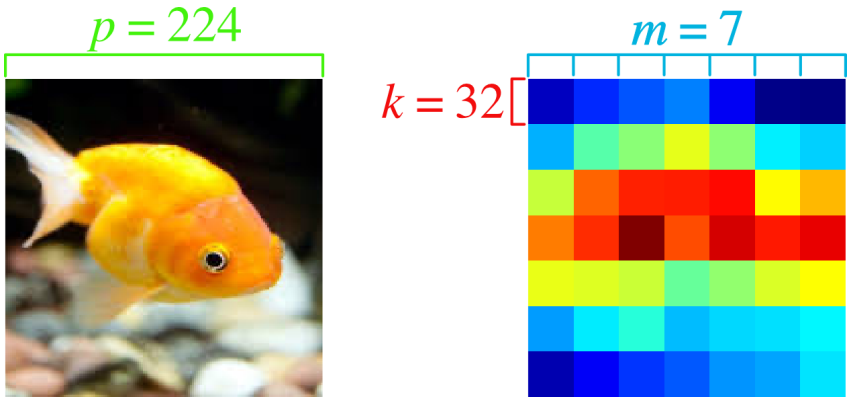

FIGURE 5.1: An overview of the relationship between the image and the CAM

accurate. While other methods have sought to create explanations with larger resolutions through the use of network perturbation, it is typically achieved through a large number of iterations (for example RISE using 8,000 passes). It is not the perturbations themselves that increase the resolution, rather the more perturbations, the more accurate the initial explanation is able to become. We propose a method based on increasing the scale of the input image, then producing explanations for patches of the new image corresponding to the original image size. These explanations can then be combined to produce a new explanation with an increased size. Whereas the original CAM prior to resizing is typically 7×7, we can choose any size of explanation with the understanding that as sizes increase, so do computation and memory costs required to produce it. Importantly however, compared to perturbation techniques, we find that Jitter-CAM is much more efficient. For example: 64 iterations to create a 14×14 explanation compared to 8,000 to create an 8×8 explanation using RISE. To summarise, we are able to increase the spatial resolution of existing CAM techniques to resolutions similar to those afforded to us by perturbation techniques, in a much less inefficient manner.

Before outlining our method we suggest the following notation for use throughout the section. Let the size of the activation map be defined as $m \times m$. Each element of the activation map corresponds to k pixels in the input image, where $k = p/m$. Here, p is the height or width of the input image (i.e. 224 pixels). An example of this relationship is shown in Figure 5.1. To increase the size of the CAM from $m = 7$ to $m = 10$, an increase, d, of 3, we would therefore

need to increase the size of the input image by dk pixels. We increase the size of the image using bilinear interpolation in an attempt to reduce artefacts in the image. We call this resized image I. A CAM is created by passing an image, X, to a CAM explanation function $E(X)$. $E(X)$ returns a CAM explanation of size $m \times m$.

A Jitter-CAM explanation can be achieved in relatively few steps. An overview of the following steps is shown in Figure 5.2. We begin by determining how much we would like to enlarge the size of the CAM prior to resizing. When this is determined, we resize the input image to the corresponding size. Example: suppose we would like to increase the CAM size, m, from 7 to 12, an increase value, d, of 5. For an input image of $p = 224$, this would be an increase of $5k$ where $k = 224/7$. This would give a new image size of 384×384. Patches of this enlarged image are taken corresponding to size $p \times p$, and a stride of k. For a CAM size increase of 5, this would give 36 patches ($= (5+1)^2$). As a side note, we approach the creation of the image patches in a structured way, creating a grid of patches from the resized image. It is entirely plausible that this could be achieved using random affine transformation as is used in data augmentation. However, this is not something that we attempted as part of this work, although it holds interest for future work.

To create a Jitter-CAM explanation, we need to create and combine CAM explanations from each of the patches. Extracting these patches from the resized image I and creating a combined explanation J (of size $(m+d)\times(m+d)$) is done as so:

$$J_{i:i+m,j:j+m} = J_{i:i+m,j:j+m} + E(I_{ki:ki+p,kj:kj+p}), \tag{5.1}$$

where i, j is the starting location for the explanation to be added to J. Here $i, j = [1...d+1]$. Multiple explanations will be created at any given point, with more being created corresponding to centre regions. To account for this, we create a count of how many times each region of the image has had an explanation created for it. We call this C, a matrix of size $(m+d)\times(m+d)$, and define it as:

$$C_{i:i+m,j:j+m} = C_{i:i+m,j:j+m} + 1. \tag{5.2}$$

Again, $i, j = [1...d+1]$. Finally, we produce the pre-resized Jitter-CAM explanation:

$$\text{Jitter-CAM} = \frac{J}{C}. \tag{5.3}$$

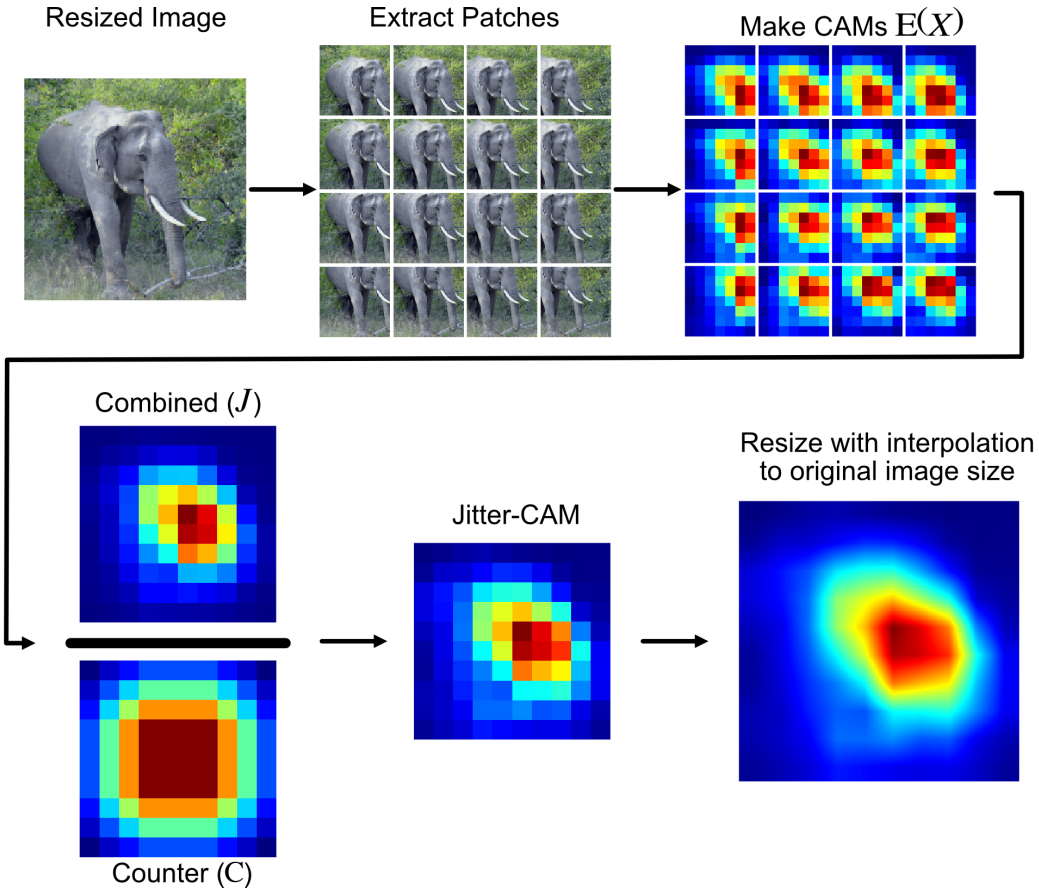

FIGURE 5.2: A high level overview of Jitter-CAM. This example resizes to a new CAM size of 10×10, which uses 16 image patches.

Given this higher resolution CAM, we can now resize it to the original input image size using bilinear interpolation to give us a familiar looking CAM explanation.

The key parameter for this technique then is by how much we increase the size of the CAM. If we increase it too much, we run the risk of the explanations not being exposed to enough of the image and, therefore, unable to produce explanations for the object in the image. Too small and we may not offer any improvement over the standard Grad-CAM method. In Figure 5.3, we show an example of how the increase in the CAM size affects the final explanation. Here, we can see that the original Grad-CAM is fairly coarse, highlighting a number of regions outside of the target object. However, as we begin to increase the resolution of the CAM, we see that the regions marked as important become

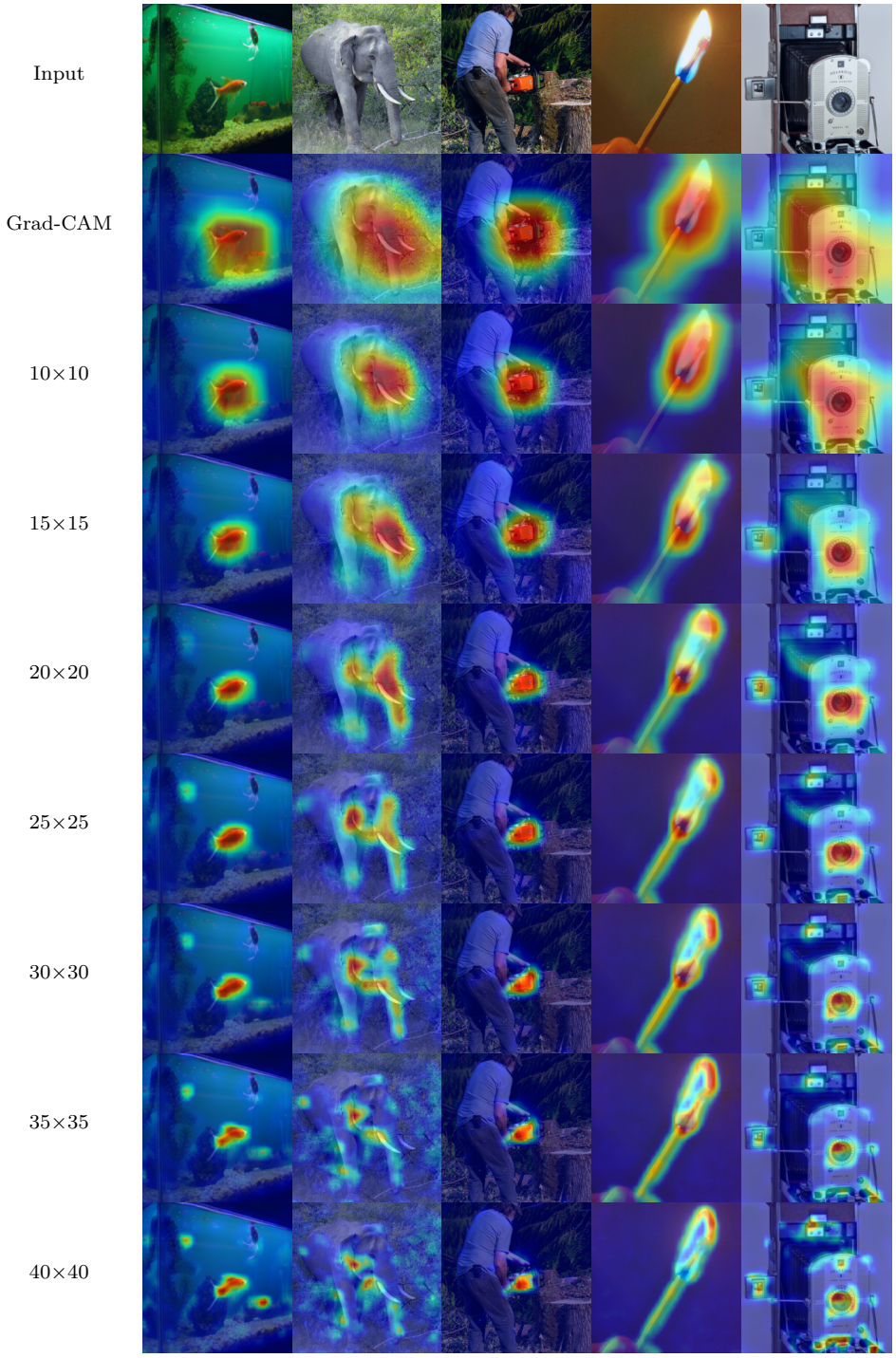

FIGURE 5.3: Qualitative comparison between differing sizes of CAM and the original Grad-CAM. Examples taken from ImageNet and explanations produced using ResNet50.

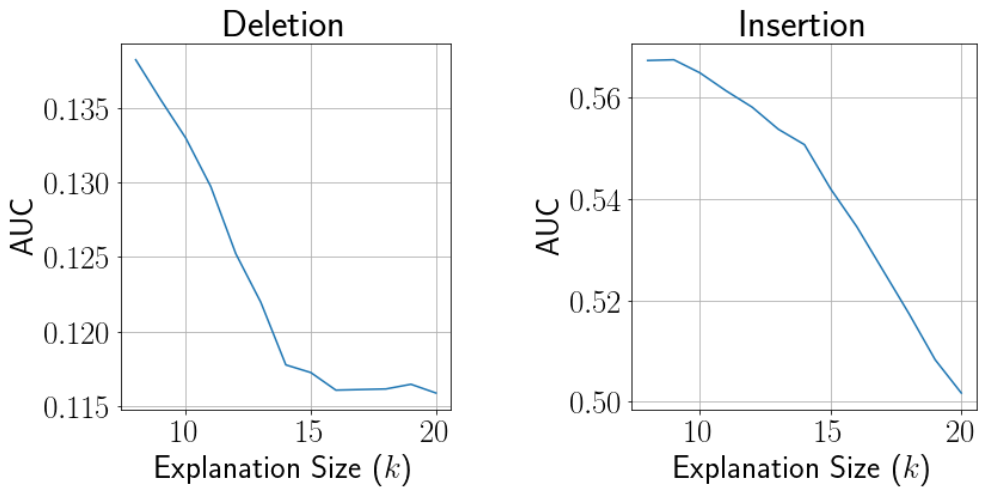

FIGURE 5.4: Showing how the deletion and insertion scores are affected by the value of the dimension (k) of the CAM

less coarse and more detailed. For example, notice how the explanation for the elephant class begins to form around the trunk when resized to a 15×15 CAM. As the CAM continues to increase in size, we begin to see the explanation become less cohesive. This is because the network is being given smaller and smaller regions of the original image to produce explanations for. This figure also demonstrates that what seems to be an optimal resolution for one image, may not be the best for another. For example, the goldfish seems to be better explained using a much larger resolution compared to the elephant. The best we can hope to achieve, without setting the spatial resolution for each explanation individually, is to find an optimal size that works well for the entire dataset in use.

How then do we find the optimal size for a Jitter-CAM explanation? As with the previous chapter, we are driven by the use of the deletion and insertion metric from RISE. We experiment using the ResNet50 network, and vary the size of the CAM from $k = 7$ (the original Grad-CAM) to $k = 20$. We then show the corresponding deletion and insertion charts in Figure 5.4. It is important to note here that all results in this chapter are based on ImageNet, as such, these results are currently dataset specific. In the future it would be useful to see how consistent these results are across multiple datasets. As we saw with SWAG-V, the insertion metric decreases as the deletion metric increases.

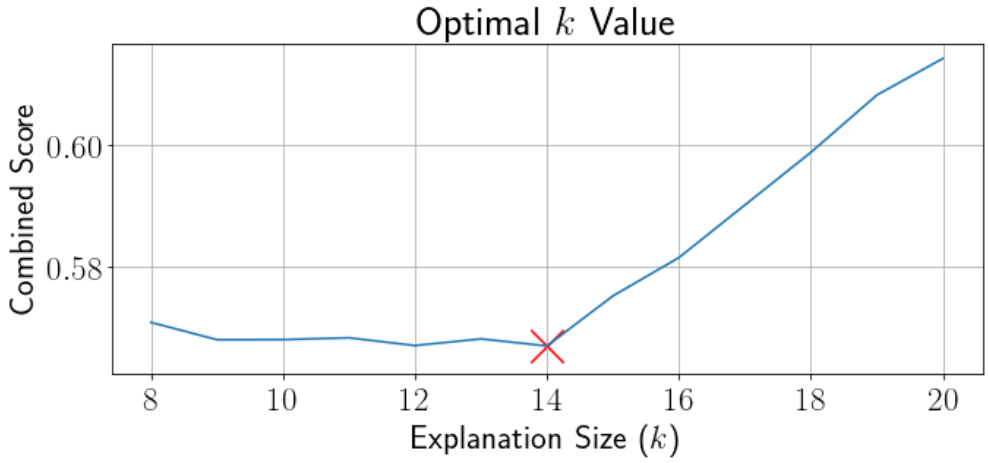

FIGURE 5.5: Finding the optimal value of k using ResNet50 by combination of the insertion and deletion scores. Here we find that $k = 14$ is the optimal value (double the resolution of the original explanations k.

However, unlike SWAG-V, where the deletion results continues to improve as we make the superpixels smaller. Here, the deletion score plateaus around $k = 14$. As with SWAG-V, we determine the optimal k value by combining the insertion and deletion scores together: (deletion + (1−insertion)). Taking the k value at the lowest position should therefore give us the optimal size for out Jitter-CAM explanations. These results are shown in Figure 5.5 with the lowest point marked. Here, we see that $k = 14$ seems to be the optimal size, double the initial size of the original CAM explanations from a ResNet50 model. For the Inception architecture [145] which has a final activation layer of 8×8, we experimented with increasing it to 14×14, and 16×16. We found that Inception gave better results when being explained using a Jitter-CAM explanation of 16×16.

When we refer to Jitter-CAM it will be with an explanation of double the original explanations size. This gives ResNet50 and DenseNet a 14×14 CAM prior to resizing, and Inception a 16×16 CAM.

5.4 Experiments

In this section, we present the experiments we will run and their associated results. We run a number of experiments we have already discussed in this thesis, namely local accuracy, weak localisation, and efficiency, alongside additional experiments that are common in the CAM literature. All experiments are conducted using the ImageNet validation set (50,000 images) over a variety of models. We use the following pre-trained models from the PyTorch model zoo: ResNet50 [68], DenseNet121 [108], and Inception V3 [145]. The first two networks produce CAM explanations of 7×7 prior to resizing, while the Inception network produces one that is 8×8.

We hypothesise that as Jitter-CAM is able to increase the spatial resolution of an explanation, local accuracy metrics should become more robust. We expect that Jitter-CAM will produce good deletion results, but still be outperformed in the insertion metric. This is expected from an explanation technique that increases the spatial resolution compared to coarse explanations. We expect Jitter-CAM to perform well in weak-localisation metrics. CAM methods typically perform well at localisation tasks, and we expect that Jitter-CAM will be able to improve upon this underlying performance by better adhering to object boundaries thanks to the increased spatial resolution.

5.4.1 Qualitative Inspection of Results

We begin with a qualitative assessment of the explanations. Examples of each of the CAM methods tested are presented alongside Jitter-CAM examples. These are shown in Figure 5.6. Here, we see that using Jitter-CAM to double the size of the explanation prior to resizing, allows us to produce an explanation that is more compact around the object in the image. We see this clearly in the first image of a goldfish, where our Jitter-CAM explanation focuses less on the surrounding tank, and more on the goldfish itself. Also notice, in cases such as the gymnast or boat, how Jitter-CAM manages to highlight more of the object than previous methods. Note, that in examples where more regions of an object are highlighted, Jitter-CAM still adheres strong to the object boundary than previous CAM methods. This can be seen in the gymnasts leg or the boat. Perhaps the most striking aspect of the qualitative comparison is how visually similar all the previous CAM methods are. This is likely because they are

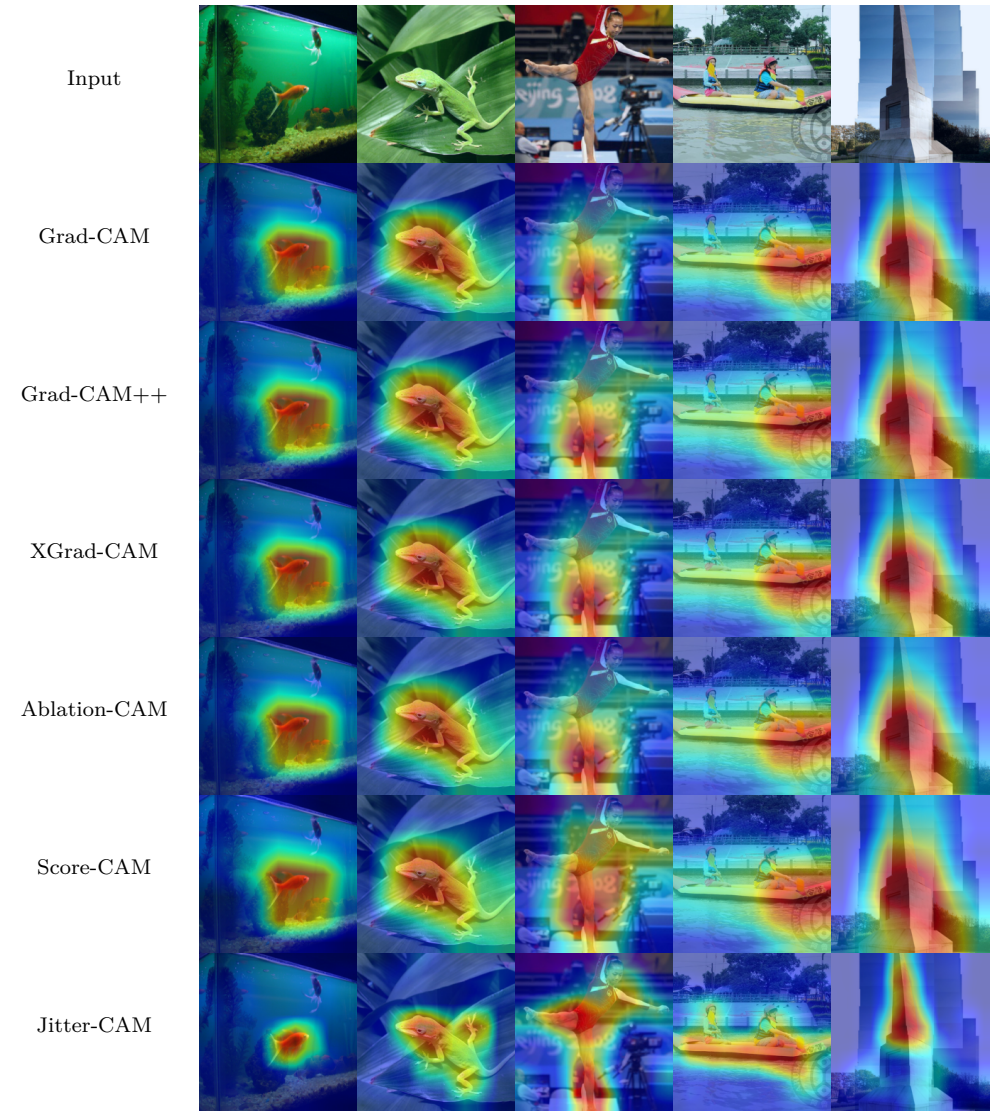

FIGURE 5.6: Qualitative comparison between various CAM methods and Jitter-CAM. Examples taken from ImageNet using ResNet50 [68].

all constrained to using the same activation maps to form their explanations, and improvements in scoring their importance will only go so far. Additional examples for the other architectures tested can be found in Appendix C.

In the following sections, we will show that although the previous CAM explanations are visually similar there are quantitative differences. We will compare these to Jitter-CAM and show the strengths of our proposed method.

5.4.2 Faithfulness

We begin by discussing faithfulness, a metric that is often deployed in CAM research [23, 24, 159]. In this section, we show that it is *not* an appropriate measure of an explanation, and perhaps should be discontinued as a metric in future research. First introduced by Chattopadhay *et al.* [23], the faithfulness metric was intended to be a measure of how well the regions deemed important by an explanation aligned to those used by the model. This is achieved by performing a point-wise multiplication of the explanation and the input image to create a masked image E^c for class c:

$$E^c = L^c \circ I. \tag{5.4}$$

Here, L^c is the original explanation for class c scaled between 0...1 and I is the input image. The masked image, E^c is then passed to the model and the change in the softmax results observed. This softmax score is then used to inform two measures of faithfulness, namely the average drop (AD), and the increase in confidence (IIC).

AD is a measure of how much the models confidence drops when shown the masked image compared to the original image. The intention is that a good explanation should highlight the regions of an image important to a network, and give low scores to those that are unimportant. Therefore, E^c will be an image where useful regions are kept, and those not seen as useful are suppressed. AD is then measured as:

$$\frac{1}{N}\sum_{i=1}^{N}\frac{\max(0, Y_i^c - O_i^c)}{Y_i^c} \times 100. \tag{5.5}$$

Here, Y_i^c and O_i^c are the models softmax output for class c and the i^{th} input image, and it corresponding masked image respectively. N is the number of images in the dataset. For an AD, a lower value is desirable as it reportedly indicates that the masked image has only kept useful regions.

IIC is the complement to this and seeks to determine if the masked image results in the models confidence increasing. The idea behind this is that the explanation has the potential to mask out regions of the image that are

detrimental to the networks prediction. Removing these should, therefore, increase the networks confidence. IIC is given by:

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{1}[Y_i^c < O_i^c] \times 100, \tag{5.6}$$

where $\mathbb{1}$ is an indicator function equal to 1 if the condition in brackets is true, 0 otherwise. For IIC a large value is desirable.

It is notable that the CAM methods that score well on this metric cover a large region of the image with a heatmap. It transpires that this is not coincidental. What these metrics are actually rewarding is having a large number of constant high valued pixels. When all pixels of the original image are masked evenly, it has the effect of only causing the confidence to drop slightly (as the image stays relatively the same as the original) and perhaps more interestingly, increases the model's confidence. To test this, we propose a baseline consisting of an explanation which is completely made up of values of 0.9, except for two random pixels with a value of 1 and 0. These random pixels ensure that during any rescaling, the constant values are not scaled to either 0 or 1. We label this baseline as 'Constant'. In Table 5.1, we show this baseline compared to Grad-CAM and Grad-CAM++. Here, we can clearly see that our constant baseline scores exceptionally well compared to the CAM techniques, while also clearly being a terrible explanation method. While it made sense that the AD metric would improve as multiplying the input image be a constant close to 1 should produce little deviation in the models prediction. It is not immediately obvious though why the IIC score should improve. We, therefore, extracted an additional measurement from the IIC of the mean amount of improvement. This is measured as $O_i^c - Y_i^c$ for any explanation that showed an increase the model confidence. We found that the reason that the constant baseline was able to improve so much over the CAM methods was that it improved more images, but with a lower mean improvement. The baseline improved the confidence of 24,094 images, with an average confidence increase of 0.022. Grad-CAM and Grad-CAM++ improved the confidence of 20,103 and 19,348 images respectively with mean increases 0.091 and 0.081. However, none of this is represented in these faithfulness metrics. This suggests that they should be discouraged from future use as they are, at best, misleading and could lead to the future development of explanation methods that are inappropriate.

TABLE 5.1: Faithfulness scores for (IIC) and (AD). For IIC higher is better, for AD lower is better. We see that the baseline outperforms all methods suggesting this is not a good metric for explanations.

	ImageNet					
	ResNet50		DenseNet		Inception	
Method	IIC	AD	IIC	AD	IIC	AD
Grad-CAM	40.21	14.04	39.65	13.22	40.46	16.98
Grad-CAM++	38.70	14.22	37.54	13.69	38.89	17.02
Constant	**48.22**	**2.70**	**49.14**	**1.08**	**47.36**	**1.32**

5.4.3 Local Accuracy

Faithfulness is often presented alongside the local accuracy measure of deletion and insertion introduced alongside RISE [9]. However, there is a crucial difference between the two types of metric that is important to measuring explanations. Both measurements alter the input image based on the explanation, but while the faithfulness metric simply performs a point wise multiplication (which we have shown is a poor method), deletion and insertion masks pixels iteratively according to their importance. Masking the input image in this way allows us to understand how well the explanation ranks the pixels importance. Indeed, this form of metric has become much more common compared to faithfulness, with a number of variants of removing or inserting pixels based on their importance being introduced [9, 10, 58, 65, 102, 160]. As with the previous chapters, we perform the metric over 28 iterations. The results for both insertion and deletion are shown in Table 5.2. From these results we can see that Jitter-CAM is much better at the deletion metric than the other CAM methods, but this is achieved via a trade-off with the insertion metric. This suggests that Jitter-CAM is able to better locate the pixels deemed most important to the models prediction, but in doing so, is less able to determine which pixels are required when rebuilding the image from scratch. Indeed, Qi *et al.* [161] found similar results in their work, showing that as deletion scores improve, typically insertion scores fall.

5.4.4 Weak Localisation

CAM-based methods are often used for weak localisation tasks. Indeed, the original CAM method was primarily aimed at the localisation of objects with

TABLE 5.2: Area under the curve for the local accuracy experiment. Lower is better

	ImageNet					
	ResNet50		DenseNet		Inception	
Method	Del	Ins	Del	Ins	Del	Ins
Random	0.303	0.413	0.280	0.382	0.287	0.425
Centre	0.177	0.420	0.172	0.374	0.165	0.460
Grad-CAM	0.142	**0.576**	0.137	0.536	0.128	**0.585**
Grad-CAM ++	0.147	0.564	0.141	0.524	0.132	0.572
XGrad-CAM	0.142	**0.576**	0.137	**0.540**	0.128	**0.585**
Score-CAM	0.150	0.568	0.142	0.532	0.131	0.579
Ablation-CAM	0.144	0.567	0.140	0.531	0.129	0.577
Jitter-CAM	**0.118**	0.551	**0.120**	0.522	**0.107**	0.570

its explanation properties as an additional trait. The ability of CAM-based methods to localise an object well has subsequently been used in number of tasks such as segmentation [162–164], object recognition [165], and person re-identification [166]. As such, weak localisation metrics are often used in CAM papers. In this chapter we present two weak-localisation metrics. The first is the weak-localisation metric we used in the SWAG chapter, which uses ground truth bounding boxes for the ImageNet validation set. In this method, the explanation is threshold using one of three methods, and a bounding box drawn around the thresholded explanation. This is done over a range of thresholds and the best score for each is presented. The first method of thresholding is based on scaling the explanation between 0 and 1, then sweeping through a range of thresholds in range $[0 : 0.05 : 0.95]$. This is labelled as 'Val'. The second set of thresholds is obtained by multiplying the mean value of the explanation with a value in the range $[0 : 0.5 : 10]$. This is labelled as 'Mean'. The final method is based on thresholding the heatmaps by the percentage of energy that covers a subset of the explanation in range $[0 : 0.05 : 0.95]$. This is labelled as 'Eng'.

The results for this three methods of thresholding are shown in Table 5.3. From these results we see that for ResNet50 and DenseNet there is a dramatic improvement over previous CAM techniques, improving by around $2\% - 5\%$ depending on thresholding technique. The likely reason for this improvement in localisation ability is seen in the qualitative results. Here, we see that Jitter-CAM is able to better highlight more reasons of the object than previous

TABLE 5.3: Weak-localisation results as % of localisation error. Lower is better.

	ResNet50			DenseNet			Inception		
Method	Val	Mea	Eng	Val	Mea	Eng	Val	Mea	Eng
Random	57.43	58.96	57.39	57.43	58.96	57.39	57.74	59.10	57.66
Centre	47.58	48.18	47.68	47.58	48.18	47.68	48.56	48.34	47.84
Grad-CAM	45.94	45.89	44.35	45.44	44.99	43.48	44.84	45.29	44.60
G-CAM ++	45.76	45.83	43.85	44.89	44.88	42.88	45.05	44.94	44.87
XGrad-CAM	45.94	45.89	44.35	45.67	45.38	43.96	44.84	45.29	44.60
Ablat-CAM	45.88	45.88	44.30	45.52	45.26	43.71	45.25	45.34	45.00
Score-CAM	47.53	46.86	45.32	47.29	46.35	44.74	45.72	45.59	45.19
Jitter-CAM	**39.83**	**42.30**	**40.64**	**40.24**	**41.55**	**40.44**	**38.38**	**39.10**	**39.86**

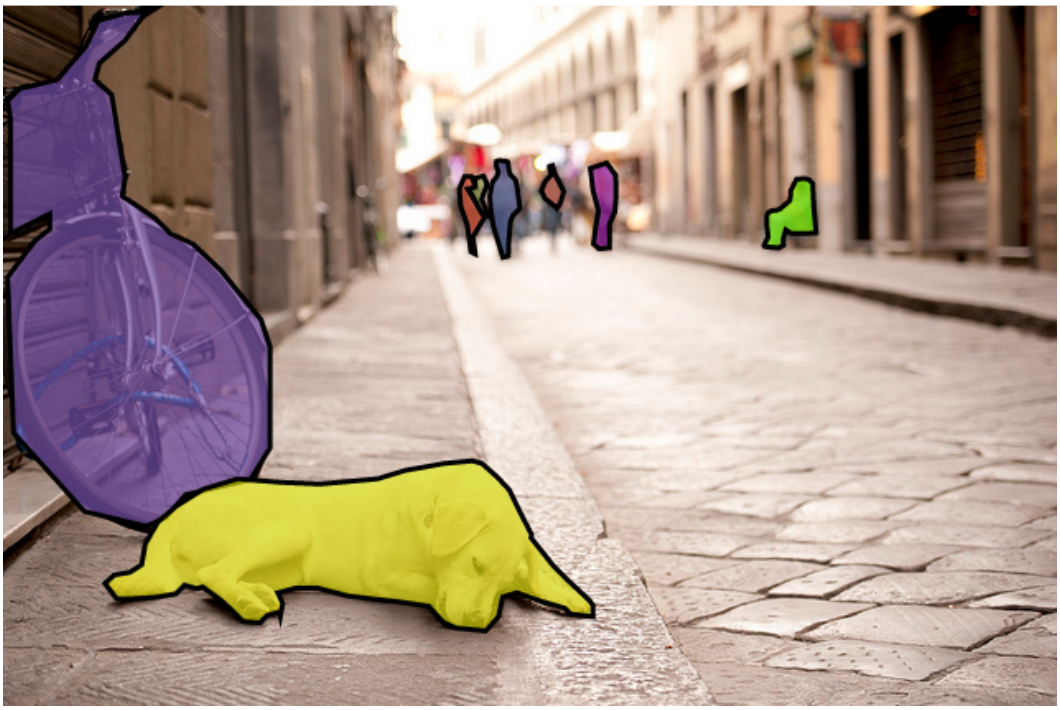

FIGURE 5.7: An example of a difficult image used in the pointing game. Three classes are shown in the image: dog, bike, and person. An explanation must point to within the coloured area to successfully locate the object.

CAM methods. This in turn results in regions that, when thresholded, better align to the groundtruth bounding boxes.

5.4.4.1 Pointing Game

The second weak-localisation metric that is often presented is the pointing game first introduced by Zhang *et al.* [64]. This takes a different approach to the previous weak localisation metric in two ways. The first is that instead of ImageNet, the COCO dataset [122] is used. This is a dataset of images that each can contain multiple objects in each picture. There are accompanying annotated regions for each object in the images. For every class of object in the image, an explanation is made for that class. The second difference is that rather than threshold the explanation to find regions which overlap with the ground truth bounding box, the maximum point on the explanation is used instead. This maximum point is said to be a hit if it falls on one the correct annotated region in an image (within a 15 pixel margin of error), otherwise it is considered a miss. The accuracy of an explanation method is then calculated as:

$$Acc = \frac{\mathrm{h}}{\mathrm{h} + \mathrm{m}}, \tag{5.7}$$

where h is the total number of hits, and m the total number of misses. One final aspect to the pointing game is that results are presented on all of the dataset, and a difficult subset of COCO images. For images to be included in the difficult subset they must meet two criteria. The first is that the total number of labelled regions for location must be smaller than 25% of the image by area. An example of a difficult image is shown in Figure 5.7. The second is that there must be more than one other object from another class in the image to locate. This results in a set of images with multiple differing classes of small objects. An example of the explanations created and the subsequent localisation of the target object is shown in Figure 5.8.

To implement the pointing game, we used the code released by Fong and Vedaldi [58] along with their pre-trained ResNet50 model. The pointing game uses images from the COCO dataset without resizing them prior to input. To accommodate this, the model is altered to be fully convolutional by changing the linear layer to be 1×1 convolutions. While this does not effect the generation of CAM-based explanations, it does affect the size of the explanation map. While, previously, we would always encounter a 7×7 activation map, the COCO images are scaled to only have their smallest dimension be 224 pixels.

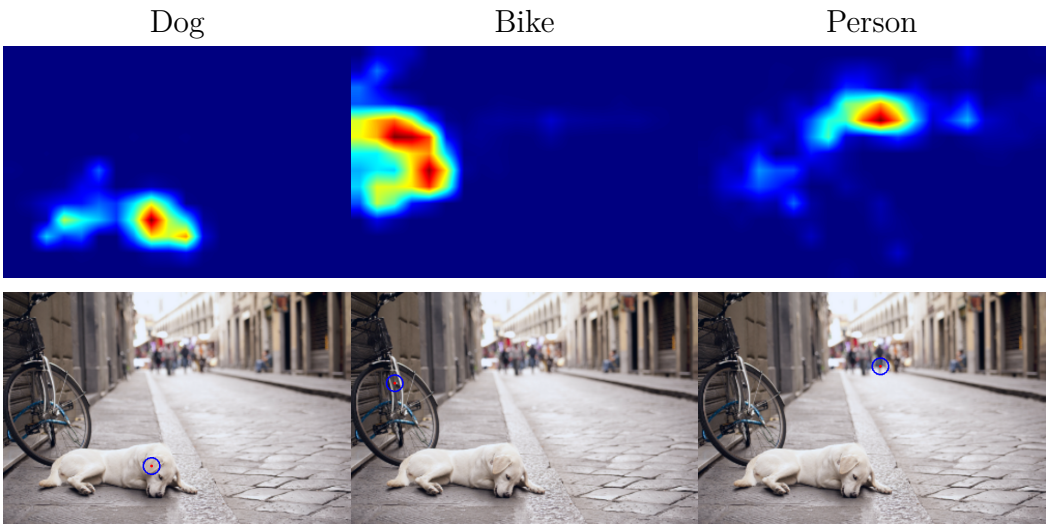

FIGURE 5.8: An example of the results for a single image (from Figure 5.7) in the pointing game using Jitter-CAM. Top: An explanation is created for each class in the image. Bottom: The red dot is the maximum point of the explanation, with the blue circle being the 15 pixel margin of error.

This results in activation maps where only one dimension is of size 7. To allow Jitter-CAM to work around we simply double the size of whatever activation map is produced. For example, an image of size 224×292 would create an activation map of size 7×9, therefore our subsequent Jitter-CAM explanation would be 14×18 prior to resizing with bilinear interpolation. We were unable to get Ablation-CAM to work using this method due to the change of the final linear layer to a fully convolutional one. In making the change, the pre-softmax score all became negative which broke Ablation-CAMs method of creating the weights.

The pointing game results for both the complete and difficult dataset using ResNet50 can be seen in Table 5.4. From these results we can see that as with the previous localisation metric, Jitter-CAM offers significantly better localisation abilities than previous CAM methods. By increasing the resolution of the CAM prior to being resized, we are able to be much more precise in where we can point to. Previous CAM methods are constrained by the coarseness of the produced explanation. Interestingly Jitter-CAM also achieves very good results for the difficult subset of COCO images, easily surpassing the previous methods scores for either the full or difficult sets.

Table 5.4: Pointing game results. High is better. Jitter-CAM outperforms all other CAM methods easily in both categories.

Method	ResNet50	
	All	Difficult
Centre	24.61	17.93
Random	12.33	7.99
Grad-CAM	53.47	49.40
G-CAM++	47.26	42.93
XGrad-CAM	53.46	49.39
Score-CAM	47.28	42.70
Jitter-CAM	**64.08**	**61.10**

5.4.5 Efficiency

Using Jitter-CAM introduces an element of inefficiency to the process of creating an explanation due to the multiples CAMs required to be generated. We defined the number of CAMs required previously as the difference in the original CAM size and the desired size. Using ResNet50 (which would give a 7×7 CAM) and our proposed Jitter-CAM size of 14×14 would require 64 $(= (14-7+1)^2)$ CAMs to be created. In contrast, Grad-CAM, Grad-CAM++, and XGrad-CAM only require a single forward and backward pass, while Score-CAM and Ablation-CAM require as many passes as there are activations in the final layer. For ResNet50, this equates to 2,048 forward and backward passes. Our implementation in PyTorch of Jitter-CAM, Score-CAM and Ablation-CAM makes use of batching to create an explanation more efficiently. For this test, we use a batch size of 32.

In Table 5.5, we show the mean time in seconds taken to compute a single explanation. We compute this average time over 1,000 images. As expected, the methods requiring only a single pass (Grad-CAM, Grad-CAM++, and XGrad-CAM) are the most efficient to computer. Score-CAM is by far the slowest method, even with batching implemented. This is due to the large number of images required to be generated and passed through the network. Ablation-CAM, despite also being a perturbation method, is much faster. This is because the scores for the activation maps can be computed efficiently by only perturbing the final activations and passing them to the classifier. Score-CAM

Table 5.5: Mean computation time in seconds

Method	ResNet50	DenseNet	Inception
Grad-CAM	**0.03**	**0.07**	**0.06**
G-CAM++	**0.03**	**0.07**	**0.06**
XGrad-CAM	**0.03**	**0.07**	**0.06**
Score-CAM	3.83	1.92	5.01
Ablation-CAM	0.60	0.30	0.60
Jitter-CAM	0.37	0.40	0.67

on the other hand requires the input image to be altered and passed through the entirety of the network.

Jitter-CAM is slower than the single pass methods, performing similarly to Ablation-CAM, but much faster than Score-CAM. We believe that the additional time increase compared to the single pass methods is justified by the improvements to the accuracy and weak-localisations metrics.

5.5 Future Work

The use of Jitter-CAM has shown that we are able to increase the spatial resolution of a CAM explanation prior to it being resized. It would be interesting to see if we could continue this work for use with video inputs as we did with SWAG-V. Potentially, the techniques discussed in this chapter, could also be used to increase the resolution of the temporal dimension when applied to video explanations.

5.6 Chapter Summary

In this chapter, we have proposed Jitter-CAM, a novel method that allows us to improve the spatial resolution of explanations created existing using CAM techniques. Rather than spend resources trying to improve the accuracy of the activation layer weights, we instead rescale the image and take multiple explanations. These are then combined into a single explanation.

Through both visual inspection and quantitative measurement we show that this technique improves local deletion accuracy, and greatly improves weak-localisation ability.

A limitation of this chapters work is that we could have conducted both global accuracy metrics, and additional experiments to see how well Jitter-CAM can replace other CAM techniques. Additionally, Grad-CAM is often used in other techniques for it's localisation ability. For example, in the adversarial erasing work by Wei *et al.* [162] , Grad-CAM is used as a guide to remove regions before retraining to create a more robust network. It would have been insightful to repeat this process using Jitter-CAM to see if the benefits transfer for a task such as this.

While this and the previous chapters have focused on creating explanations that aim to identify the regions of an image that have been used to classify an image correctly, what happens when the image is misclassified? Current methods for creating explanations do not take this into account and therefore provide no, or limited, insight into the reasons a network may fail to make a correct prediction. In the next chapter, we discuss this further and propose a method for approaching this problem.

CHAPTER

6 Explaining Failure Using Surprise and Expectation

6.1 Introduction

In this chapter, we discuss the limitations of current explanation techniques in identifying reasons for a model to misclassify an image. We propose a novel technique for building an understanding of how a model has learnt to represent the training data, and then leverage this understanding to identify reasons for failure. In this context, we understand a model by gaining an insight into how individual filters in the final convolution layer react to images from the training set and build distributions for each image class and filter pair. The final convolution layer is primarily chosen as the activations maps produced by it are known to be indicative of the networks predictions. This is the reason they are used in CAM explanation methods. When unseen data is passed through the network, we can then compare the filter reactions to the distribution from the training data. For the sake of clarity, we highlight that the term filter and channel are often used interchangeably in the literature. However, in this context they both represent an element of a convolution layer that outputs an activation map.

This novel technique is a different approach to understanding a network than has previously been taken in this Thesis. Rather than explicitly trying to find regions of the input image that are important to a networks prediction, we are instead attempting to identify the importance of individual convolution filters to the networks output. Using these insights we introduce two concepts

based on how the filters react: expectation and surprise. We define expectation as when a filter expected to be activated highly, does not. Conversely we define surprise as when a filter that would not typically have activated highly, does. To assign a quantitative value to how much a filter reacts, we propose a method that allows us to assign a value to a filter when an image is passed through the model. This score shows us whether a filter is beneficial or detrimental to the prediction. Using the above technique allows us to both identify reasons for misclassification, and visualise them using a modification of our SWAG technique.

We discussed in the literature review the need for a technique that is able to take elements from both input centric and network centric technique to allow for a better appreciation of the causes for failure. We hypothesise that a greater understanding of the individual features used by the network can be obtained by bridging the gap between methods such as Grad-CAM that present an understanding of the final convolution filters for a single image, and prototypes and criticisms which finds trends across the entire training dataset. We propose a technique that passes through every image from the training data and observes how the filters in the final convolution layer react. Of particular interest is how the filters react to different classes. As the network has learnt to represent the classes based on features within the image, we should see similar reactions from the filters to images of the same class. By building a distribution for each individual and class pair, we would be able to build an understanding of how images from the same class react and use this to begin to find outliers.

In this proposed setup, an image typical of one found in the training dataset should produce none or minimal outliers across the filters. However, if an image is atypical of the training dataset, we should begin to see outliers occur. We believe this will start to provide questions to the answers posed earlier with regards to a lack of features, or a distracting feature causing failure. If an unseen image causes a filter to have a strong reaction compared to the distribution from the training data, then this suggests that there is some feature in the image not typically seen in the training data. If an unseen images causes a filter to have a very weak reaction compared to the training data distribution, then this suggests that some feature that is typically present is missing from the image.

6.2 Measuring Surprise and Expectation

So far we have only discussed our technique in broad terms. In this section we more concretely define the overview of the algorithm, and detail the processes of each step. The core idea of our method is a two stage process:

- Computing the filter scores.
- Building distributions.

The first stage is to pass all of the images from the training dataset and generate a score for each individual filter in the final convolution layer. We store these scores alongside all of the others from the same class. This gives us a separate distribution of scores for every filter and class pair. The second stage is to define values that allow us to find outliers when using an unseen image. In the context of this chapter, we use unseen image to mean any image that has been taken from the ImageNet validation set. In this way it is distinct from the images used to train the network, but still allows us access to the ground-truth class label.

6.2.1 Grad-AMap: Filter Importance Measure

Perhaps the first important aspect to discuss is how we assign a value to each filter in the final convolution layer when an image is passed through. We refer to this as a filter importance measure. It is important to note that in this Chapter the filter importance measure itself does not give the surprise and expectation insight. Rather, the understanding gained by assigning a score to every filter in a network for each image/class combination allows us to gain this insight. Numerous methods existing for assigning some value to a filter to gauge its importance, either in the context of a specific prediction, or to the network as a whole. We find examples of filter scoring techniques in the explanation literature [15, 23, 70] (specifically CAM related techniques) and network pruning literature [167–169]. However, while both of these camps have a common aim of assigning some value to each filter, their motives are quite different. Scoring methods found in the explanation literature focus on trying to find how important a specific filter is to a given input. This is because the filter reacts to features within the image, and a strong reaction indicates an important feature that is worth highlighting. Scoring methods found in the network pruning literature have two main differences. The first

is that the scoring methods used are looking to forward how important the filter is to the network as a whole, not just for an individual prediction. The second is that retraining is an important part of network pruning, so filters must be scored in such a way as to also take into account their effect on the stability of the network. As we have noted in previous chapters, CAM based techniques are a common way of producing visualisations. At the heart of the method is the weighting of the activation maps prior to visualisation. The different techniques for producing the weightings is the primary variation between all of the CAM techniques. More modern CAM methods, such as Score-CAM [72] or Ablation-CAM [24], iteratively perturb the feature maps to produce a weighting. The computational requirements of these methods mean they are unsuited to our proposed method where we need to obtain scores for every image in a training dataset. A better method from the CAM techniques is to compute the filter scores using a combination of the gradients and the activation maps. It is important to note that when discussing the CAM techniques, that all of the gradients are back-propagated.

The first of these techniques is Grad-CAM [15]. Here, the mean of the gradient at each filter is used as the weight for each activation map. Using the mean of the gradient gives an idea of how important the activation map produced by the filter is the network, a high mean gradient would suggest the network finds it useful for discrimination, while a low value suggests it does not.

Grad-CAM++ [23] was proposed as a method to provide both better visualisation explanations (although we see from accuracy results in Chapter 3 this was not the case) and better weak localisation performance (which we confirm in Chapter 3). To obtain the weights, only the positive gradient is used in combination with a weighted activation map.

Axiom-based Grad-CAM (XGrad-CAM) [70] was proposed to satisfy two axioms: sensitivity and conversation. Sensitivity suggests that each point of the explanation should be equal to the change observed when removing that point from the input image. Conservation suggests that the sum of all the explanation responses should match the magnitude of the model output, that is the sum of the explanation should equal the prediction score. To achieve this, Fu *et al.* [70] propose a method of determining the weights by taking the

sum of the product between the activation maps and the gradient maps. To ensure the conservation axiom is met, the activation map is normalised.

Activation based explanation techniques are not the only methods for determining the importance of filters, another applicable area is network pruning. The goal of network pruning is to iteratively remove filters and retrain a pre-trained network in order to produce a much more compact network that offers similar prediction properties to the original. Unlike the discovery of weights for a visual explanation, these methods are trying to determine the importance of a filter to the network as a whole. However, the techniques used are very similar. Unlike the activation based the techniques however, the gradient is back propagated form the loss rather than a class specific value. This allows these techniques to focus on the network as a whole rather than the state of the network at a single input. A good example of this is a technique introduced independently by both Molchanov *et al.* [167] and Figurnov *et al.* [168]. This technique uses the Taylor expansion to approximate the effect on the network when a filter is removed. The mean of the product of the activation map and gradient map. The absolute value of the mean value is taken, to give the the first-degree Taylor polynomial. Molchanov *et al.* [167] found that taking the absolute value is beneficial for network pruning as it takes the score for a very poor performing filter (which would be negative) and ranks it similar to the best performing filters. The benefit arises during the retraining, when removing filters that either strongly benefit or disadvantage the network have the effect of destabilising the network which makes retraining harder. Taking the absolute value ensures that filters that are the most average in terms of performance are ranked lowest.

Finally, a much simpler solution was proposed by Li *et al.* [169] which uses only the activation map as an indicator of how useful the filter is to the network. They found that taking the l_1-value of the activation map gave an indication the corresponding filter to the network.

A technique that we do not explore in this section, but is worth discussing is channel selection. This is a selection of techniques that try to find the optimal subset of channels for a given task. Channel selection techniques have found applications in several fields such as medical applications [170] , network pruning [171] , and Generative Adversarial Networks (GANs) [172] amongst others. However, a major difference between the techniques mentioned above

and channel selection, is that channel selection aims to find an optimal selection of channels for a given task. As such they often employ techniques such as greedy algorithms to discard unnecessary channels. The above methods instead are concerned with assigning a value to each filter regardless of their use to the network. Due to this reason, we do not experiment with channel selection methods.

Below are the equations for each of the methods discussed. For each, α_k^c is the weight of filter k for class c. Z is number of individual values in an activation map, A_k is the activation map for filter k, and y is the prediction score for class c. The index for the width and height of A are given as i and j respectivly.

- **Grad-CAM [15]:**

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k}. \tag{6.1}$$

- **Grad-CAM++ [23]:**

$$\alpha_k^c = \sum_i \sum_j \left[\frac{\frac{\partial^2 y^c}{(\partial A_{ij}^k)^2}}{2\frac{\partial^2 y^c}{(\partial A_{ij}^k)^2} + \sum_i \sum_j A_{ij}^k \{\frac{\partial^3 y^c}{(\partial A_{ij}^k)^3}\}} \right] . \text{ReLU}\left(\frac{\partial y^c}{\partial A_{ij}^k}\right). \tag{6.2}$$

- **XGrad-CAM [70]:**

$$\alpha_k^c = \sum_i \sum_j \left(\frac{A_{ij}^k}{\sum_i \sum_j A_{ij}^k} \frac{\partial^2 y^c}{\partial A_{ij}^k} \right). \tag{6.3}$$

- **Taylor [167, 168]:**

$$\alpha_k = \left| \frac{1}{Z} \sum_i \sum_j \frac{\partial O}{\partial A_{ij}^k} A_{ij}^k \right|. \tag{6.4}$$

- **l_1-norm [169]:**

$$\alpha_k = \left| A_{ij}^k \right|. \tag{6.5}$$

From the above techniques, we can see two common elements that are used repeatedly: the activation map and the backpropagated gradients. However, we believe that a simple method for assigning a score to the filter has been overlooked. In the above examples, put simply, the activation map is a view into how much the filters were activated by the input. The gradient map on the other hand is a view into how useful these activations are to the networks

prediction. Simply because a filter was strongly activated by a region of the input, it does not mean that the region is useful to the models final prediction.

While other techniques have combined the activation map and the gradients together to create weights. We believe that a simple but effective approach has been overlooked. In this chapter, we propose to use the *product* of the signed gradients and activation maps from the final convolution layer of a network. We call this *Grad-AMap*. Using the notation we have established above, our Grad-AMap method can be shown as:

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k} A_{ij}^k. \tag{6.6}$$

The intuition behind why this technique has the potential to perform better than other techniques for measuring a filters importance is due to the combined use of the activation map and the gradient. These both reveal different aspects of our network. An activation map that shows a high activation means that some feature in the input image has triggered that particular filter to activate. Through the use of the gradient, we gain an insight into which filters *are* useful to the network. If a negative gradient is given to a filter then it suggests that the model found that filter unhelpful in its final prediction. By taking the product of the two we assign a low score to filters that activated highly, but were given a negative gradient. For example, an activation map with a strongly positive score combined with a strongly negative gradient will give a strongly negative score. We note the similarity of Grad-AMap to the Taylor series method. However, there are two crucial differences. The first is that to ensure the Taylor series method is taking the first-degree Taylor polynomial, the absolute value has to be taken for it to be valid. The second is that Grad-AMap uses gradients backpropagated from a class specific softmax, while the Taylor method backpropagates from the overall loss.

6.2.2 Evaluation of Filter Ranking Methods

To measure the ability of each of the discussed methods, we propose an experiment based on the local accuracy deletion measure by Petsiuk *et al.* [9] that we used extensively in the previous chapters. Rather than assigning a score to pixels in an input image, we assign a score to each filter in a specific convolutional layer. In the local accuracy measure by Petsiuk *et al.* , these

pixels are iteratively removed (pixels are set to 0) and the accuracy or softmax is recorded and averaged over all validation images in a dataset. Typically, regions are removed from most important to least. As the regions are removed, the accuracy or prediction score for the target class drops accordingly. We follow this pattern for our experiment, recording the accuracy of the network (which we normalise to be between 0% – 100%) as we remove the most important filters first. We do this for every image in the ImageNet validation set. The better a technique is at producing accurate filter scores, the fewer filters we should be able to remove before the model can no longer accurately predict the correct class and the overall model accuracy drops.

In our experiment, we therefore rank the filters in the selected convolution layer and incrementally zero out the channels of the activation map produced by a networks final convolution layer based on the filter rankings. Zeroing out a single channel of an activation map is the equivalent of pruning the corresponding convolution filter. We increment in steps of 16 filters, for a total of 32 increments for a layer consisting of 512 filters. To gain a single value for each method we measure the Area under the Curve (AUC) value as per the metric used in RISE [9]. An AUC score approaching 0 suggests that a method is able to better rank the importance of filters, as we are able to rapidly reduce the accuracy of the network. For this experiment, we use VGG16 and ResNet18 models to test our method. We use ResNet18 instead of ResNet50 for this experiment for efficiency reasons. ResNet18 contains only 512 filters in the final convolution layer as opposed to the 2,048 found in ResNet50s. In addition to experiment with Grad-AMap and the other techniques discussed, we propose the use of two additional methods. The first is to use a complete random set of filter rankings as a baseline. Any method that suggests it can rank filters, should outperfrom this baseline. The second is to use the Taylor series method, but with gradients obtained by backpropagating from a class specific score, rather than the overall loss.

The results for filter removal versus model accuracy can be seen in Figure 6.1 and Table 6.1. These results confirm that Grad-AMap, our proposed method of measuring filter importance, performs the best compared to other common methods. In particular, we note that Grad-AMap performs better than the CAM methods for both the ResNet18 and VGG16 networks. Indeed, it seems that the CAM methods that do use the product of both the activation maps

TABLE 6.1: Filter removal results. A lower AUC score suggests we can identify and remove the most important filters first, resulting in the network accuracy to drop.

	VGG16	ResNet18
Random	0.685	0.733
Grad-CAM++ [23]	0.147	0.230
Grad-CAM [15]	0.079	0.225
XGrad-CAM [70]	0.119	0.225
l_1-norm [169]	0.057	0.618
Taylor [167, 168]	0.132	0.629
Taylor (class)	0.052	0.392
Grad-AMap (Ours)	**0.042**	**0.146**

and gradients perform worse that the gradients only Grad-CAM. The Taylor series method performs very poorly for both networks, and only narrowly beats the random baseline for ResNet18. The alternate version using the class specific gradients (Taylor class) performs much better, indicating how important this form of the gradients is to creating an input specific filter scoring method that performs well. Taylor class is still hampered by the use of the absolute values, while this does not seem to affect the scoring of the filters much when VGG16, it seems to deeply disrupt the ability to score the filters well for ResNet18. The l_1-norm is unique amongst all of the methods for only using the activation map to inform the filter scoring. While this seems to perform admirably for VGG16, the non-linear nature of the ResNet18 seem to deeply inhibit this method to perform the filter scoring accurately.

An interesting property, displayed in the ResNet18 results, is the small increase in accuracy as the filters are removed. This is likely happening because we are pruning a single convolution layers activation, which are then added back to the activations stored from the previous block. The effect of taking the absolute value in the Taylor method results in both the best and worst filters being removed early, which accounts for the earlier bump in accuracy.

6.2.2.1 Aside: Better CAM Explanations?

A question that could be plausibly asked is, if Grad-AMap is so effective at scoring filters, why can it not be used to inform the weights when creating a

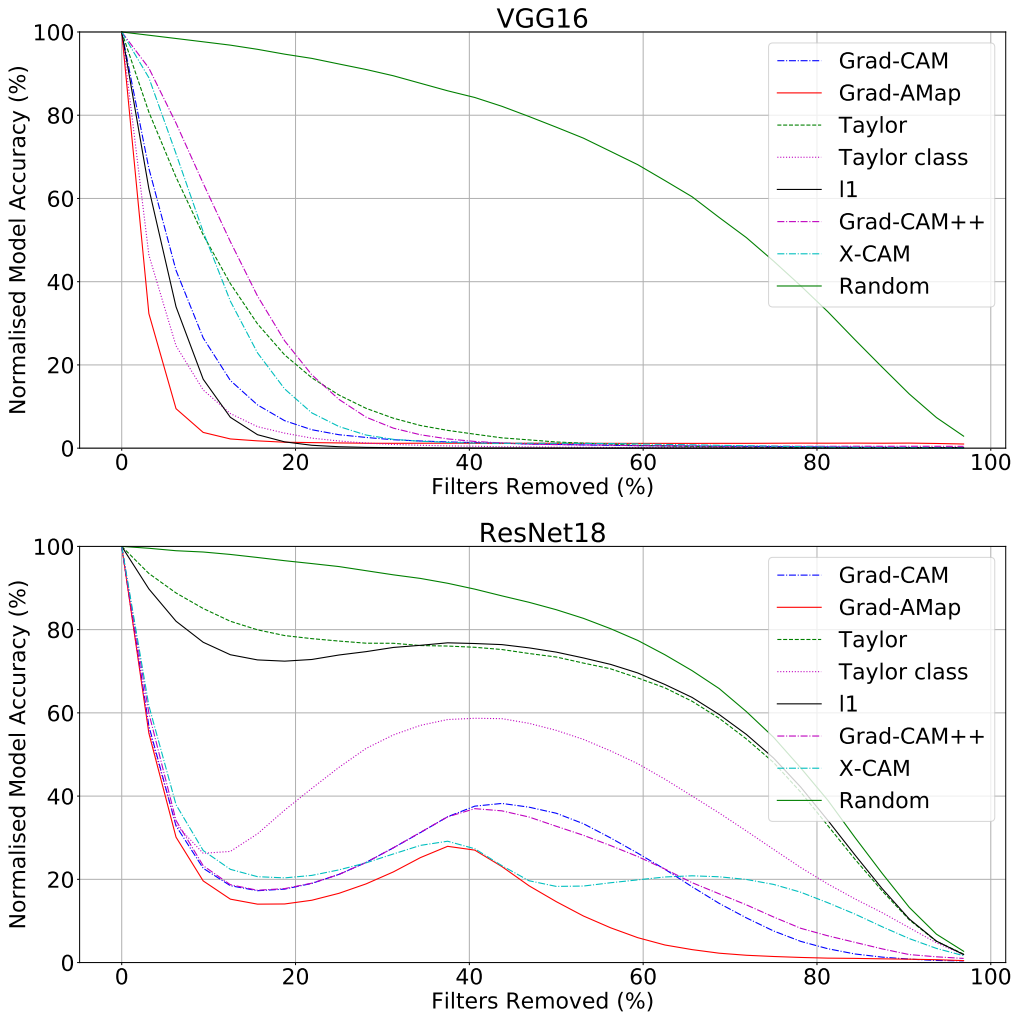

FIGURE 6.1: Showing the percentage of filters than can be removed vs the normalised model accuracy. Our proposed method is able to quickly decrease the accuracy be removing important filters.

CAM visualisation and produce better explanations? Indeed, this is an avenue that we explored and is worth discussing as a brief aside.

The primary difference between creating a visual explanation with a CAM technique, and scoring filters, is that typically the very final activation maps are used prior to the classification layer/layers. This means that rather than taking the activations from a specific convolution layer, we take the activations after they have passed through all stages of the network. In older architectures, such as VGG16, this works perfectly well. However, in more modern architectures the use of batch normalisation [173] has become standard. In a network such

TABLE 6.2: Showing the local accuracy results for various CAM explanation methods including Grad-AMap. A lower score is better. Observe how Grad-CAM++ and XGrad-CAM (which combine the gradient and activation) do not score consistently well for networks with batch normalisation.

	VGG16	VGG16-BN	ResNet50
Grad-CAM++ [23]	0.111	0.121	0.147
Grad-CAM [15]	0.105	**0.107**	**0.142**
XGrad-CAM [70]	0.105	0.108	**0.142**
Grad-AMap-CAM (Ours)	**0.101**	0.108	0.144

as ResNet, a batch normalisation layer follows the convolution prior to the combining with the identity activations. This is problematic for our Grad-AMap as it relies on the sign of the activation values. If a value is positive, it is beneficial to the networks prediction, negative if not. When the batch normalisation process is introduced, any activation maps that pass through it are shifted to being zero centred. This results in the possibility of what were once positive values being shifted to have a negative value.

To demonstrate this we use the local accuracy deletion metric to evaluate various CAM methods against Grad-AMap. Here, we take the final set of activations from each network to visualise and refer to our method as Grad-AMap-CAM. We test using both VGG16 and ResNet50, and to ensure that it is batch normalisation causing the problem as we hypothesised, we also test VGG16 with batch normalisation (VGG16-BN). The Results can be found in Table 6.2. From the results we can see that our method performs well with VGG16, but does not perform as well when using ResNet50 or VGG16-BN. Also interesting to note is that both Grad-CAM++ and XGrad-CAM, which also make use of the product of the activations and gradients, show the same traits. The variation in scores seems to be due to the weighting given to the activation map when creating the weights. For example, with XGrad-CAM the activation maps are first divided by the sum of all the activation maps prior to computing the dot product with the gradient. This has the effect of giving more prominence to the gradients, which in turn makes it more similar to Grad-CAM. This is likely why it performs worse than ours for VGG-16 but better for ResNet50.

6.2.3 Building Filter Score Distributions

With a method developed for accurately scoring the effect of a filter on a model's prediction, we are able to turn our attention to how these scores can be combined to allow a better understanding of a network. We do this by building a distribution of the filter scores for each class/filter pair. For example, for a 10 class dataset with 512 filters in the final convolution layer, we would gather 5120 distributions. The filter scores are obtained by passing the entirety of the training dataset through the network and calculating the Grad-AMap score for each image. Prior to input into the network the images are resized so that the smallest edge is 224 pixels in length. A centre crop is then taken at size 224×224. An example of the distributions created using ImageNet with VGG16 are shown in Figure 6.2.

From the distributions, we observe that they are typically skewed unimodal suggesting that when a filter activates for a given class, it tends to do so in a similar way for the remainder of the images in that class. From these distributions we can discover the mean activation for every filter in each class: μ_{cn}, where c is the class and n is the filter index. Knowing the mean value of the distribution is important, as we have seen that the distributions are unimodal, so activations should all fall close to the mean value. If an input were to cause the filter to score in a way that could be considered an outlier then this may be reason to the flag the input for further investigation. For example, we would expect a filter that has learnt to activate on dog faces to activate highly when a dog is present and weakly if not. If an image from the dog class does not subsequently activate the 'dog face' filter then this could be seen as something interesting to follow up on.

6.2.4 Definition of Surprise and Expectation

By building the distributions for each filter/class pair, we are able to begin to building concepts that will be important to help discuss how the network reacts to a given input image. A method such as the one introduced by Kim *et al.* [20] seeks to find images whose features are not captured by the network. This is a useful approach and the rationale is that by understanding which images are not modelled well allows an insight into how the model has learnt how to represent the classes of the dataset. However, in work by Kim *et al.* [20],

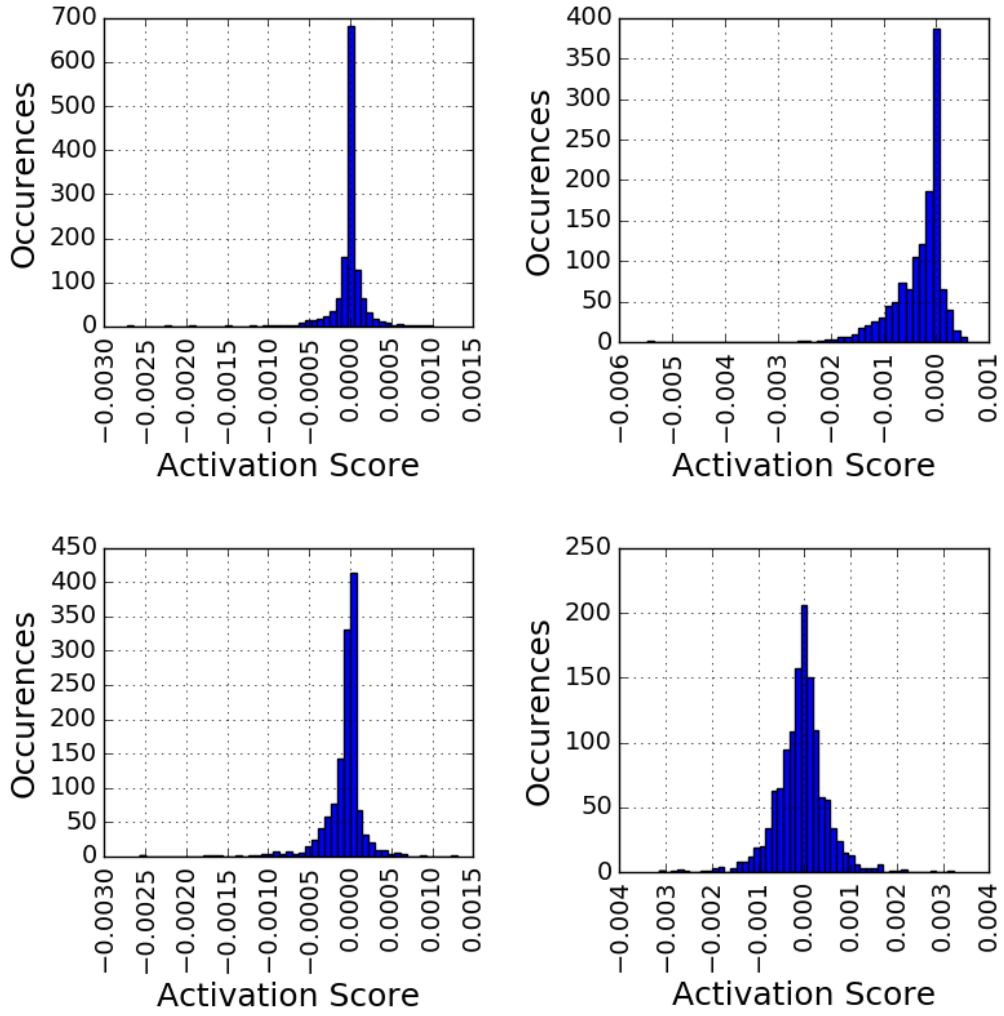

FIGURE 6.2: Examples of four typical distributions of filter importance scores made using Grad-AMap with ImageNet and VGG16. Each distribution shown here represents a class filter pair (i.e. class 30, filter 100).

the images that are not captured well are given the binary label of criticism. With such a binary approach, there is little understanding as to which regions of the images are problematic to the network other than a visual assessment. Having only a visual assessment as the means to understanding why an image may be not be captured is problematic, as it introduces a users bias to how they think the model might have capture the information of a certain class.

In this section, we begin to show how we can use our technique to anchor the explanations of why models fail to individual filters within the final convolution

layer of the network. We propose that an image fails to be successfully predicted as a certain class for two primary reasons. The first is that it is lacking some feature that the network has learnt to represent that class with, the other is that some feature present in another class is overpowering the other features in the image and confusing the network. We call the first concept *expectation*, that is the network was expecting to see some feature in the image that was missing. The second concept we call *surprise* which suggests that the network is surprised to see a certain filter score so highly. We use the expressions surprise and expectation because unlike in previous chapters, we assume that we have access to the ground truth for any input. This allows us to explore how the network reacts to what would have been the correct class, and compare to how it learnt to represent it from the training data. More concretely, we define these two concepts as occurring when there is a filter that is given a Grad-AMap score that is clearly an outlier compared to the distribution built on the training data. As our distributions are not normal, we define surprise and expectation as happening when filters score outside of a certain percentile. More specifically, we define the filter as surprising if $\alpha_n > P_{cn}(99.9)$ and the filter expecting if $\alpha_n < P_{cn}(0.1)$. Where $P_{cn}(x)$ is the x^{th} percentile, c is the ground truth class and n is the filter index. These percentile values were chosen through experimentation and observation of the results. In the future it would be beneficial to take a more rigorous approach to uncover how this effects the results. Values that fall within these percentiles are considered to be within the expected values for that class and filter combination. These two percentiles are noticeably very close to the limit of what the model is expecting to see, and when compared to the training data, it is a very rare occurrence to see any filter activate past these percentiles. However, we did find that often in practice filters would be flagged as an expectation filter when they had barely registered any Grad-AMap score. It transpired that this was due to the a filter scoring very lowly over the training data, but then scoring even lower for an unseen image. To ensure that we are only working with filters that have some bearing on the prediction, we calculate the mean value μ_{cn} for each of the filters n for a given class c. As a side point, we chose the mean for expediency, it is possible that other methods such as using the median or variance could produce useful results too. We, then, check that the mean of the filters for our

unseen image is greater than than the mean of the all of the filters for that class:

$$\frac{1}{N}\sum_{n=1}^{N}\mu_{cn}. \tag{6.7}$$

This prevents filters with a very low μ_{cn} from being labelled as expected if an even lower α_n is encountered.

6.2.5 Deviation from Mean Filter Activation – β

While surprise and expectation will be useful for understanding how the individual filters react to an unseen image, we would also like to develop an understanding of how an unseen image as a whole is represented by the network. It stands to reason that if an input image is modelled well by the learnt distribution, we should expect our unseen α_n to fall somewhere close to μ_{cn}. Once the Grad-AMap filter scores (α_n) have been created for an unseen image we can asses them using their deviation from the training mean ($\alpha_n - \mu_{cn}$). The more the value deviates from 0, the more the filter is under- or over-performing.

We also propose doing this over the entire set of filters for a given image to compute a value, β, that measures how a filter's importance scores, α_n, for an unseen image, deviate from the mean μ_{cn} of the target class training data. For every unseen image in the ImageNet validation set, we subtract the mean of the training data from their importance scores and take the mean of those values:

$$\beta = \frac{1}{N}\sum_{n=1}^{N}(\alpha_n - \mu_{cn}). \tag{6.8}$$

Large values of β signify that an image is a strongly typical representative of the training set while small β values suggest that the image is strongly atypical of the training set.

6.3 Exploration of Failure

At this point, we now have two techniques developed that allow us to explore possible reasons for failure. The surprise and expectation measure is suitable for understanding the behaviour of individual filters, while the β score is applicable to understanding how the entire image is represented by the network. In this

section, we explore possible reasons for failure using these techniques. We begin with a look at the overall dataset using the β before moving on to look at individual examples of failure using our surprise and expectation metrics. We explore the ImageNet dataset using pretrained models from the PyTorch model zoo [132]. Specifically, we use the VGG16 and ResNet50 architectures. We use the ResNet50 architecture for this section as it is a more commonly used network than ResNet18, and allows our surprise and expection technique to be tested more fully due to the 2,048 filters present in the final convolution layer.

6.3.1 Understanding the Reasons for Failure

Throughout this chapter we have been asking what are the reasons for an images class to fail to be predicted successfully. Is it due to a lack of features the network is expecting to see or is there some overpowering feature that is causing confusion. Using the β score we can now begin to seek answers to these questions.

We begin by passing all of the 50,000 images from the ImageNet validation set through the network and storing the β scores for each image along with the corresponding predicted class, pre-softmax prediction score, and the softmax score. When these values are plotted we observe some very striking patterns. In Figure 6.3, we show the β scores plotted against the pre-softmax scores for both VGG16 and ResNet50. The first thing to note is just how linear the relationship between the β score and pre-softmax prediction score is. It seems that having an unseen image with a low β score makes it much more likely that the network will struggle to give a strong prediction score. This relationship holds across both VGG16 and ResNet50 suggesting that it is not a phenomena limited to a singe architecture type. To ensure that this relationship is also not purely limited to ImageNet, we show alternative plots for models trained on the CUB200 and Food-101 datasets in Figure 6.4. As we see, this relationship remains consistent. While the pre-softmax score is often indicative of an image being classified correctly, the network is trained using a softmax layer. This has the effect of normalising the scores into a probability distribution which has the effect of reducing all values to fit into [0, 1]. In Figure 6.5, we show the plots for β scores versus the softmax scores. Here, we see that a relationship still exists

just not as pronounced as with the pre-softmax scores as they have now been normalised to fit into the new range of values. A problem with plotting the all 50,000 β values in this way is that it is difficult to discern the relationships between images in a single class. For example, there is significant overlap between correctly and incorrectly predicted values around the $\beta = 0$ region. It could be that images from the same class are actually forming clusters rather than displaying the same linear relationship as the entire dataset does. We investigate this by plotting only images from within a selection of randomly chosen class, these are shown in Figure 6.6. The intention of this figure is to allow ease of viewing of the β scores. By disentangling the classes from the dataset as a whole we can observe how the relationships play out in images related to each other. We again see a strikingly linear relationship between the β scores and the pre-softmax score. With the softmax scores we see that that the majority of images are given a high prediction score which then reaches a β value before the score begin to decrease and they are no longer able to be classified correctly.

While we can clearly see visible trends for the classes that we have visualised, it is desirable to obtain some idea of how these extend to the dataset as a whole. To achieve this we perform Spearman rank correlation between all the β values within a class, and the prediction scores both pre- and post-softmax for each image. For VGG16, the mean Spearman score over all the classes between β values and the pre-softmax predictions is 0.998 while the score for post-softmax predictions is 0.782. For ResNet50, the mean Spearman score over all the classes between β values and the pre-softmax predictions is 0.999 while the score for post-softmax predictions is 0.801. These results clearly indicate that β is a reliable measure of how well an unseen image is represented by the network.

So far, we have been focusing on the relationship between β and the networks predictions scores. However, we set out to try and understand and explain failure. It is, therefore, worthwhile to look at the relationship between whether an image was classified correctly or incorrectly and its β value. For the images classified incorrectly, we found that for VGG16 86.60% of them had a negative β value, while only 45.88% of correct classifications have a negative value. For ResNet50 we found these values to be 90.20% and 47.67% respectively. This suggests that the majority of images fail to be classified

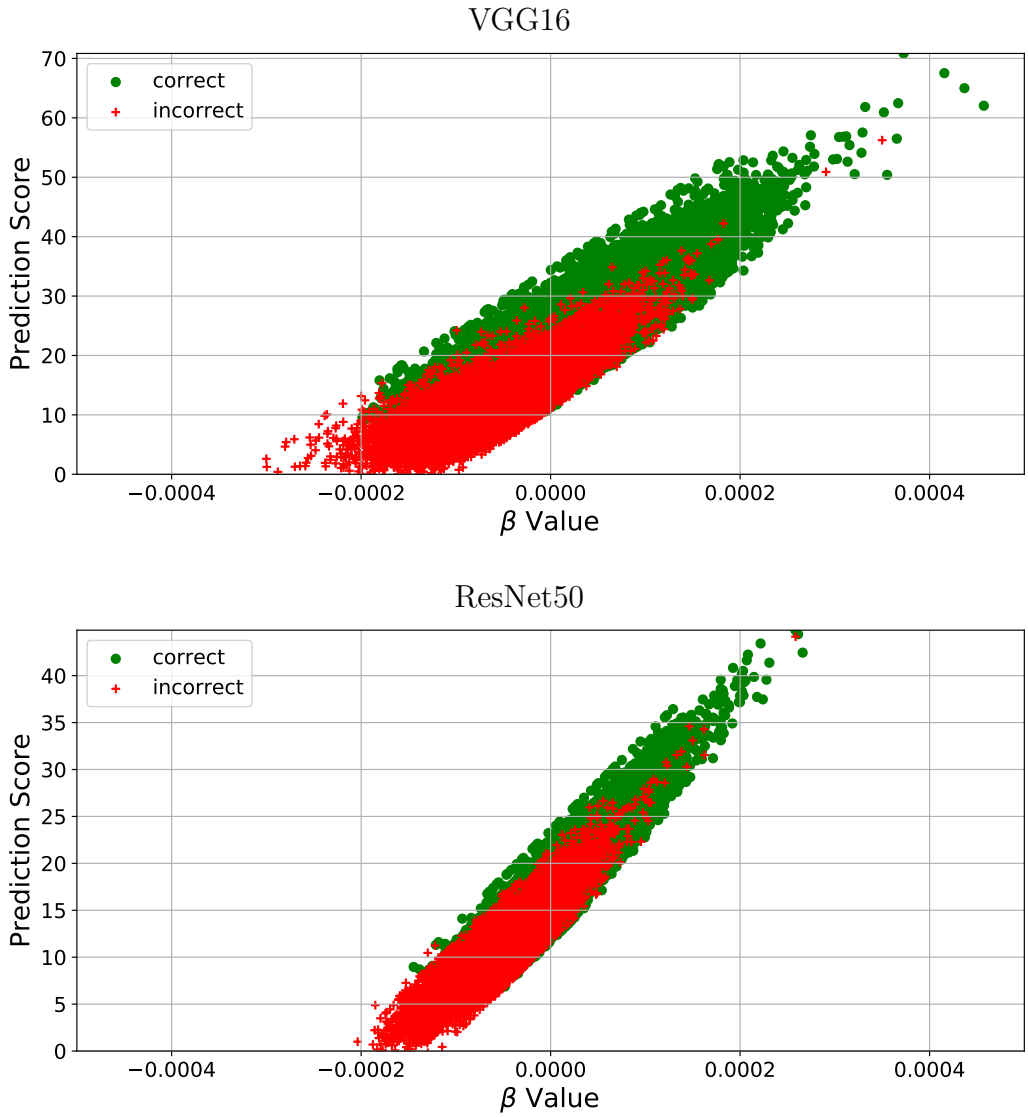

FIGURE 6.3: All β values for the validation images within ImageNet using VGG16 (top) and ResNet50 (bottom).

correctly because they lack the features the network is expecting, rather than that something in the image is surprising the network and overpowering other features.

To try and understand how these unseen images are being represented by the network, we take images from a handful of classes and display them according to their assigned β value. For the 50 images present in a selected class from the ImageNet validation set, we show (from left to right) images

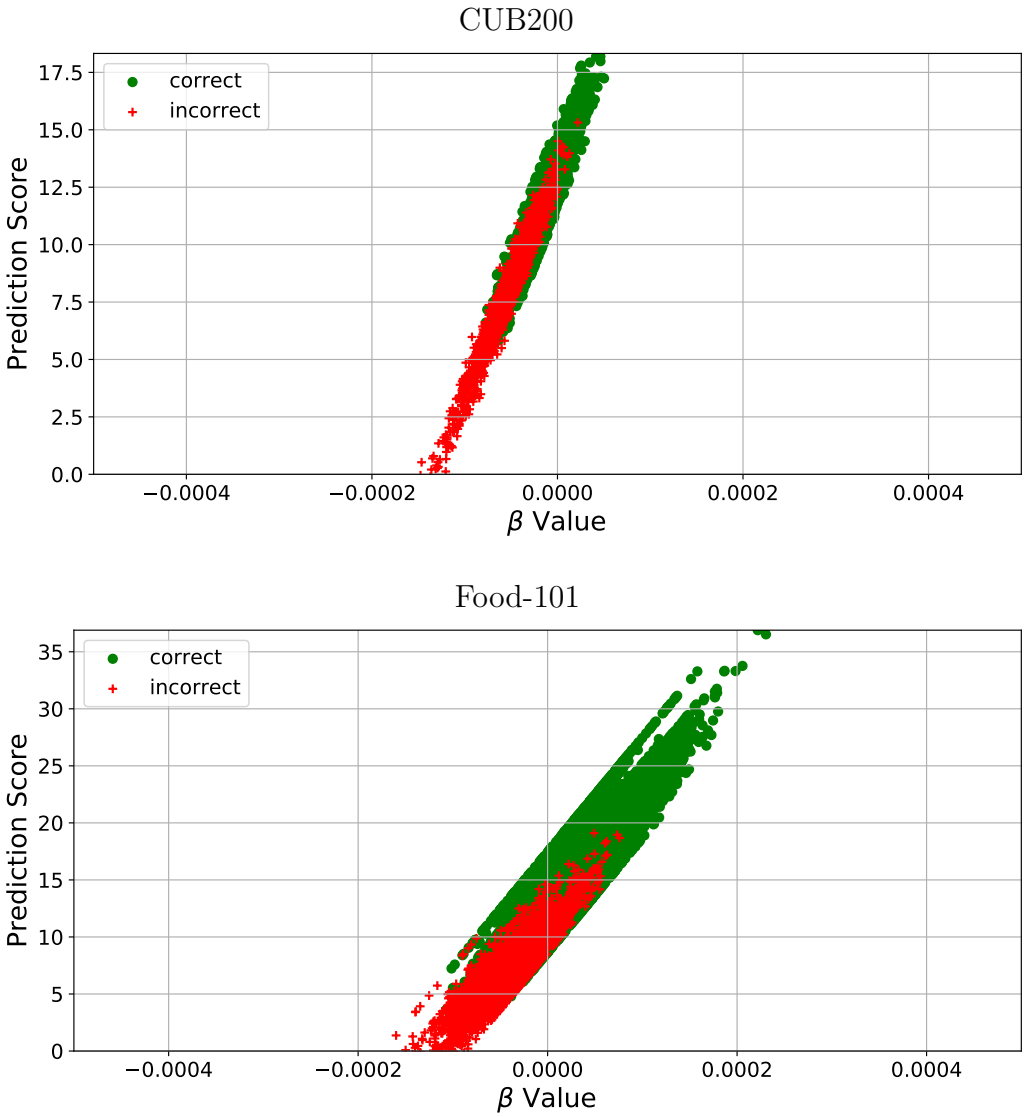

FIGURE 6.4: All β values for the validation images from CUB200 (top) and Food-101 (bottom) using the ResNet50 architecture.

ranked as so: [1,13,25,37,50]. Here, 1 is the image with the lowest β score in the class, and 50 is the highest. The images can be found in Figure 6.7. From these we can see that images with a higher β score look like they could almost be the perfect representation of the class. For example, the final images across all the classes show all of the object, at a sensible scale and are typical of what you would expect for the class. At the other end of the range of images are those that have a very low β score. These seem to be atypical of the class

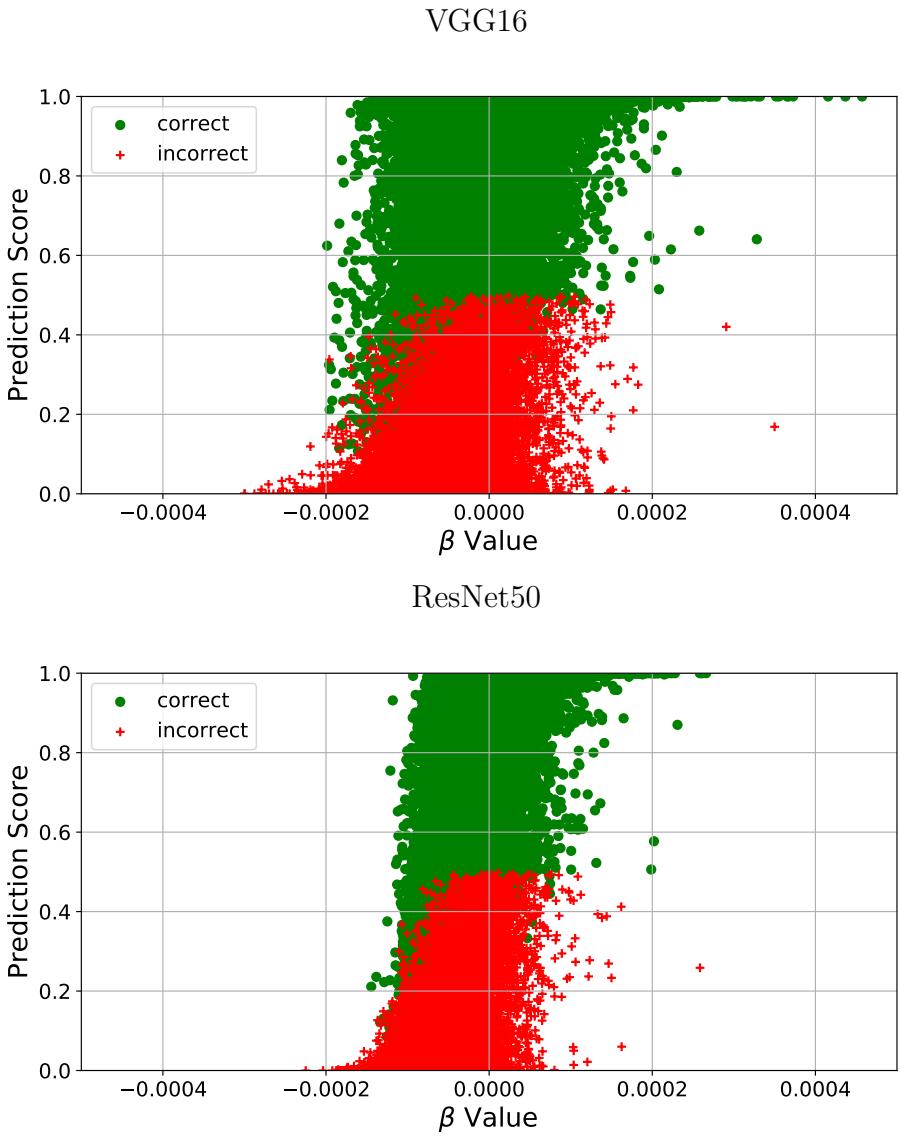

FIGURE 6.5: All β values for the validation images within ImageNet using VGG16 (top) and ResNet50 (bottom) showing the relationship between the β values and the softmax scores. Note the β values are identical to those in Figure 6.3. The β values are still able to differentiate between good and bad, however the softmax has the effect of moving most of these values close to either 0 or 1.

at best or often not contain the object class at all. For example the second image in the pizza class. At this stage we make no differentiation between images that have been classified correctly and those that have not. However, in the following sections we separate misclassified images into two groups, those with a very high β value and those with a very low β. By separating the

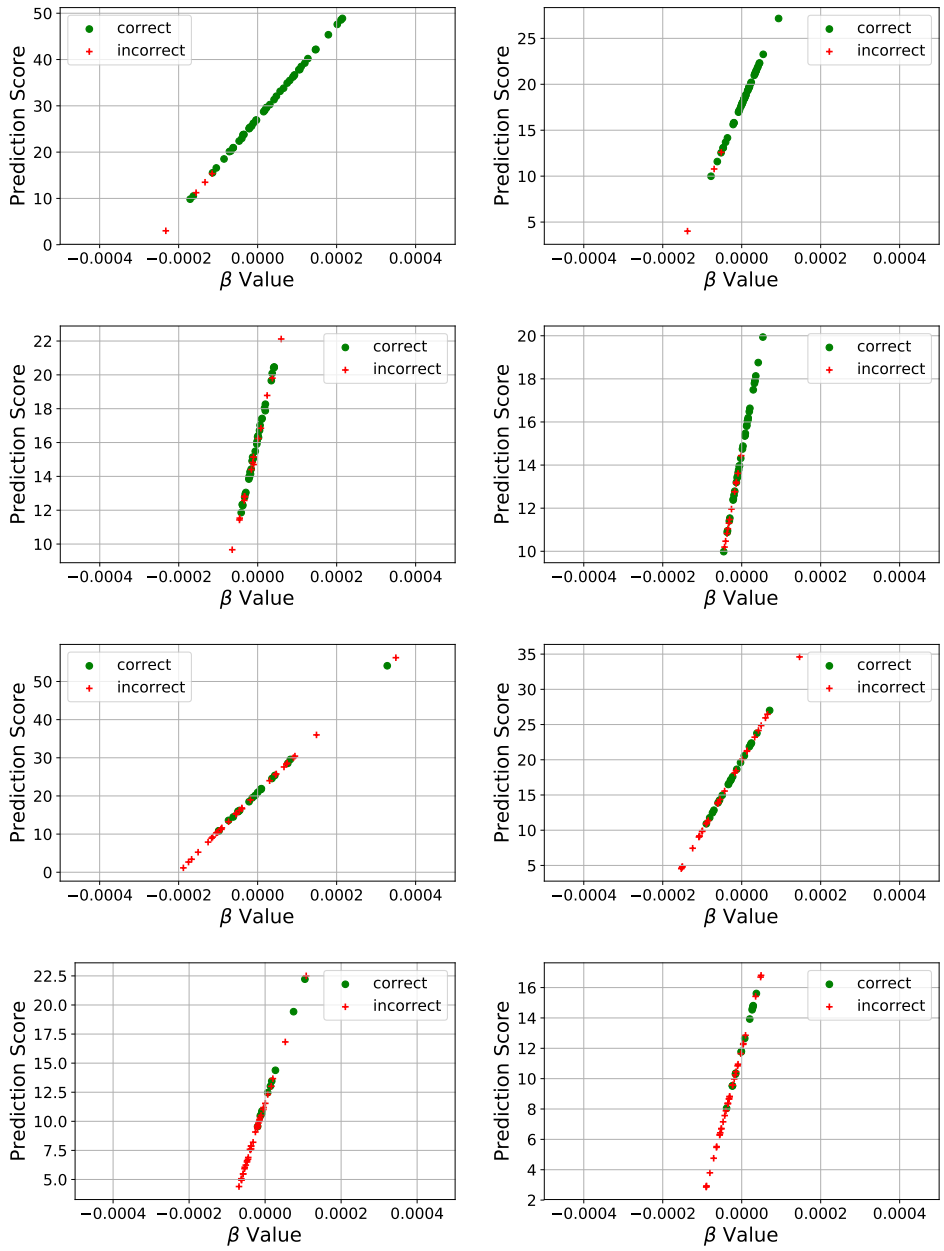

FIGURE 6.6: Class specific β values using VGG16 (left column) and ResNet50 (right column).

misclassified images into these two groups, we are able to explore the more obvious reasons for failure. Looking at misclassified images with a mid-range β value may be inconclusive, or at worst, misleading as we use our own biases to infer what we think is wrong with the image.

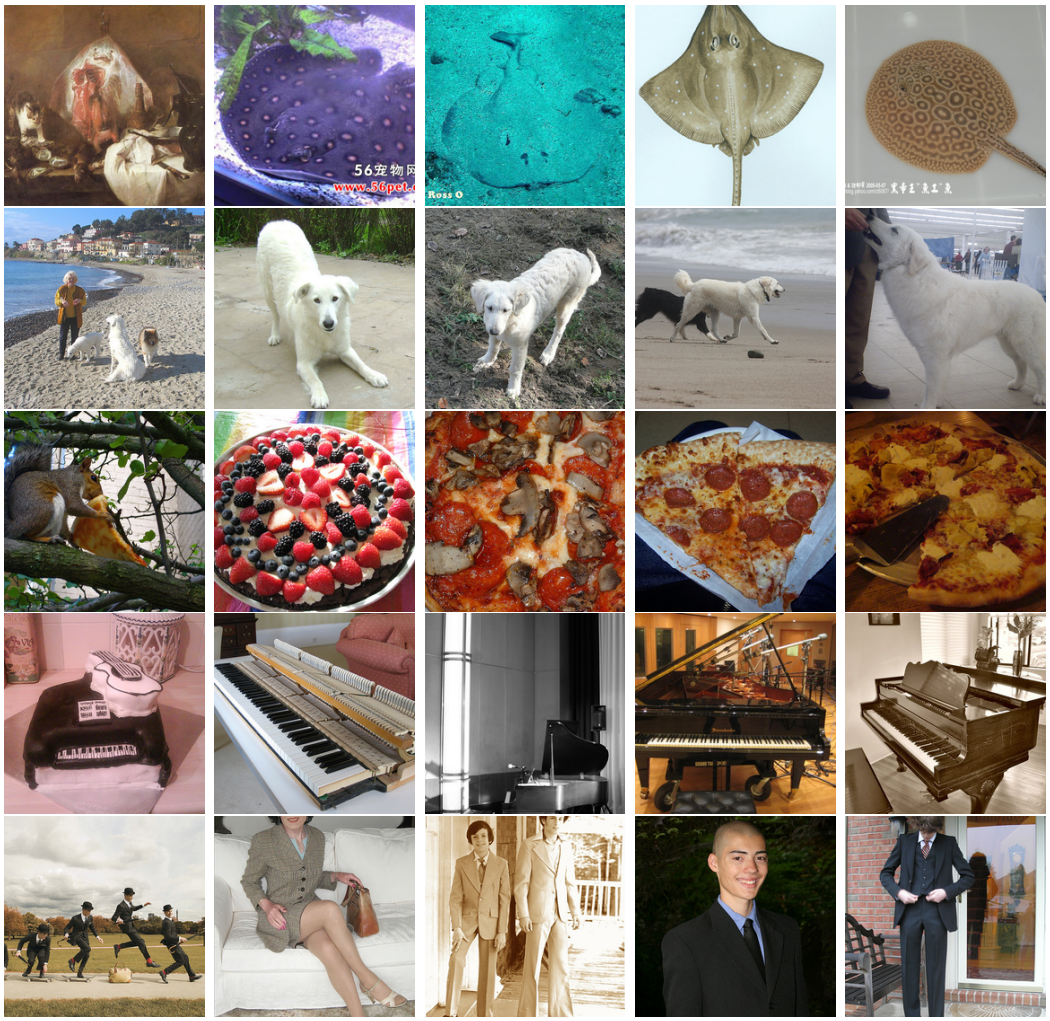

FIGURE 6.7: Each row consists of images from the same class. The images with the lowest β are on the left, moving up to the highest on the right. These image are taken from ImageNet using ResNet50.

6.3.1.1 Misclassification with High β Values

We begin by looking at images that were misclassified but still obtained a high β value. This class of images is somewhat of an anomaly and, by looking at the plots in Figure 6.3 and Figure 6.5, there seems to be much more uncommon than misclassified images with a low β value. We suggest that this is anomalous because an image having a high β value suggests that every filter activated in the way it was supposed to for that class. If this is the case, how can an image activate the filters in an expected way, but still be misclassified?

We hypothesise that what is happening is that two classes share very similar features, if an image accidentally posses slightly more of some feature, it can push it over into the other classes expected activations. We visualise misclassified unseen images with high β scores for both VGG16 and ResNet50 in Figure 6.8. For each image, we show the predicated label and the ground truth label. From viewing these, it becomes clear that our hypothesis seems to be correct. All of the images containing very high β scores seem to be both typical of both the ground truth class and the predicted class. This could either be because they are visually similar, as in the case of the horned viper / horner rattlesnake or the terrapin / mud turtle, or they are typically co-located 'in the wild'. Examples of co-located objects are the the drum / drumstick and the mortarboard / academic gown. Clearly these objects are commonly located with one another, so without some additional help at training, it may be difficult for the network to learn to differentiate between the two classes.

6.3.1.2 Misclassification with Low β Values

While misclassified images with a high β value appear to be typical of both the predicted class and the ground truth class, is the opposite true for misclassified images with low β values? We have seen previously in Figure 6.7 that images with low β values seem to be atypical of the class, however we previously looked at all images from the dataset. In this section, we specifically discuss unseen misclassified images with a low β value. As before, we found the best way to begin was to visualise a selection of the relevant images from the unseen validation set. In Figure 6.9, we show both misclassified images with very low beta values β alongside an image from the same class that is classified correctly with a high β value. Presenting the images this way allows us to begin to understand the reasons that these images may have failed.

From Figure 6.9 we see some clear trends. The first is that some misclassified images are simply not typical of the medium that the remainder of the dataset represents the images in. The examples of this are the giant panda, ticks, and oxcart images. While these are all the images of the class they purport to be, they are visually distinctly different. Both the tick and oxcart are drawn representations, while the panda seems to be an origami version. While a human may be able to classify these image correctly, there are obviously features that are missing that the network requires to classify these correctly.

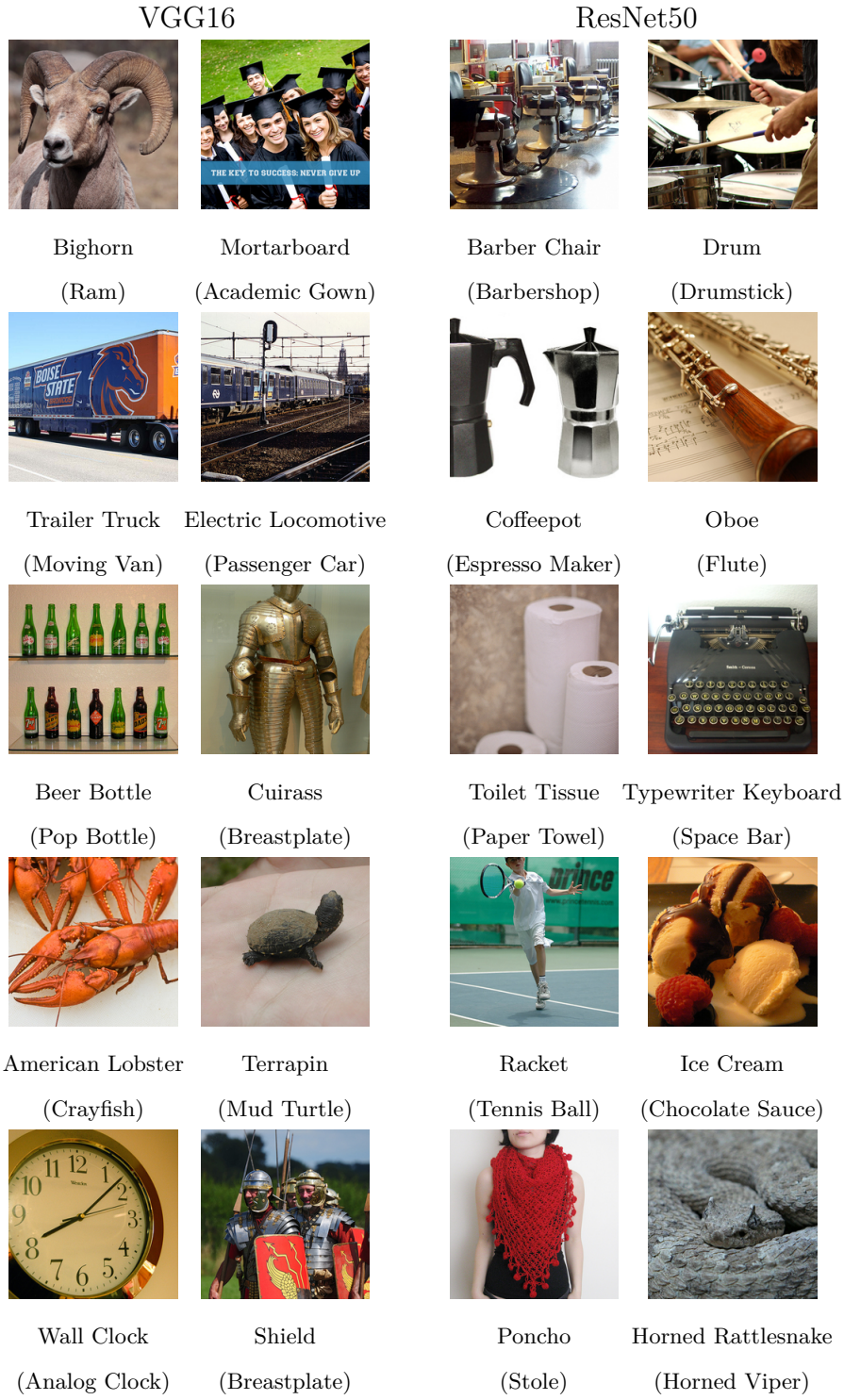

FIGURE 6.8: Examples of unseen, misclassified images with high β values. The caption under each image is the predicted class with the corresponding ground truth in parentheses.

A second mode of failure seems to be the scale of the image. This is either where the object is to small in the image, or too large. Examples of this is found in the jay, peacock, and electric locomotive images, where the misclassified images appear to be much smaller in scale than those that were classified correctly with a high β value. In these examples, it seems that although the object does clearly exist within the image, the features are not present in the correct scale for the network to classify them correctly.

A final mode of failure that we detected seems to be where the object itself is only an ancillary part of some other object. Examples of this are seen in the mortarboard and bottlecap images. In both of these the target object is not the focus of the main image. In the mortarboard image, it would be hard to argue that the focus of the image was not the dog and the bottlecap image is primarily a model plane with the bottlecap being a component of that.

6.3.2 Visualising Surprise

So far in our analysis of why images fail, we have only looked at the macro reasons exposed through the use of the β value. Through this we discussed the high-level reasons for failure such as images being visually similar to another class, or being the wrong scale or visually distinct. So far, we have not distinguished our technique from those such as Kim *et al.* [20] who are able to find training images that are typical and atypical of the training data. Because we have tied the β value to individual filters, we are able to work back and identify the effect of individual filters through their surprise and expectation values. This is distinct to the method from Kim *et al.* who use the values extracted from the final classification layer, meaning there is no direct route back to individual filters for further analysis. In this section, we use our proposed surprise measurement to visualise images that appear to have some surprising feature.

The first hurdle that needs to be overcome is how to visualise the filters that have been flagged as surprised. Simply visualising the individual filter as we may have done with Grad-CAM would have been too coarse to accurately identify specific features that activate a certain filter. However, through the use of SWAG we are able to identify much more detailed regions of the image due to the use of superpixels. To enable us to view the effects of a single filter, we use SWAG with a number of variations.

FIGURE 6.9: Misclassified images with low β values on the left of each column alongside an image from the same class that both classifies correctly and has a high β value, suggesting it displays features the network desires. Captions are the ground truth.

First we use regular SLIC superpixels rather that any of the proposed variants. This is because we want to maximise the alignment with regions in the image space so as to better draw comparisons with other images. Second, we only visualise 50 superpixels rather than the 300 used previously. This is because we want to visualise a single contiguous region, rather a small region that could be part of a larger feature. Third, we use vanilla gradients rather than guided-backpropagation to weight the superpixels.

We found vanilla gradients to perform better in this context. Possible reasons for vanilla gradients better performance is discussed in Section 6.4.1. The final, and perhaps most important variation, is that to ensure we only visualise the regions important to a specific filter, we zero out all gradients produced by filters in the same layer as the surprised one. This has the effect of only backpropagating the gradients back to the input pixels that are relevant to the surprised filter. Our original SWAG method produced a heatmap however, to improve clarity for this particular task, we instead draw a border around the superpixel that scores the highest.

Another consideration is that once we have outlined a region of the image that we suspect of being surprising to the network, how do we put this into the context of how the network is representing that feature? To remedy this, alongside the image showing the region that represents the surprising feature, we place images from the *entire training set* that most activated the filter. Doing this allows us to view both the feature in the misclassified unseen image, alongside the training data that most represents the filter. To find the training images that most activate the filter we pass all the training images through the network and observe the mean value of each activation map. The images that have the highest mean activations for a specific filter are stored. To remove regions of the stored image that are not important to the network, we multiple the resized activation map for the specific filter with the image and display this.

We place examples of these images in Figure 6.10. Here, we display a misclassified image with the region that activates the surprising feature outlined in yellow. Alongside each misclassified image are the three images from the training set that most activated the surprising filter. These images are selected based on their Grad-AMap score, the three shown being the highest scoring for that class and filter combination. The masked images are created

by passing the image through the network and visualising the product of the selected filters activation map and the input image. Viewing these images like this allows us to begin to draw conclusions as to why the image may be being misclassified. For example, the top image is an image from the goldfish class misclassified as being from the roundworm class. It seems that the bottom fish highly activates a surprising filter. When we view the training images that activate the filter, we see that they are wheels. This high activation of the 'wheel' filter is presumably what causes the image to be misclassified as a roundworm. Indeed, we show in Section 6.4.1 that this image could be made to classify correctly by removing the bottom fish. For other images, such as the velvet misclassified as swimming trunks, it is easy to see how a filter that activates highly for jigsaw may have fired, but not clear why it contributes to the swimming trunks class.

6.3.3 Visualising Expectation

Visualising the filters that did not activate highly is problematic due to a lack of features in the image that activate the specific filters. By identifying misclassified images that have an expected filter we are able to find images from the training set's target class that maximally activate the expected filter. Displaying these next to the misclassified image gives us an indication as to why the image may be misclassified. The results can be seen in Figure 6.11. We find that these images take two forms. The first is where the misclassified image simply does not seem to contain the correct features. For example, with the 'gila monster', the expected filter is highly activated by white objects around the gila monster rather than the animal itself. This potentiality indicates a bias in the training data that the model has learnt to represent. A second example of this is the image of the partridge surrounded by barren branches. In contrast, the regions of the partridge training images that maximally activate the expected filter seem to be leaves or foliage. Again, this indicates a possible underlying bias within the partridge class. The second type of misclassified image seems to be where the target object is present, but small within the image and the expected filter does not activate highly due to the scale of the features. The image of the violin is a good example of this, although the violin and bow are present in the image, they are small. The filter that expected to

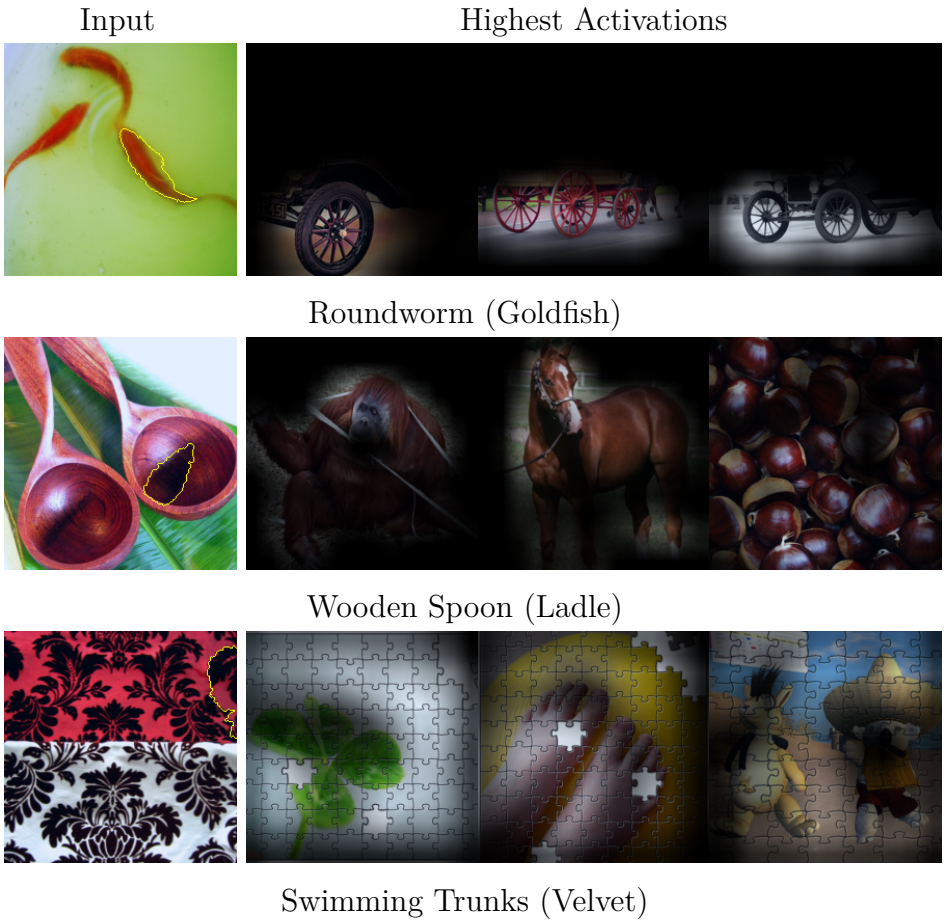

FIGURE 6.10: A misclassified image shown alongside the images from the training data that activates the surprising filter, masked to show the regions that activate the filter. The yellow boundary shows the region important to the filter. The caption is the predicted class with the ground truth in parentheses.

be activated more is one that seems to activate highly on the bow region of the training data. The peacock image is similar to the violin, where the target object is present, but the expected filter seem to activate highly on a peacock’s tail, as well as the crest on its head. Although these features are present in the misclassified image, they are not there in the scale required.

6.4 ‘Fixing’ Incorrect Classifications

To this point in our analysis we have only focused on what visual observation of the dataset can tell us. However, as we have mentioned throughout the

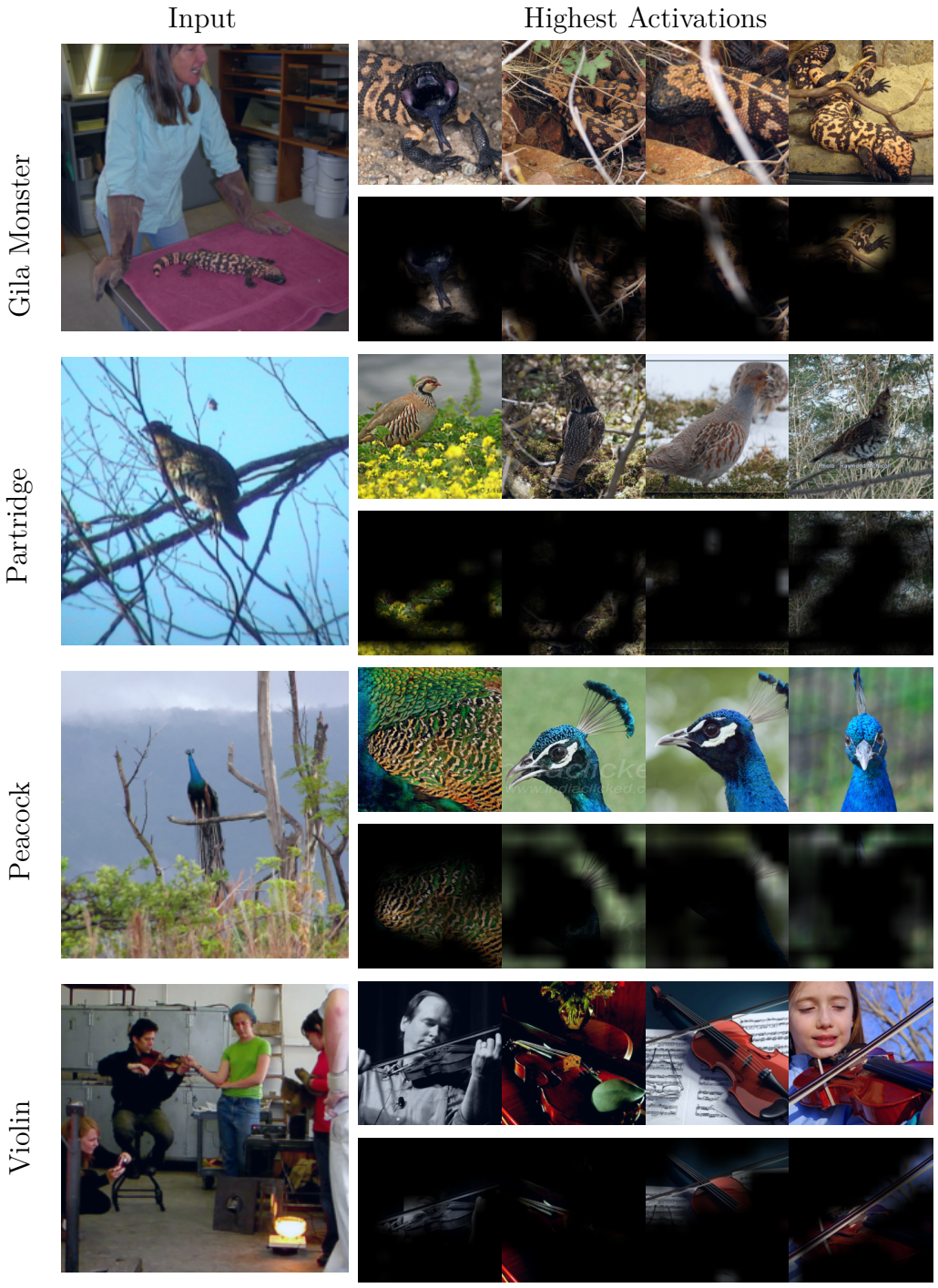

FIGURE 6.11: A misclassified input image (left column) is shown alongside the top four images (top row for each input) that maximally activates the expected filter from the training data within the same class. The bottom row for each input image is the product of the corresponding top four image and the expected filter.

thesis, relying on visual assessment alone allows human biases to creep in. In this section we move away from visual assessment and attempt to ‘fix images’ that have been misclassified. In this case fixing an image means modifying the existing image so that the network now classifies it correctly. No changes to the model itself are made. The following techniques are all supervised, they require knowledge of the ground truth class to ‘fix’ the image. As such they could not be deployed at runtime. However, they can be used post-hoc to visualise the reasons for failure in a more concrete way than visual assessment alone. Because of the separate approaches required to address the defects within images that cause surprise and expectation, we approach each one separately.

6.4.1 Suppressing Surprise

We begin by looking at the images that have activated a filter that is surprising to the network. This is the easier of the two tasks as there must be something present in the image that activated the filter. Knowing that there is some feature present in the image allows us an opportunity to locate and remove it.

We begin by passing every image from ImageNet’s validation set and identifying any images with surprising filter activations as per Section 6.2.4. We then use our modified SWAG technique (Section 6.3.2) to identify the regions in the image that are surprising the filter. Unlike the visualisation of surprise where we highlighted a single large superpixel, we use a range of superpixel sizes and amounts to remove and then iteratively delete them by setting any pixels within the superpixels to 0. At each iteration we check to see if the image is classified correctly or not. This should have the effect of suppressing the amount of surprise occurring in the identified filter allowing the network to use the other features present to better classify the image. Alongside our proposed method of using SWAG filtered through the surprised filter, we also conduct the same experiment using SWAG without the filtering. If our hypothesis that identifying the surprising filter is correct then we should be able to ‘fix’ more images than using the unfiltered gradients.

We show the results for this experiment in Figure 6.12 using VGG16 and ImageNet. This figure shows the number of image that were initially misclassified but are now able to be classified correctly. This value is shown for each combination of the amount of superpixels ([100, 200, 300, 400])

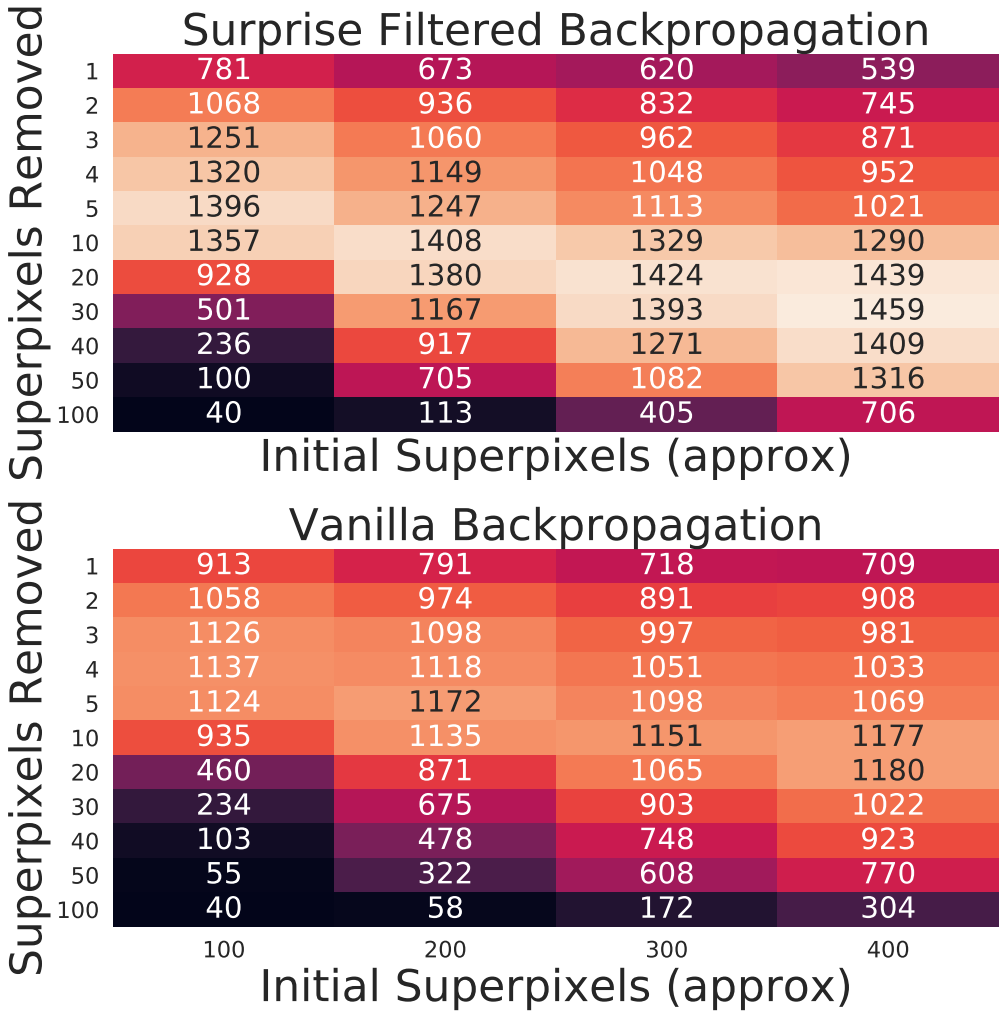

FIGURE 6.12: Results for suppressing the superpixels of both vanilla backpropagated gradients, and only the gradients backpropagated through the most surprising filter. These results are for VGG16.

and the number of superpixels set to 0 ([1,2,3,4,5,10,20,30,40,50,100]). Here, we see that the use of gradients backpropagated only through the surprised filter allows us to more precisely locate the detrimental regions of the image compared to backpropagating through all filters. This results in our method being able to ‘fix’ more misclassified images. Using VGG16 and sweeping through all the superpixel sizes and removal amounts, we are ultimately able to ‘fix’ 5,116 misclassified images, using our technique versus 4,554 using vanilla

backpropagation without surprise filtering. This is approximately $\sim 36\%$ and $\sim 32\%$ of the misclassified images respectively.

Interestingly, we found that using guided-backpropogation instead of the vanilla gradients actually performed poorer despite giving the best results for visual explanations when combined with SWAG in Chapter 3. When we conducted our experiment using guided-backpropagation with surprise filtering, we were only able to fix 4,841 misclassified images, a reduction of 275 images. We hypothesise that this is potentially due to the guided element of guided backpropagation where the gradients are only kept if they were positive during the forward pass. This in turn may may it less suitable for scoring pixels for a class that has not been predicted strongly. However, further experiments are needed to confirm this.

A nice by-product of performing this experiment is that we are able to produce visual explanations of the regions causing the misclassification such as those shown in Figure 6.13. These are distinct to the images used for visualising the surprise in Section 6.3.2 where we used a rough estimate (by simply visualising a large superpixel) to allow us to draw visual connections between the misclassified image and the training data. The visual explanations now created are much more precise in their identification of features through the use of superpixels produced at multiple scales and removal amounts. It is likely that more accurate explanations could be produced by using smaller intervals in both our superpixel amount and removal amounts, although this would increase the computational requirements. However, note that the region identified for the misclassified goldfish image from Figure 6.10 is shown to align closely with the region of the image deleted to make it classify correctly.

6.4.2 Correcting Expectation

A more challenging application of our method is correcting images that failed due to an expected filter not activating. This means the image did not contain enough of a certain feature to adequately activate the filter to ensure a correct classification. This becomes a difficult task as it is not a case of identifying and removing pixels as with suppressing surprise, rather we are required to identify and enhance or insert the required features. An example of this can be seen as the input in Figure 6.14. This shows an image from the ImageNet class gila

FIGURE 6.13: Surprise suppression examples from VGG16. All images initially misclassified, but ‘fixed’ using our method. Here the gray areas are those that have been deleted (set to 0).

monster that fails to be classified correctly. By visualising the training images that maximally activate the most expected filter, we gain an insight into which features the network is expecting to see. In Figure 6.14, we visualise the top six images for the expected filter. The first interesting aspect to notice is that none of the images seem to be activate the filter with the gila monster itself. Rather, the region surrounding the animal seems to activate the expected filter. The second is that all the images that activate the filter appear to have a regions surrounding the animal that is lightly coloured. We hypothesise that altering the surface found in our misclassified image to a colour more consistent with the training data will boost this filters activation score and the overall score from the network. Using a photo editing tool, we select the magenta pixels from the surface and make them white. This image now classifies correctly suggesting that our proposed technique has correctly highlighted the pixels in the input image that were inconsistent with the training data. This suggests there is a bias towards gila monsters that are located on certain surfaces. This technique has the potential to be a powerful addition to a developer’s toolkit when trying to debug a network. However, the additional analysis required by a user makes it difficult to use in practice. Whereas the technique used to visualise or suppress surprise was fully automated, a large amount of human

FIGURE 6.14: Top left: input. Top right: 'fixed' image. Bottom: the images and masked regions that most activate the expected filter.

intuition is currently required to produce corrected images from expected filters.

6.5 Future Work

We have shown that using the techniques proposed in this chapter, we are able to locate regions in an image that cause failure. A natural next step would be to see if we could modify the training data that causes the most confusion. This could be achieved by suppressing the regions of images (setting to 0) that a pre-trained model suggest causes surprise. In addition to this it would be interesting to see how this technique could work using other modalities. For example this could include video data or audio data in the form of spectrograms. The networks used to predict a classification for these forms of data are fundamentally similar to image networks, so should allow for our techniques to be easily transferable. For example using spectrograms to represent audio,

we would potentially be able to locate certain groups of frequencies that are confusing the network for a given prediction. This could potentially be a very valuable technique for the community as a whole.

6.6 Chapter Summary

In this chapter we have introduced a number of complimentary techniques, the first of which is an accurate way of ranking the importance of a convolution's filters. This in turn allows us to accurately build distributions representing how each filter reacts to every class present in the training data. With this pair of techniques we can begin to investigate a dataset using the β values for each image. Here the β value is the mean deviation across all filters when compared to the mean filter scores for the target class. We showed, using the β values, how the images of an unseen dataset can be explored to offers potential reasons for failure. We then showed that using our SWAG technique from Chapter 3, we can highlight regions within an image and remove them, allowing images to be classified correctly. Finally we show a method of exploration to visualise potentially missing features from an image, although in its current state this is a difficult undertaking.

CHAPTER

7 Conclusion

In this chapter, we summarise the contribution of each of the four chapters that form the body of this thesis, and how they answer our original research questions. In addition to this we also discuss the future work that could be undertaken to develop these contributions further.

This thesis was motivated by the need to have accurate, medium-grained methods for explaining why a CNN has produced a particular prediction. In addition to this we highlighted an area where current explanations were unable to contribute - the explanation of failure. To this end, three novel methods were proposed: two methods for creating medium-grained explanations, and one for explaining the reasons behind a networks failure.

The first of these methods, Superpixels Weighted by Average Gradient (SWAG), was presented in Chapter 3. This contribution addresses our first research goal, the understanding of whether gradients can be pooled into regions to create medium-grained explanations in an efficient manner. In SWAG, we proposed a method that allows the coarseness of the explanation to be defined. Previously this was only available to perturbation methods, which often take a considerable amount of time to compute. By defining the regions to explain we then showed that they could be weighted effectively using backpropagated gradients. A secondary contribution that arose during this chapter was the use of backpropagated gradients to inform the superpixel creation process. By creating superpixels using a modified implementation

of SLIC, we were able to show that using these bespoke superpixels created regions that were allowed more accurate explanations to be made. Using SWAG and its associated superpixel creation methods, we demonstrated that we could produce explanations that had better local and global accuracy than coarse-grained methods, but were more interpretable than fine-grained methods.

With SWAG, we showed that explanations could be created successfully for CNN's that took an image as an input. In Chapter 4, we extended SWAG to work with video inputs (SWAG-V). This addressed our second research goal. We did this through the use of superpixels that used both spatial and temporal dimensions. This addressed an existing problem with a number of methods for creating video explanations whereby they were unable to accurately represent the temporal element. This was due to the compression of the temporal element throughout the CNN. Using SWAG-V, we were able to define regions that spanned the entire temporal element, and weight them using backpropagated gradients. We showed that this produced both better explanations in terms of accuracy, as well as being able to better localise the action occurring within the video.

While our SWAG methodologies presented in Chapters 3 and 4 showed that we could offer improvements over existing methods, activation-based methods remain popular. In particular, they were shown to be able to better localise objects when using images as an input, and provide better results in interpretability metrics. CAMs are a popular method of using activations from the final layer of a network, and as such there have been numerous methods introduced to create explanations in this manner. However, the majority of these techniques aim to improve the way that these activations are weighted prior to being combined to form the explanation.

Our third research goal was to investigate whether multiple CAM explanations could be combined to produce explanations that are finer than existing CAM methods. In Chapter 5, we introduced Jitter-CAM. By rescaling the original input image and generating multiple CAM explanations over this new image, we were able to produce explanations that were more accurate with only a small drop in interpretability. An additional benefit to this was a much improved ability to localise the objects within an image compared to previous CAM methods. While we contributed Jitter-CAM in this chapter, we also

demonstrated that a common metric in the CAM literature, faithfulness, is not a reliable measure of a good explanation technique. We proposed that this metric be discontinued in future explanation research.

The research in the Chapters 3, 4, and 5 was based around creating explanations that could offer some insight into why a network made the prediction it did. However, these techniques are primarily aimed at understanding why a *predict* correction was made. In Chapter 6, we explored how we could understand the reasons for failure. We proposed the use of surprise and expectation as a measure of how much an unseen image activated the filters of a convolution layer compared to those of the training images. In doing so we introduced a method for ranking the importance of a convolution layers filters that was shown to be more accurate than comparable methods. Using this method we were able to better understand how the model learnt to represent the training data, which in turn allowed us to begin to understand the reasons for failure. Taking this technique a step further, we were able to combine our SWAG technique as a method for showing which regions of the image were responsible for a significant subset of the misclassified images.

7.1 Future Work

Throughout this thesis, we have suggested additional work that could be undertaken to further our contributions. In this section, we present these suggestions.

With SWAG we showed that we could create medium grained explanations and out of all the contribution in this thesis was the one we were able to explore the most with. Indeed, one of the main potential avenues of future work (the use of SWAG with videos) was explored further. However, there were a number of interesting aspects of the work that we would like to investigate further. The first is to address the weak localisation ability of the SWAG technique. This could either be achieved through the creation of alternative superpixels, or through a different attribution method. Secondly, by focussing on measuring both the local and global accuracy, we concentrated on the deletion metrics. In the future we would be interested to run the insertion metrics for the SWAG work.

We extended SWAG to work with video inputs for use in action recognition networks, however, there are many aspects of these explanations that could still be explored. Primarily this involves extending SWAG-V to work with two-stream action recognition networks. This could be a fruitful area of research for explaining action recognition networks and is could allow the motion to be explained separately to the combined appearance and motion stream.

With Jitter-CAM we showed that we could improve the spatial resolution of existing CAM techniques. There a couple of avenues of future work that we would like to investigate further. Of particular interest would be to see if we could apply a similar technique to improve the temporal resolution when trying to explain networks using a video input. In addition we would like to investigate the possibility of combining explanations created at different resolutions. This could allow explanations that combine the best aspects of the various levels of coarseness that we have investigated in these chapters.

Finally, with surprise and expectation we would like to apply the techniques we proposed to investigate if the training data can be altered to remove regions which are detrimental to the network. Doing so could potentially allow us to improve the classification accuracy of the network. Another interesting improvement that could be made to this work is to combine it with techniques that apply labels to the filters. Doing so would allow us to further explain the reasons for failure e.g. a misclassified image activated surprisingly highly on the ‘striped’ filter.

Bibliography

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[2] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems (NIPS) 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[4] Facebook. Announcing PyTorch 1.0 for both research and production. https://developers.facebook.com/blog/post/2018/05/02/announcing-pytorch-1.0-for-research-production/, 2018.

[5] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv*, 2017.

[6] Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun, and Jeff Dean. A guide to deep learning in healthcare. *Nature medicine*, 25(1):24–29, 2019.

[7] Waqas Sultani, Chen Chen, and Mubarak Shah. Real-world anomaly detection in surveillance videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[8] Brody Huval, Tao Wang, Sameep Tandon, Jeff Kiske, Will Song, Joel Pazhayampallil, Mykhaylo Andriluka, Pranav Rajpurkar, Toki Migimatsu, Royce Cheng-Yue, et al. An empirical evaluation of deep learning on highway driving. *arXiv preprint arXiv:1504.01716*, 2015.

[9] Vitali Petsiuk, Abir Das, and Kate Saenko. RISE: randomized input sampling for explanation of black-box models. In *British Machine Vision Conference 2018, BMVC*, 2018.

[10] Piotr Dabkowski and Yarin Gal. Real time image saliency for black box classifiers. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6970âĂŞ6979, Red Hook, NY, USA, 2017. Curran Associates Inc.

[11] Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 3449–3457, Oct 2017.

[12] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. In *International Conference on Computer Vision (ICCV)*, 2015.

[13] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?": Explaining the predictions of any classifier. In *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016.

[14] Yun He, Soma Shirakabe, and Hirokatsu Kataoka. Human action recognition without human. *ECCV 2016 Workshops. ECCV 2016. Lecture Notes in Computer Science*, 9915, 2016.

[15] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 618–626, Oct 2017.

[16] Marko Robnik-Šikonja and Marko Bohanec. Perturbation-based explanations of prediction models. In *Human and machine learning*, pages 159–175. Springer, 2018.

[17] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.

[18] Todd Kulesza, Simone Stumpf, Margaret Burnett, Sherry Yang, Irwin Kwan, and Weng-Keen Wong. Too much, too little, or just right? ways explanations impact end users' mental models. In *2013 IEEE Symposium on Visual Languages and Human Centric Computing*, pages 3–10, 2013.

[19] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.

[20] Been Kim, Rajiv Khanna, and Sanmi Koyejo. Examples are not enough, learn to criticize! Criticism for interpretability. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[21] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3319–3327, July 2017.

[22] Ramprasaath R. Selvaraju, Prithvijit Chattopadhyay, Mohamed Elhoseiny, Tilak Sharma, Dhruv Batra, Devi Parikh, and Stefan Lee. Choose your neuron: Incorporating domain knowledge through neuron-importance. In *The European Conference on Computer Vision (ECCV)*, pages 540–556, September 2018.

[23] Aditya Chattopadhyay, Anirban Sarkar, Prantik Howlader, and Vineeth N. Balasubramanian. Grad-CAM++: Generalized gradient-based visual explanations for deep convolutional networks. In *IEEE Winter Conference on Applications of Computer Vision*, pages 839–847, 2018.

[24] Saurabh Desai and Harish Guruprasad Ramaswamy. Ablation-CAM: Visual explanations for deep convolutional network via gradient-free localization. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, March 2020.

[25] Scott Mayer McKinney, Marcin Sieniek, Varun Godbole, Jonathan Godwin, Natasha Antropova, Hutan Ashrafian, Trevor Back, Mary Chesus, Greg S Corrado, Ara Darzi, et al. International evaluation of an AI system for breast cancer screening. *Nature*, 577(7788):89–94, 2020.

[26] Li Shen, Laurie R Margolies, Joseph H Rothstein, Eugene Fluder, Russell McBride, and Weiva Sieh. Deep learning to improve breast cancer detection on screening mammography. *Scientific reports*, 9(1):1–12, 2019.

[27] G Sreenu and MA Saleem Durai. Intelligent video surveillance: a review through deep learning techniques for crowd analysis. *Journal of Big Data*, 6(1):1–27, 2019.

[28] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[29] Christoph Molnar. *Interpretable Machine Learning*. 2019. https://christophm.github.io/interpretable-ml-book/.

[30] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *2nd International Conference on Learning Representations, ICLR 2014, Workshop Track Proceedings*, 2014.

[31] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. Striving for simplicity: The all convolutional net. In *3rd International Conference on Learning Representations, ICLR 2015, Workshop Track Proceedings*, 2015.

[32] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *34th International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 3145–3153. PMLR, 06–11 Aug 2017.

[33] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3319âĂŞ3328. JMLR.org, 2017.

[34] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929, 2016.

[35] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In *International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 2668–2677. PMLR, 10–15 Jul 2018.

[36] Alexandros Stergiou, Georgios Kapidis, Grigorios Kalliatakis, Christos Chrysoulas, Remco Veltkamp, and Ronald Poppe. Saliency tubes: Visual explanations for spatio-temporal convolutions. *arXiv preprint arXiv:1902.01078*, 2019.

[37] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3D convolutional networks. In *The IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 4489–4497, Washington, DC, USA, 2015. IEEE Computer Society.

[38] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6450–6459, 2018.

[39] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'14, pages 568–576, Cambridge, MA, USA, 2014. MIT Press.

[40] Christoph Feichtenhofer, Axel Pinz, and Richard Wildes. Spatiotemporal residual networks for video action recognition. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[41] Zhenqiang Li, Weimin Wang, Zuoyue Li, Yifei Huang, and Yoichi Sato. Towards visually explaining video understanding networks with perturbation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1120–1129, January 2021.

[42] Liam Hiley, Alun Preece, Yulia Hicks, Supriyo Chakraborty, Prudhvi Gurram, and Richard Tomsett. Explaining motion relevance for activity recognition in video deep learning models. *arXiv preprint arXiv:2003.14285*, 2020.

[43] Alun Preece, Dan Harborne, Dave Braines, Richard Tomsett, and Supriyo Chakraborty. Stakeholders in explainable AI. *arXiv preprint arXiv:1810.00184*, 2018.

[44] Thomas Hartley, Kirill Sidorov, Christopher Willis, and David Marshall. Gradient weighted superpixels for interpretability in CNNs. In *BMVC 2019: Workshop on Interpretable and Explainable Machine Vision*, Cardiff, UK, September 2019.

[45] Thomas Hartley, Kirill Sidorov, Christopher Willis, and David Marshall. SWAG: Superpixels weighted by average gradients for explanations of CNNs. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 423–432, January 2021.

[46] Thomas Hartley, Kirill Sidorov, Christopher Willis, and David Marshall. Explaining failure: Investigation of surprise and expectation in CNNs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[47] J. R. Quinlan. Induction of decision trees. *MACH. LEARN*, 1:81–106, 1986.

[48] Deboleena Roy, Priyadarshini Panda, and Kaushik Roy. Tree-CNN: A hierarchical deep convolutional neural network for incremental learning. *Neural Networks*, 121:148–160, 2020.

[49] Zachary C. Lipton. The mythos of model interpretability. *Commun. ACM*, 61(10):36âĂŞ43, September 2018.

[50] Quanshi Zhang, Xin Wang, Y. Wu, Hui lin Zhou, and Song-Chun Zhu. Interpretable cnns for object classification. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[51] Mengnan Du, Ninghao Liu, and Xia Hu. Techniques for interpretable machine learning. *Commun. ACM*, 63(1):68–77, December 2019.

[52] W. James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, 2019.

[53] Stan Benjamens, Pranavsingh Dhunnoo, and Bertalan Meskó. The state of artificial intelligence-based fda-approved medical devices and algorithms: an online database. *NPJ digital medicine*, 3(1):1–8, 2020.

[54] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. How to explain individual classification decisions. *J. Mach. Learn. Res.*, 11:1803–1831, August 2010.

[55] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10(7), 07 2015.

[56] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg. SmoothGrad: removing noise by adding noise. *ICML workshop on visualization for deep learning*, June 2017.

[57] Luisa M. Zintgraf, Taco S. Cohen, Tameem Adel, and Max Welling. Visualizing deep neural network decisions: Prediction difference analysis. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[58] Ruth Fong and Andrea Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. In *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

[59] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NIPS) 30*, pages 4765–4774. Curran Associates, Inc., 2017.

[60] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. In *International Conference on Learning Representations*, 2018.

[61] Matthew D Zeiler, Graham W Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *2011 International Conference on Computer Vision*, pages 2018–2025. IEEE, 2011.

[62] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.

[63] Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. Not just a black box: Learning important features through propagating activation differences. *CoRR*, abs/1605.01713, 2016.

[64] Jianming Zhang, Sarah Adel Bargal, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down neural attention by excitation backprop. *International Journal of Computer Vision*, 126(10):1084–1102, 2018.

[65] Andrei Kapishnikov, Tolga Bolukbasi, Fernanda Viegas, and Michael Terry. XRAI: Better attributions through regions. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[66] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

[67] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

[68] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016.

[69] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[70] Ruigang Fu, Qingyong Hu, Xiaohu Dong, Yulan Guo, Yinghui Gao, and Biao Li. Axiom-based Grad-CAM: Towards accurate visualization and explanation of CNNs. In *British Machine Vision Conference*, 2020.

[71] GrÃľgoire Montavon, Wojciech Samek, and Klaus-Robert MÃijller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1 – 15, 2018.

[72] Haofan Wang, Zifan Wang, Mengnan Du, Fan Yang, Zijian Zhang, Sirui Ding, Piotr Mardziel, and Xia Hu. Score-CAM: Score-weighted visual explanations for convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[73] Sylvestre-Alvise Rebuffi, Ruth Fong, Xu Ji, Hakan Bilen, and Andrea Vedaldi. Normgrad: Finding the pixels that matter for training. *arXiv preprint arXiv:1910.08823*, 2019.

[74] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells III, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention 2015 - 18th International Conference*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241. Springer, 2015.

[75] Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953.

[76] Mengjiao Yang and Been Kim. Benchmarking Attribution Methods with Relative Feature Importance. *CoRR*, abs/1907.09701, 2019.

[77] Mukund Sundararajan, Jinhua Xu, Ankur Taly, Rory Sayres, and Amir Najmi. Exploring principled visualizations for deep network attributions. In *IUI Workshops*, volume 4, 2019.

[78] Anh Nguyen, Jason Yosinski, and Jeff Clune. Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. *arXiv preprint arXiv:1602.03616*, 2016.

[79] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5188–5196, 2015.

[80] Alexey Dosovitskiy and Thomas Brox. Inverting visual representations with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[81] Dumitru Erhan, Y Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *Technical Report, UniveristÃľ de MontrÃľal*, 01 2009.

[82] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, page IâĂŞ647âĂŞIâĂŞ655. JMLR.org, 2014.

[83] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks, 2015.

[84] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2(11):e7, 2017.

[85] Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. The building blocks of interpretability. *Distill*, 3(3):e10, 2018.

[86] Andrej Karpathy. t-SNE visualization of CNN codes. https://cs.stanford.edu/people/karpathy/cnnembed/.

[87] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11), 2008.

[88] Shan Carter, Zan Armstrong, Ludwig Schubert, Ian Johnson, and Chris Olah. Activation atlas. *Distill*, 4(3):e15, 2019.

[89] Donglai Wei, Bolei Zhou, Antonio Torrabla, and William Freeman. Understanding intra-class knowledge inside CNN. *arXiv preprint arXiv:1507.02379*, 2015.

[90] Yann Le Cun, John S. Denker, and Sara A. Solla. *Optimal Brain Damage*, pages 598–605. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[91] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene CNNs. In *International Conference on Learning Representations (ICLR)*, 2015.

[92] Ruth Fong and Andrea Vedaldi. Net2Vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2018.

[93] Amirata Ghorbani, James Wexler, James Y Zou, and Been Kim. Towards automatic concept-based explanations. In *Advances in Neural Information Processing Systems*, pages 9273–9282, 2019.

[94] K. S. Gurumoorthy, A. Dhurandhar, G. Cecchi, and C. Aggarwal. Efficient data representation by selecting prototypes with importance weights. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 260–269, 2019.

[95] BBC. Uber in fatal crash had safety flaws say us investigators. https://www.bbc.co.uk/news/business-50312340, 2019.

[96] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6299–6308, 2017.

[97] Liam Hiley, Alun Preece, Yulia Hicks, David Marshall, and Harrison Taylor. Discriminating spatial and temporal relevance in deep Taylor decompositions for explainable activity recognition. *arXiv preprint arXiv:1908.01536*, 2019.

[98] Sarah Adel Bargal, Andrea Zunino, Donghyun Kim, Jianming Zhang, Vittorio Murino, and Stan Sclaroff. Excitation backprop for rnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[99] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):677–691, 2017.

[100] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (un) reliability of saliency methods. *arXiv preprint arXiv:1711.00867*, 2017.

[101] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems (NIPS)*, pages 9525–9536, 2018.

[102] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 9734–9745. Curran Associates, Inc., 2019.

[103] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28(11):2660–2673, 2017.

[104] Pieter-Jan Kindermans, Kristof T. Schütt, Maximilian Alber, Klaus-Robert Müller, Dumitru Erhan, Been Kim, and Sven Dähne. Learning how to explain neural networks: PatternNet and pattern attribution. In *International Conference on Learning Representations*, 2018.

[105] Richard Tomsett, Dan Harborne, Supriyo Chakraborty, Prudhvi Gurram, and Alun Preece. Sanity checks for saliency metrics. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

[106] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

[107] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

[108] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[109] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[110] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[111] Ertunc Erdil, Sinan Yildirim, Mujdat Cetin, and Tolga Tasdizen. Mcmc shape sampling for image segmentation with nonparametric shape priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[112] Zhuowen Tu and Song-Chun Zhu. Image segmentation by data-driven markov chain monte carlo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):657–673, 2002.

[113] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181, 2004.

[114] David Stutz, Alexander Hermans, and Bastian Leibe. Superpixels: An evaluation of the state-of-the-art. *Computer Vision and Image Understanding*, 166:1–27, 2018.

[115] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[116] Rémi Giraud, Vinh-Thong Ta, Nicolas Papadakis, and Yannick Berthoumieu. Texture-Aware Superpixel Segmentation. In *IEEE International Conference on Image Processing*, Taipei, Taiwan, September 2019.

[117] David Weikersdorfer, David Gossow, and Michael Beetz. Depth-adaptive superpixels. In *Proceedings of the 21st international conference on pattern recognition (ICPR2012)*, pages 2087–2090. IEEE, 2012.

[118] Commission International de L'Eclairage. Colorimetry. *CIE Pub*, 15(2):29–30, 1986.

[119] Jorg Wagner, Jan Mathias Kohler, Tobias Gindele, Leon Hetzel, Jakob Thaddaus Wiedemer, and Sven Behnke. Interpretable and fine-grained visual explanations for convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9089–9099, June 2019.

[120] Vitali Petsiuk, Abir Das, and Kate Saenko. RISE: randomized input sampling for explanation of black-box models. https://github.com/eclique/RISE, 2018.

[121] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(6):1452–1464, 2018.

[122] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[123] Finale Doshi-Velez and Been Kim. Considerations for evaluation and generalization in interpretable machine learning. In *Notions and Concepts on Explainability and Interpretability*, pages 3–17. Springer, 2018.

[124] Andrea Vedaldi and Stefano Soatto. Quick shift and kernel methods for mode seeking. In *European Conference on Computer Vision*, pages 705–718. Springer, 2008.

[125] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?": Explaining the predictions of any classifier. https://github.com/marcotcr/lime, 2016.

[126] Andrei Kapishnikov, Tolga Bolukbasi, Fernanda Viégas, and Michael Terry. Xrai: Better attributions through regions. https://github.com/PAIR-code/saliency, 2019.

[127] Irwin Sobel and G. Feldman. A 3x3 isotropic gradient operator for image processing. *Pattern Classification and Scene Analysis*, pages 271–272, 01 1973.

[128] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[129] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[130] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Colorado Springs, CO, June 2011.

[131] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the 2008 Sixth Indian Conference on Computer Vision, Graphics and Image Processing*, pages 722–729, 2008.

[132] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[133] Chunshui Cao, Xianming Liu, Yi Yang, Yinan Yu, Jiang Wang, Zilei Wang, Yongzhen Huang, Liang Wang, Chang Huang, Wei Xu, Deva Ramanan, and Thomas S. Huang. Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 2956–2964, December 2015.

[134] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In F. Pereira,

C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[135] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Fei Fei Li. Large-scale video classification with convolutional neural networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[136] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014.

[137] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[138] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional Two-Stream Network Fusion for Video Action Recognition. *Cvpr*, (i):1933–1941, 2016.

[139] Gul Varol, Ivan Laptev, and Cordelia Schmid. Long-term Temporal Convolutions for Action Recognition. *Hal-01241518*, pages 1–9, 2015.

[140] Georgia Gkioxari and Jitendra Malik. Finding action tubes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[141] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3D convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2013.

[142] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Sequential deep learning for human action recognition. In *International Workshop on Human Behavior Understanding*, pages 29–39. Springer, 2011.

[143] Rohit Girdhar, Deva Ramanan, Abhinav Gupta, Josef Sivic, and Bryan Russell. Actionvlad: Learning spatio-temporal aggregation for action classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[144] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4694–4702, 2015.

[145] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[146] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3D residual networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[147] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402, 2012.

[148] Bernard Ghanem Fabian Caba Heilbron, Victor Escorcia and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–970, 2015.

[149] Haroon Idrees, Amir R Zamir, Yu-Gang Jiang, Alex Gorban, Ivan Laptev, Rahul Sukthankar, and Mubarak Shah. The THUMOS challenge on action recognition for videos "in the wild". *Computer Vision and Image Understanding*, 2016.

[150] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The Kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[151] Jose M Chaquet, Enrique J Carmona, and Antonio Fernández-Caballero. A survey of video datasets for human action and activity recognition. *Computer Vision and Image Understanding*, 117(6):633–659, 2013.

[152] Yun He, Soma Shirakabe, Yutaka Satoh, and Hirokatsu Kataoka. Human action recognition without human. In *European Conference on Computer Vision*, pages 11–17. Springer, 2016.

[153] João Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A short note about Kinetics-600. *CoRR*, abs/1808.01340, 2018.

[154] João Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. A short note on the Kinetics-700 human action dataset. *CoRR*, abs/1907.06987, 2019.

[155] Jingen Liu, Jiebo Luo, and Mubarak Shah. Recognizing realistic actions from videos "in the wild". In *Computer vision and pattern recognition, 2009. CVPR 2009. IEEE conference on*, pages 1996–2003. IEEE, 2009.

[156] Kishore K Reddy and Mubarak Shah. Recognizing 50 human action categories of web videos. *Machine Vision and Applications*, 24(5):971–981, 2013.

[157] Jan C. van Gemert, Mihir Jain, Ella Gati, and Cees G. M. Snoek. Apt: Action localization proposals from dense trajectories. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 177.1–177.12. BMVA Press, September 2015.

[158] Saumya Jetley, Nicholas A. Lord, Namhoon Lee, and Philip Torr. Learn to pay attention. In *International Conference on Learning Representations*, 2018.

[159] Hyungsik Jung and Youngrock Oh. LIFT-CAM: Towards better explanations for class activation mapping. *arXiv preprint arXiv:2102.05228*, 2021.

[160] David Alvarez-Melis and Tommi S. Jaakkola. Towards robust interpretability with self-explaining neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 7786âĂŞ7795, Red Hook, NY, USA, 2018. Curran Associates Inc.

[161] Zhongang Qi, Saeed Khorram, and Li Fuxin. Visualizing deep networks by optimizing with integrated gradients. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):11890–11898, Apr. 2020.

[162] Yunchao Wei, Jiashi Feng, Xiaodan Liang, Ming-Ming Cheng, Yao Zhao, and Shuicheng Yan. Object region mining with adversarial erasing: A

simple classification to semantic segmentation approach. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[163] Huu-Giao Nguyen, Alessia Pica, Jan Hrbacek, Damien C. Weber, Francesco La Rosa, Ann Schalenbourg, Raphael Sznitman, and Meritxell Bach Cuadra. A novel segmentation framework for uveal melanoma in magnetic resonance imaging based on class activation maps. In M. Jorge Cardoso, Aasa Feragen, Ben Glocker, Ender Konukoglu, Ipek Oguz, Gozde Unal, and Tom Vercauteren, editors, *Proceedings of The 2nd International Conference on Medical Imaging with Deep Learning*, volume 102 of *Proceedings of Machine Learning Research*, pages 370–379, London, United Kingdom, 08–10 Jul 2019. PMLR.

[164] Y. Wang, F. Zhu, C. J. Boushey, and E. J. Delp. Weakly supervised food image segmentation using class activation maps. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 1277–1281, 2017.

[165] Kun Fu, Wei Dai, Yue Zhang, Zhirui Wang, Menglong Yan, and Xian Sun. MultiCAM: Multiple class activation mapping for aircraft recognition in remote sensing images. *Remote Sensing*, 11(5), 2019.

[166] Wenjie Yang, Houjing Huang, Zhang Zhang, Xiaotang Chen, Kaiqi Huang, and Shu Zhang. Towards rich feature discovery with class activation maps augmentation for person re-identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[167] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *CoRR*, abs/1611.06440, 2016.

[168] Mikhail Figurnov, Aizhan Ibraimova, Dmitry P Vetrov, and Pushmeet Kohli. PerforatedCNNs: Acceleration through elimination of redundant convolutions. In *Advances in Neural Information Processing Systems*, pages 947–955, 2016.

[169] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[170] Ye Yuan, Guangxu Xun, Fenglong Ma, Qiuling Suo, Hongfei Xue, Kebin Jia, and Aidong Zhang. A novel channel-aware attention framework for multi-channel eeg seizure detection via multi-view deep learning. In *2018 IEEE EMBS International Conference on Biomedical Health Informatics (BHI)*, pages 206–209, 2018.

[171] Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. Discrimination-aware channel pruning for deep neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[172] Hao Tang, Dan Xu, Nicu Sebe, Yanzhi Wang, Jason J. Corso, and Yan Yan. Multi-channel attention selection gan with cascaded semantic guidance for cross-view image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[173] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.

Appendix

A Additional SWAG Examples

Additional examples of explanations created using the datasets experimented with in Chapter 3 can be found here.

- Examples of explanations for the **Stanford Dogs** dataset can be found in Figures A.1 and A.2.
- Examples of explanations for the **Caltech-UCSD Birds 200** dataset can be found in Figures A.3 and A.4.
- Examples of explanations for the **Oxford Flowers** dataset can be found in Figures A.5 and A.6.

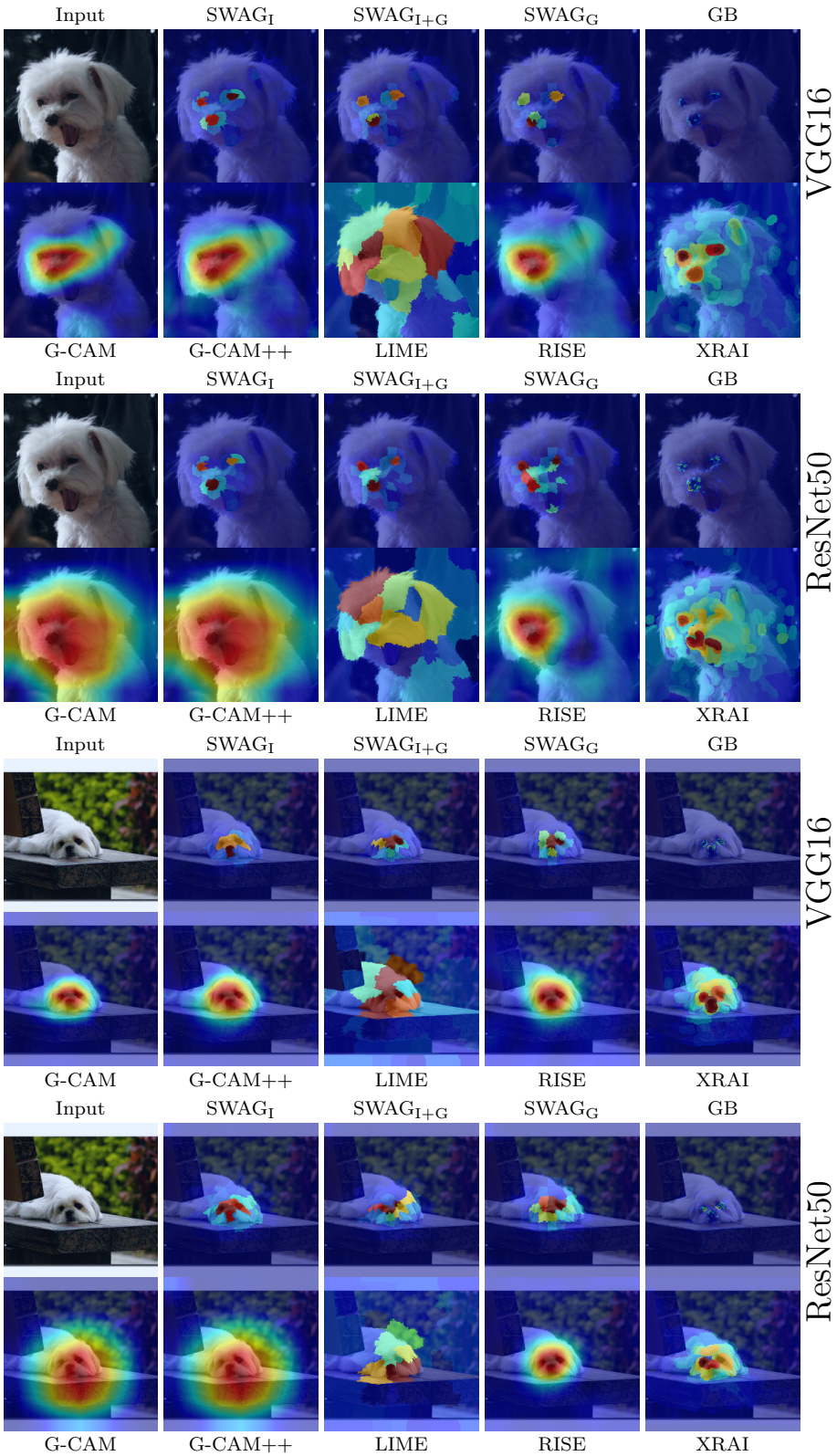

FIGURE A.1: Qualitative comparison between methods using examples from Stanford Dogs with ResNet50 and VGG16.

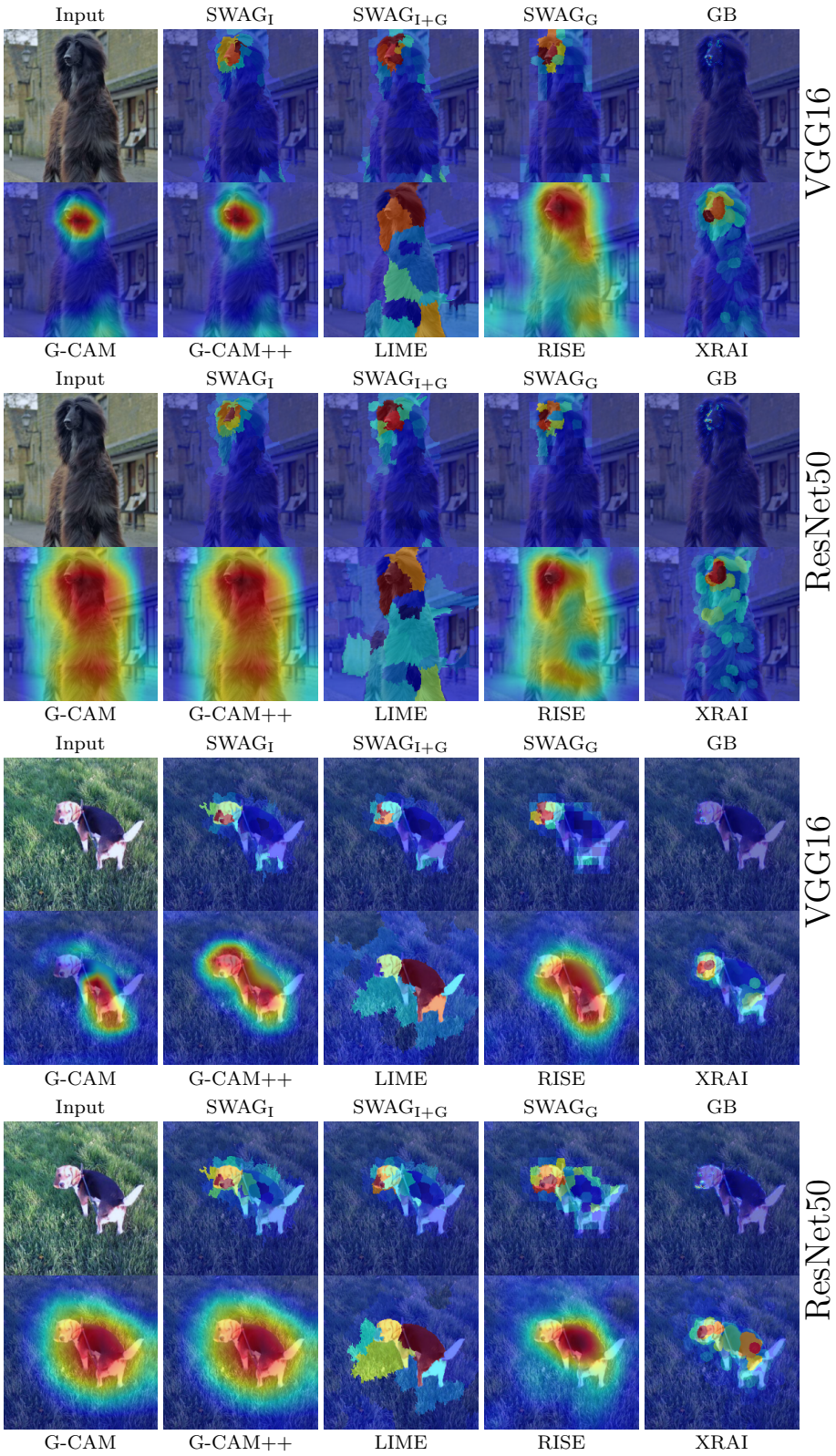

FIGURE A.2: Qualitative comparison between methods using examples from Stanford Dogs with ResNet50 and VGG16.

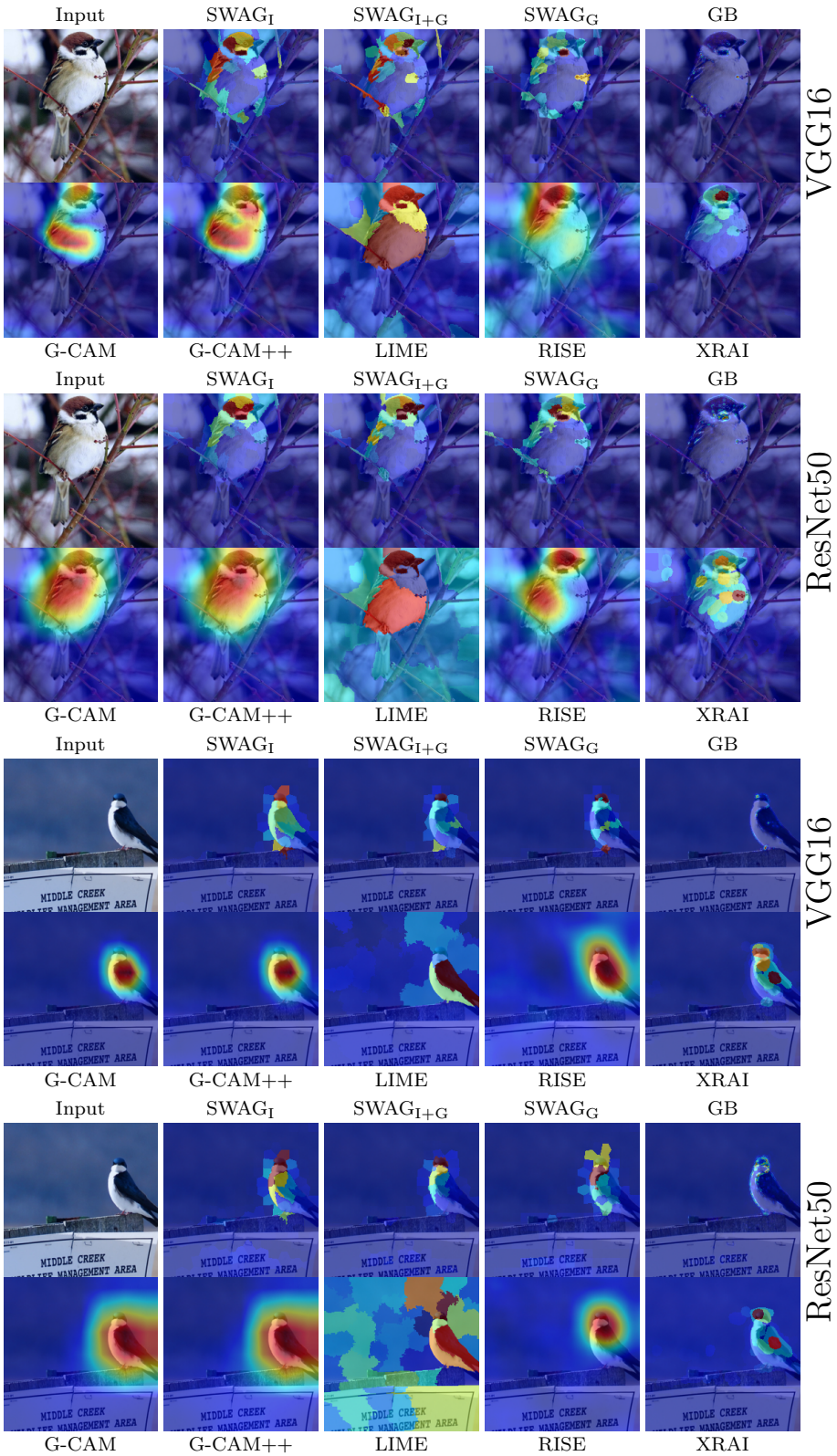

FIGURE A.3: Qualitative comparison between methods using examples from CUB200 with ResNet50 and VGG16.

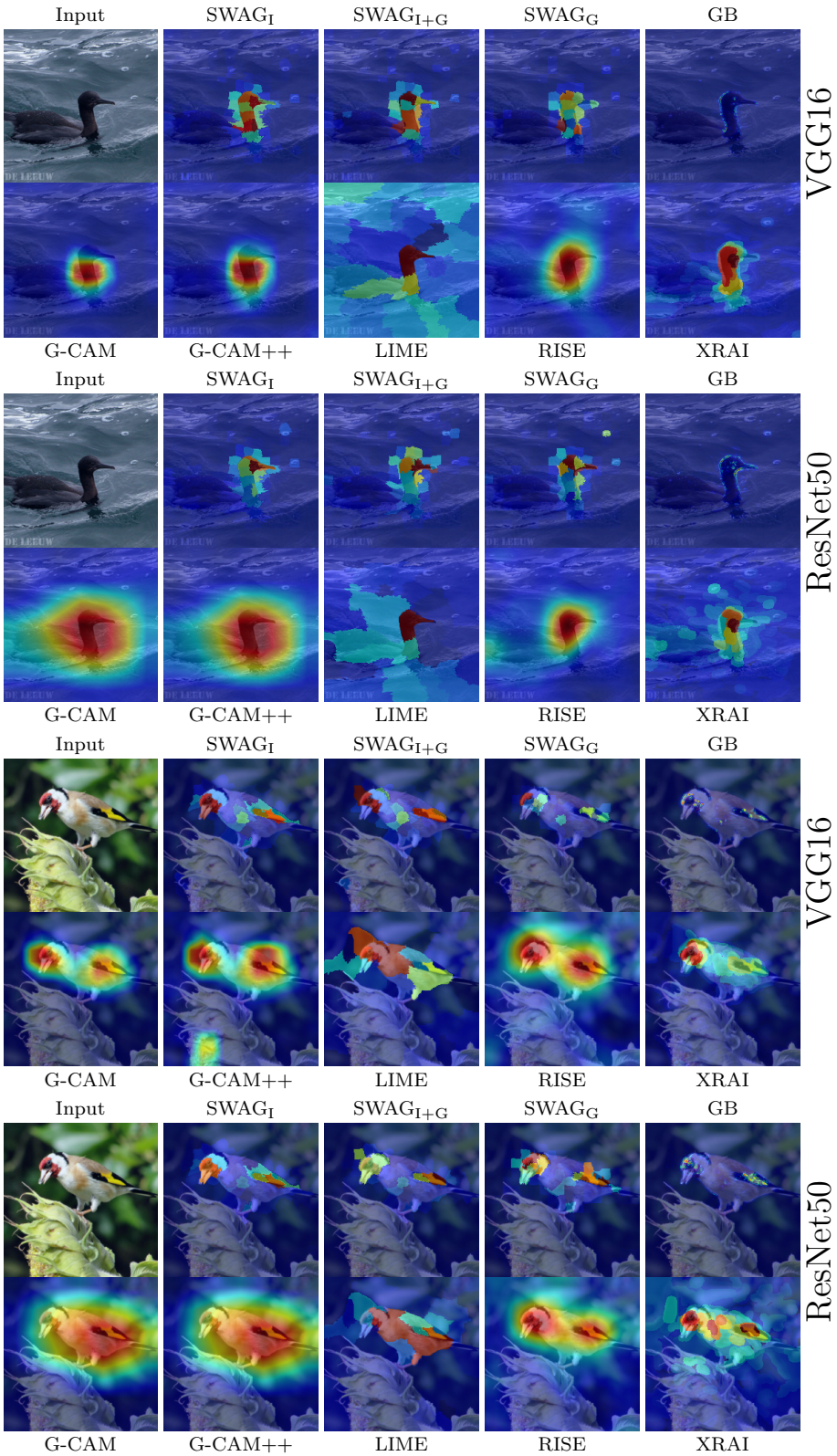

FIGURE A.4: Qualitative comparison between methods using examples from CUB200 with ResNet50 and VGG16.

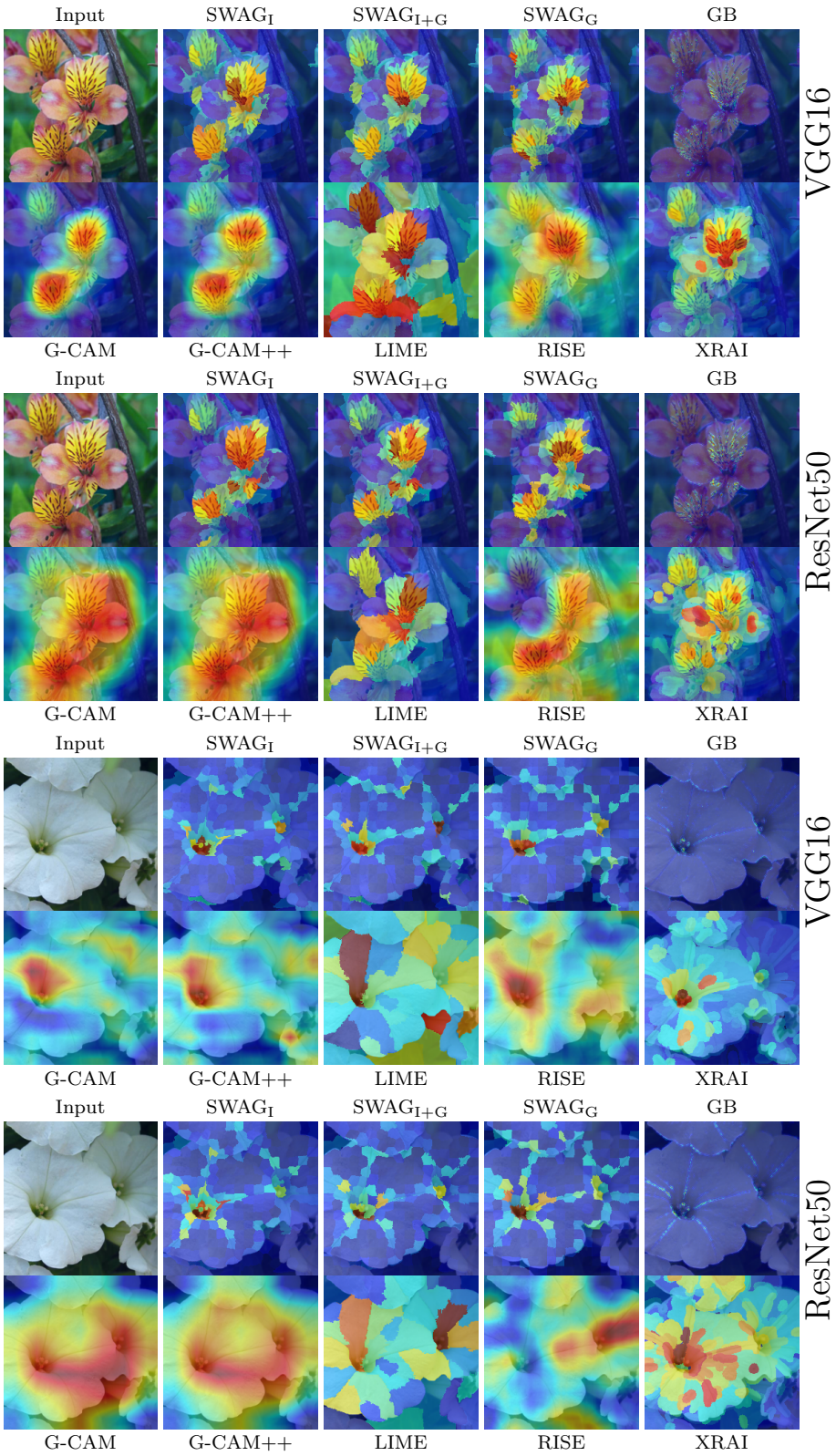

FIGURE A.5: Qualitative comparison between methods using examples from Oxford Flowers with ResNet50 and VGG16.

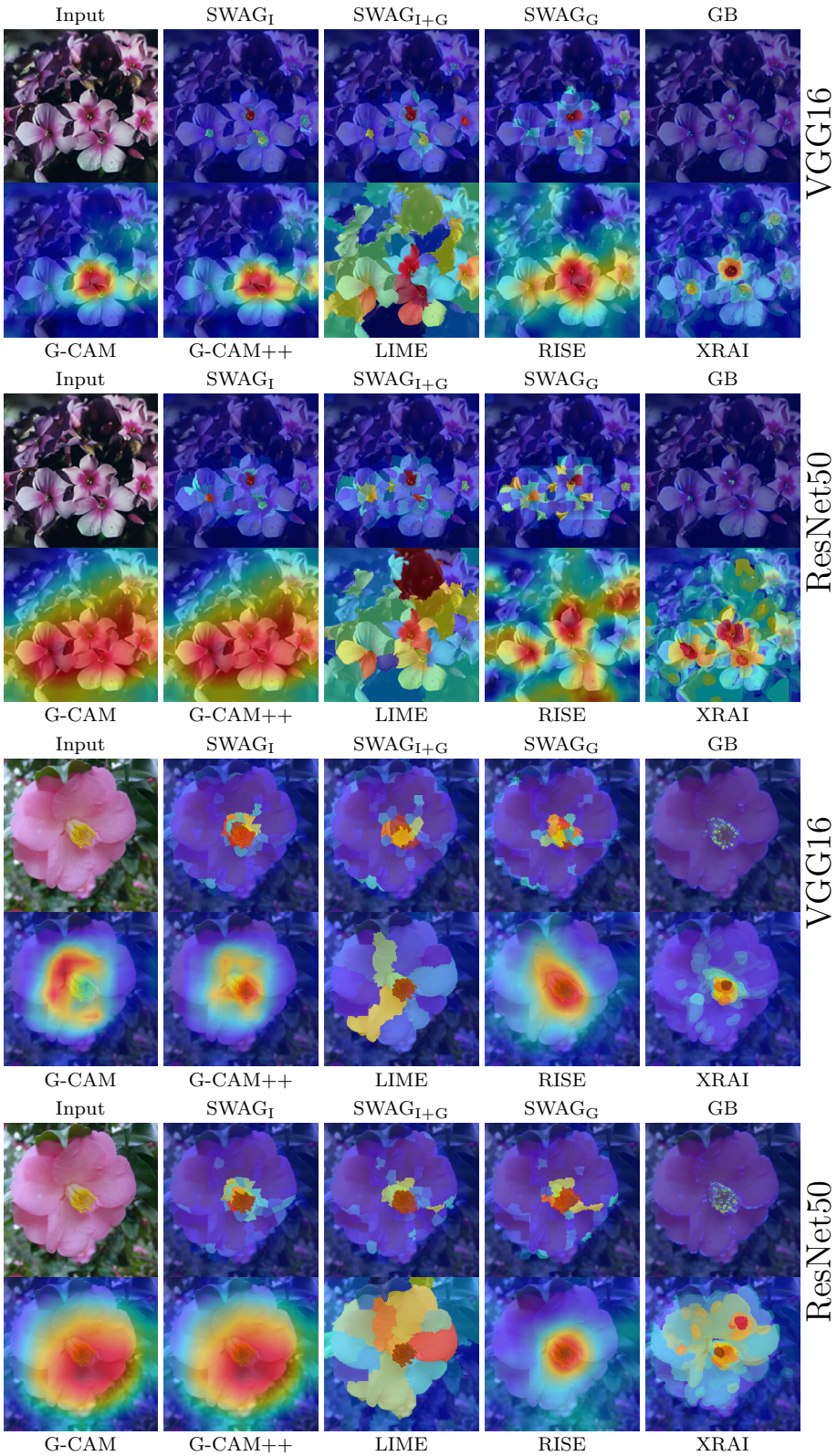

FIGURE A.6: Qualitative comparison between methods using examples from Oxford Flowers with ResNet50 and VGG16.

Appendix

B Additional SWAG-V Examples

Additional examples of explanations created for Chapter 4 using Kinetics 400 and both R(2+1)D and C3D. The following classes are shown:

- Skiing slalom in Figure B.1 (R(2+1)D) and Figure B.2 (C3D).
- Bobsledding in Figure B.3 (R(2+1)D) and Figure B.4 (C3D).
- Bungee jumping in Figure B.5 (R(2+1)D) and Figure B.6 (C3D).
- Skiing crosscountry in Figure B.7 (R(2+1)D) and Figure B.8 (C3D).

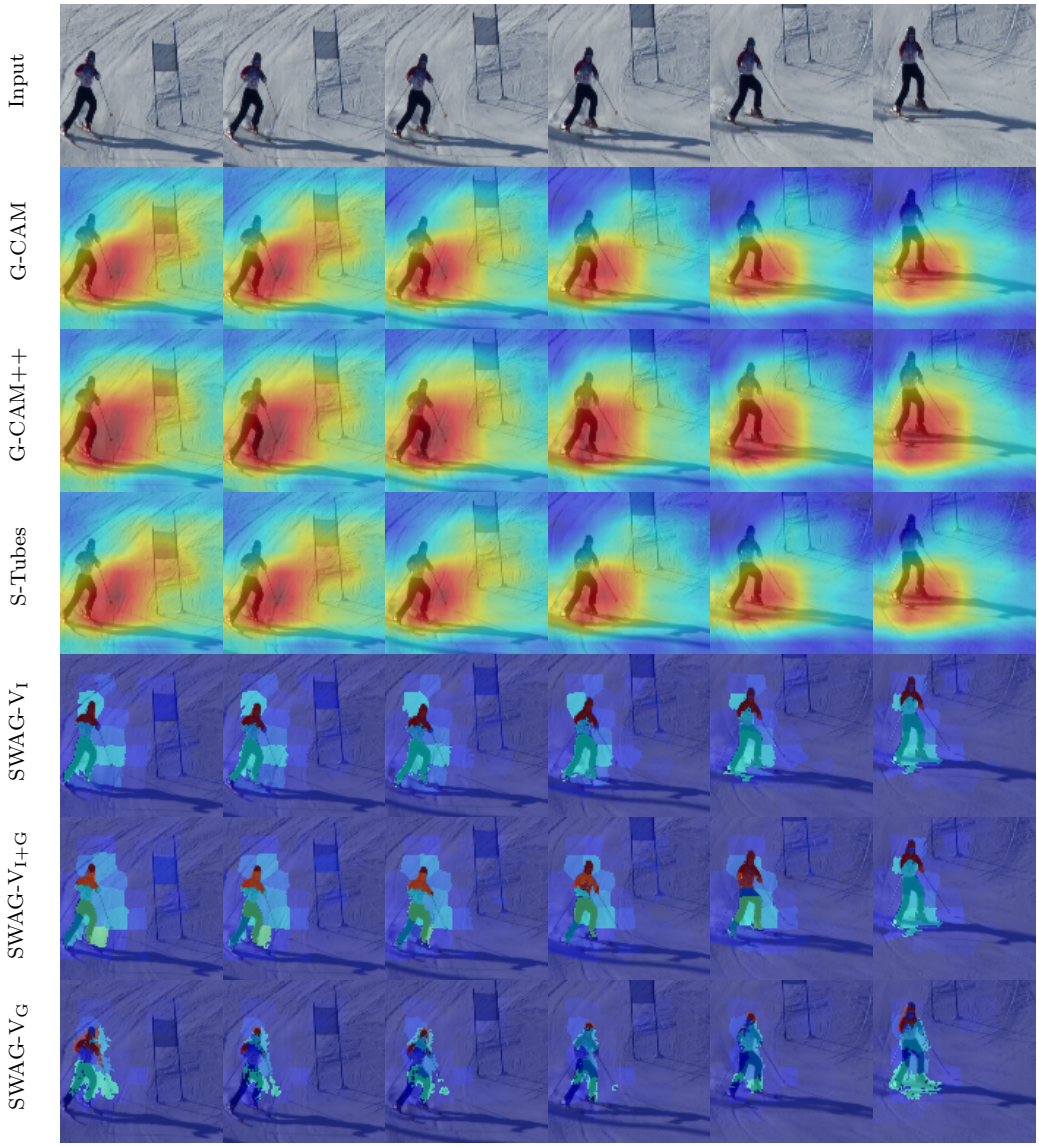

FIGURE B.1: An explanation for the Kinetics 400 class skiing slalom using R(2+1)D.

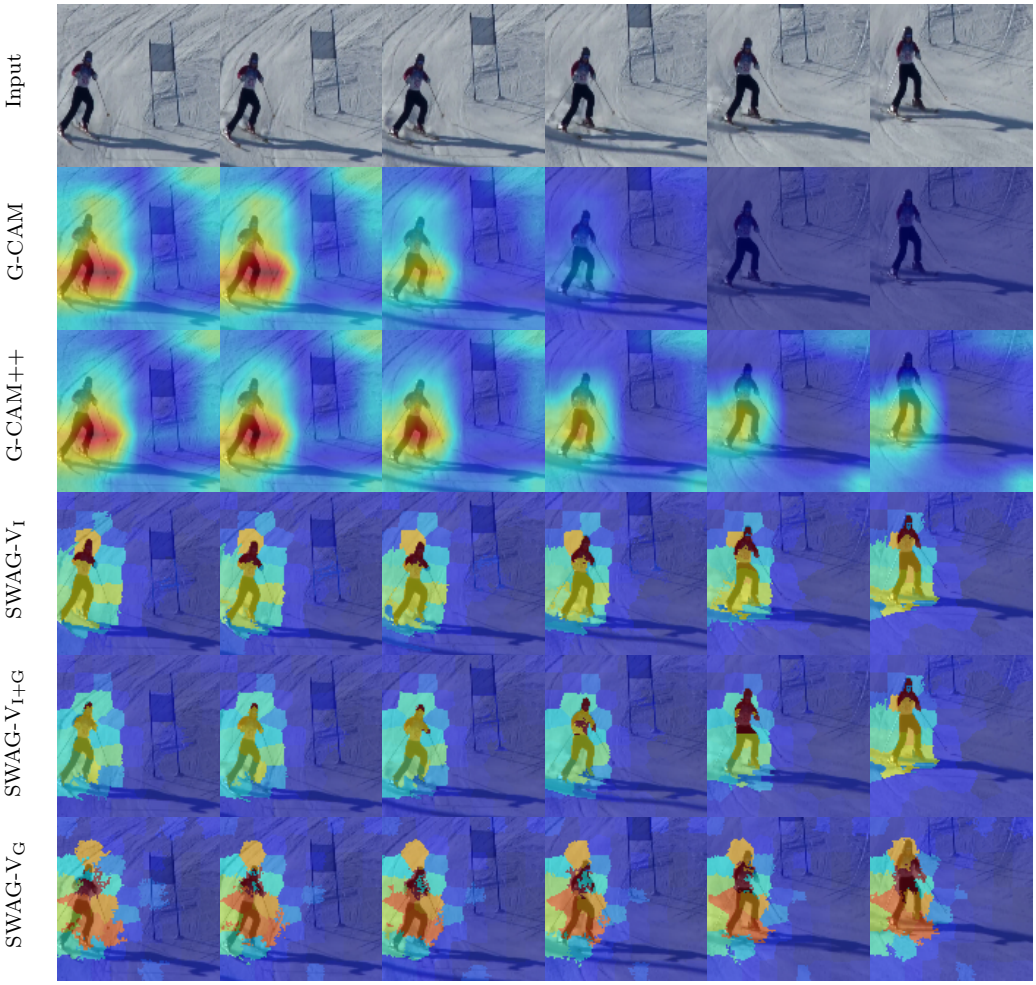

FIGURE B.2: An explanation for the Kinetics 400 class skiing slalom using C3D.

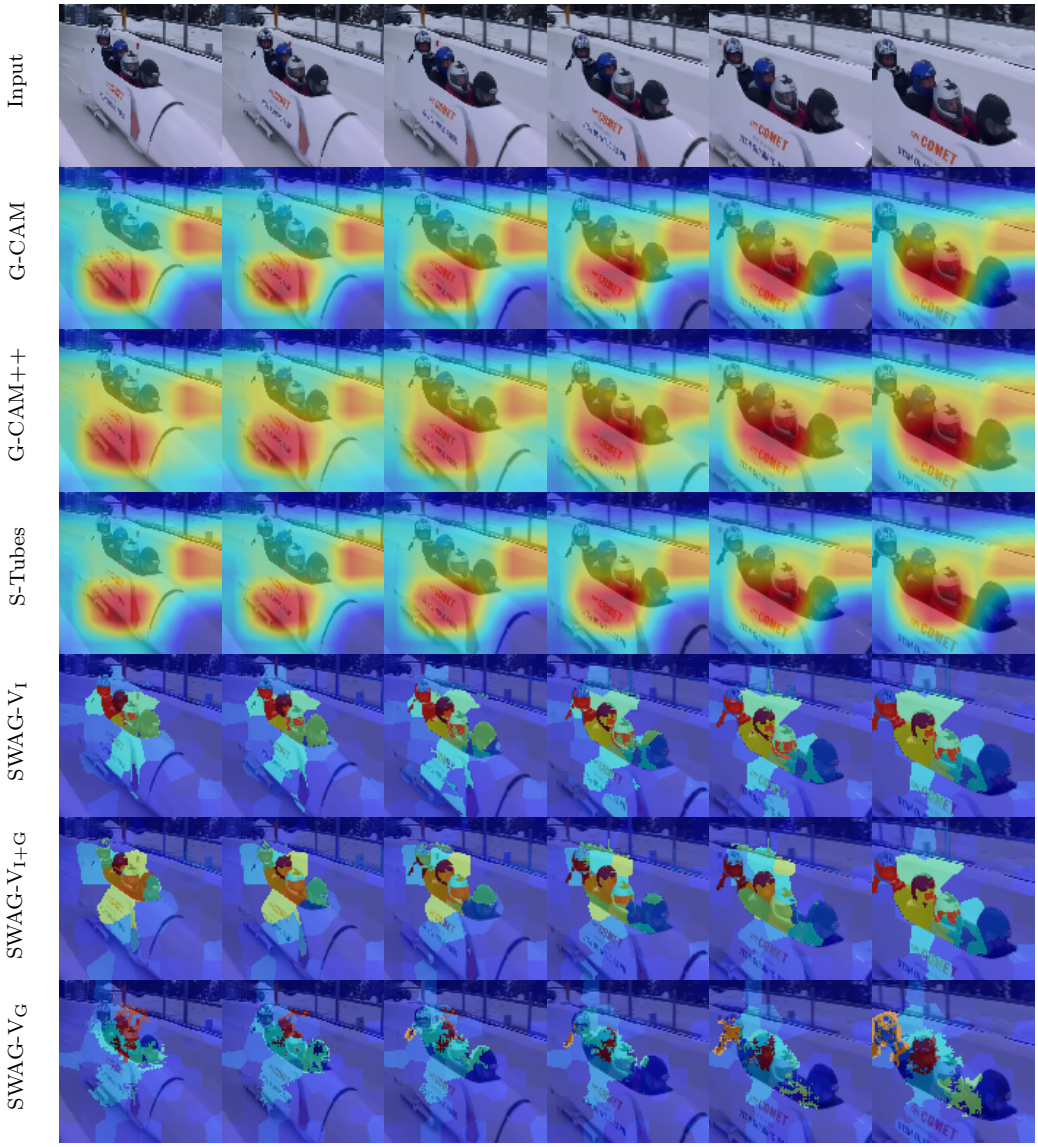

FIGURE B.3: An explanation for the Kinetics 400 class bobsledding using R(2+1)D.

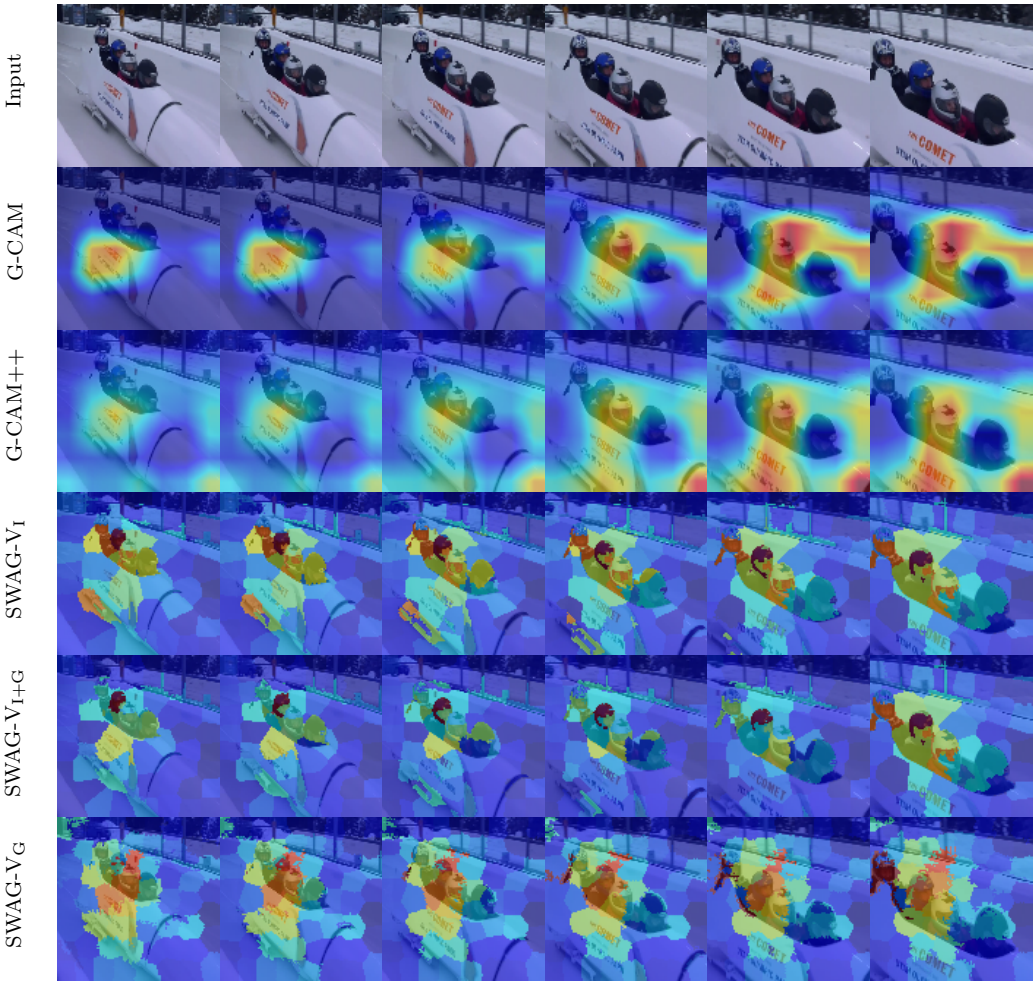

FIGURE B.4: An explanation for the Kinetics 400 class bobsledding using C3D.

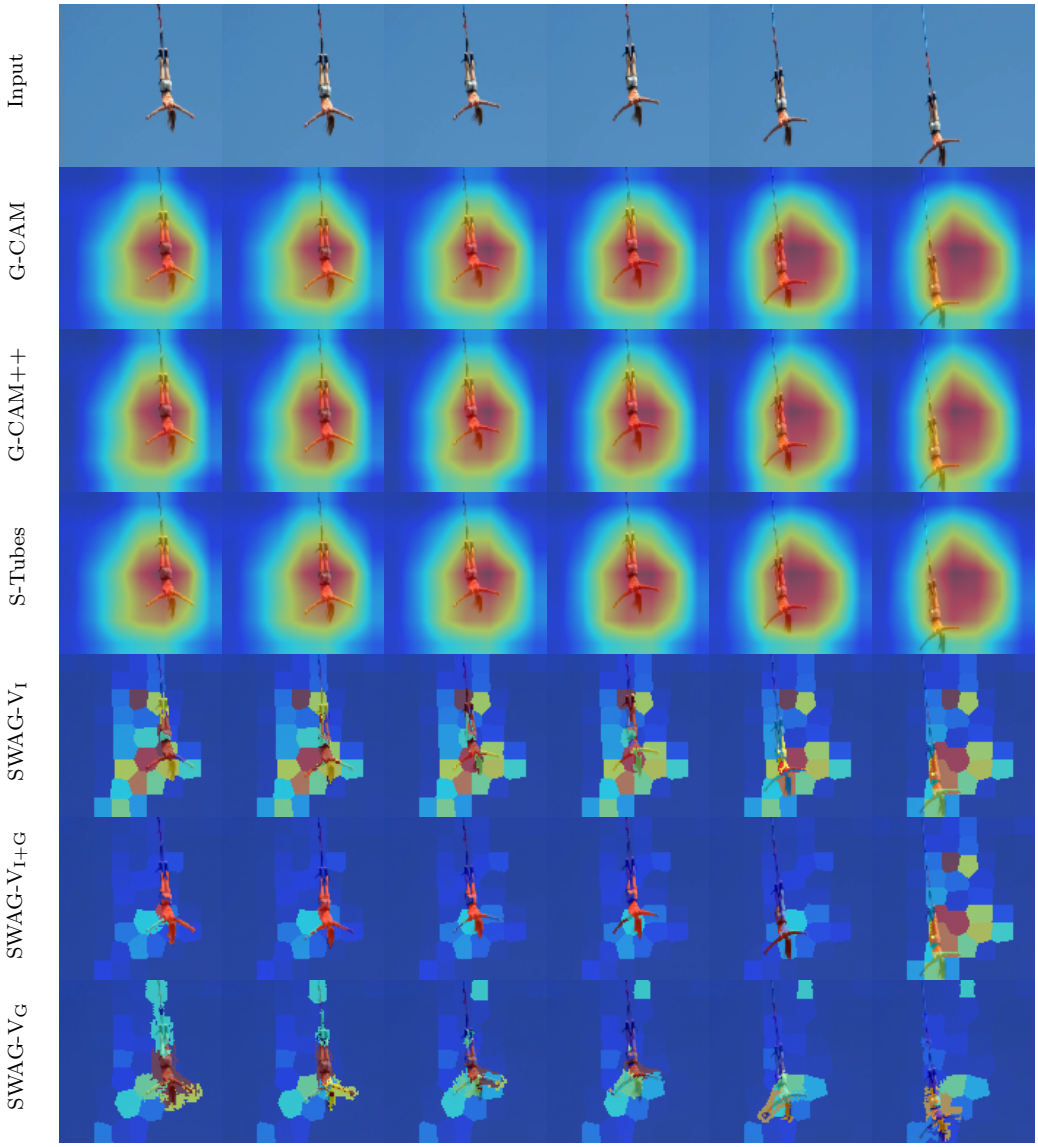

FIGURE B.5: An explanation for the Kinetics 400 class bungee jumping using R(2+1)D.

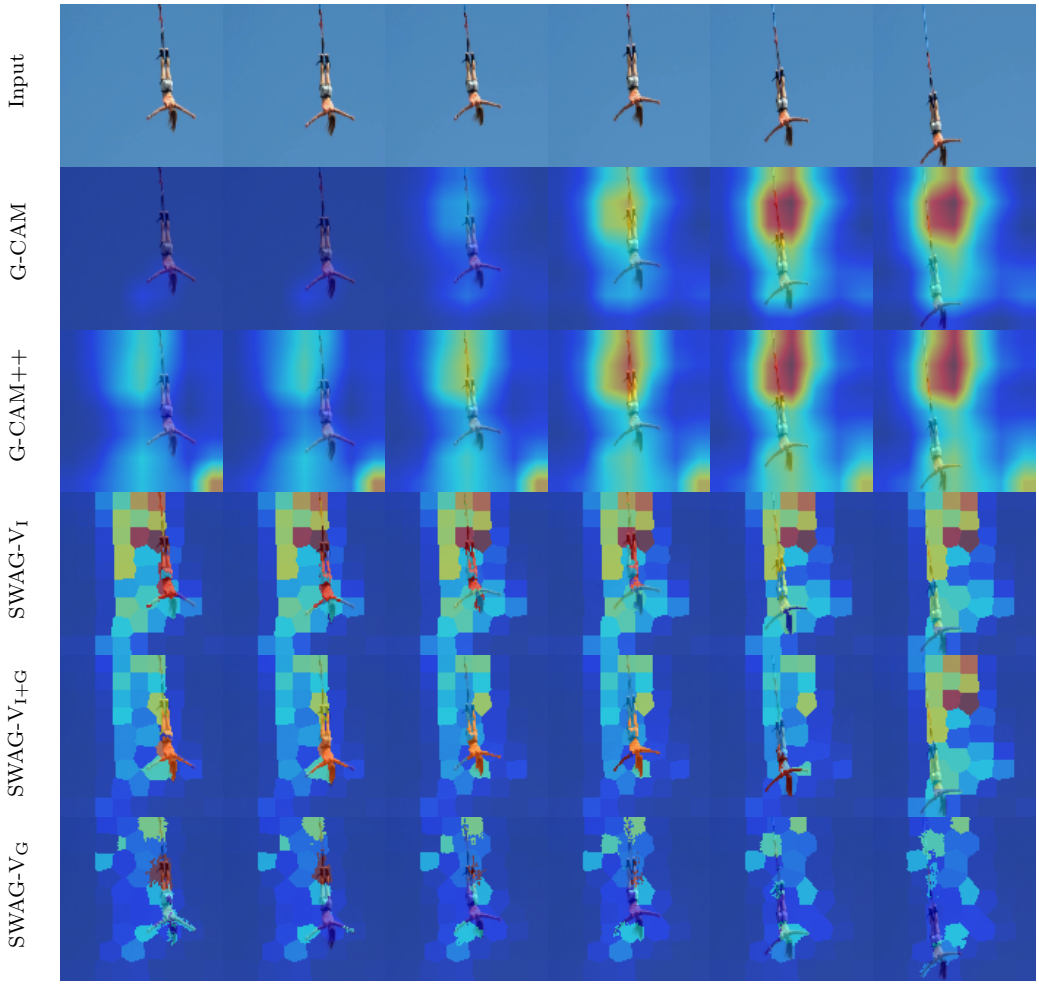

FIGURE B.6: An explanation for the Kinetics 400 class bungee jumping using C3D.

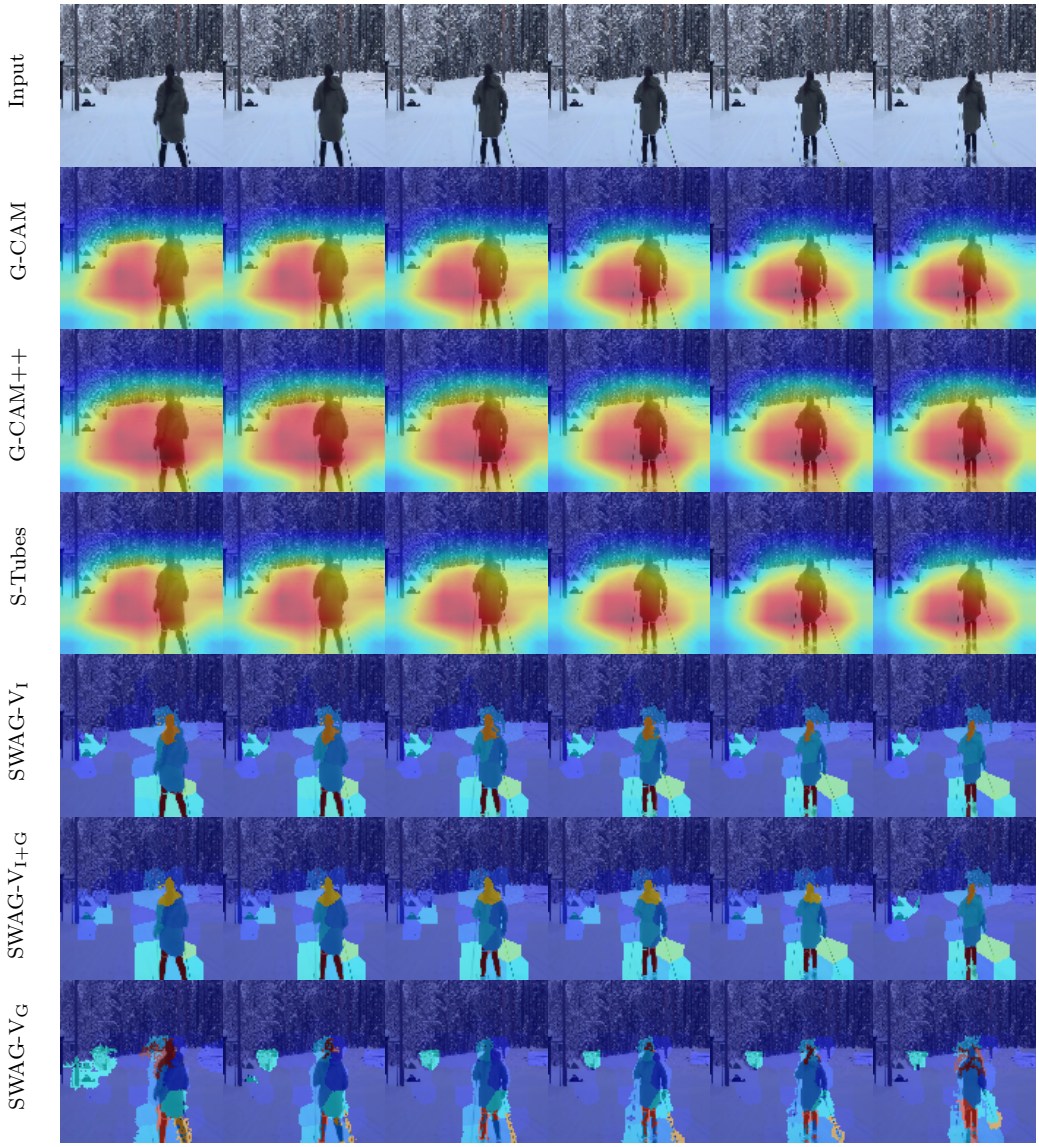

FIGURE B.7: An explanation for the Kinetics 400 class skiing crosscountry using R(2+1)D.

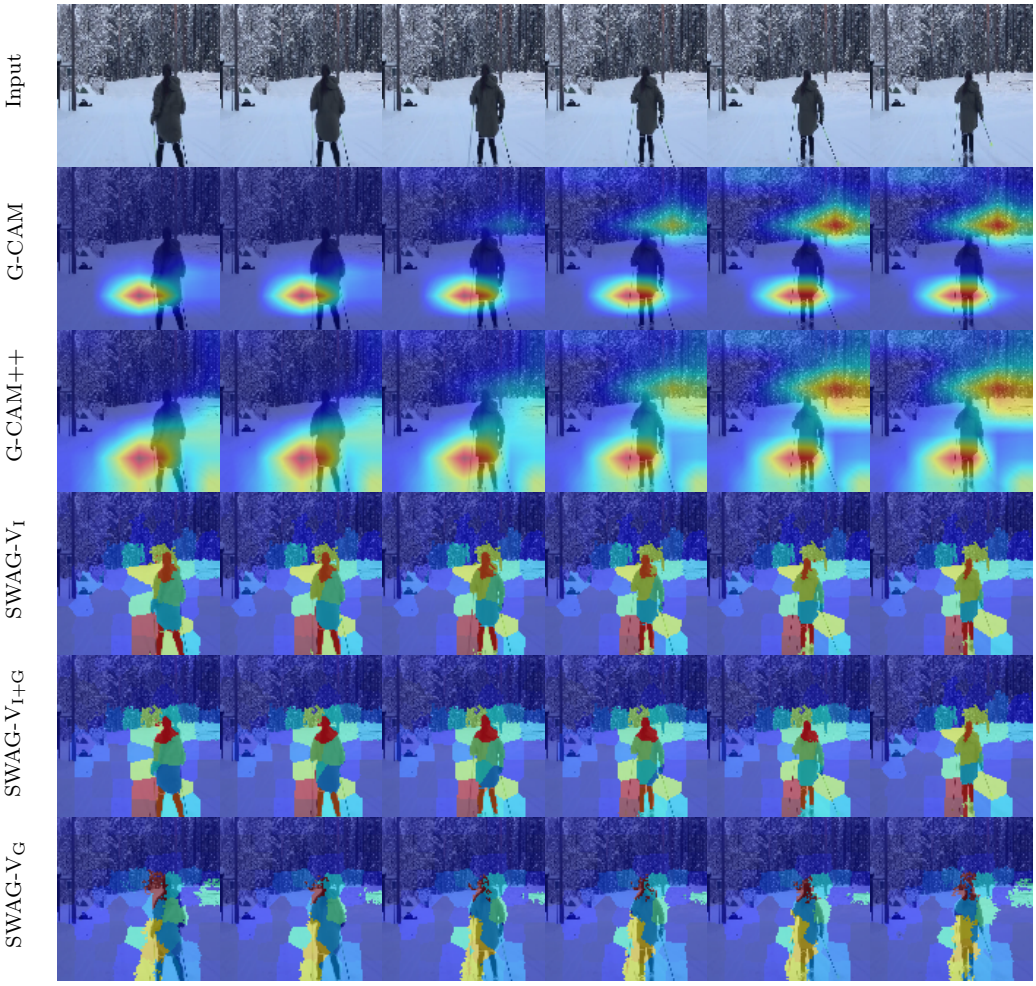

FIGURE B.8: An explanation for the Kinetics 400 class skiing crosscountry using C3D.

Appendix

C Additional Jitter-CAM Examples

Additional examples created for Chapter 5 using ImageNet.. The following models are used:

- Explanations created for ResNet50 are shown in Figure C.1.
- Explanations created for DenseNet121 are shown in Figure C.2.
- Explanations created for InceptionV3 are shown in Figure C.3.

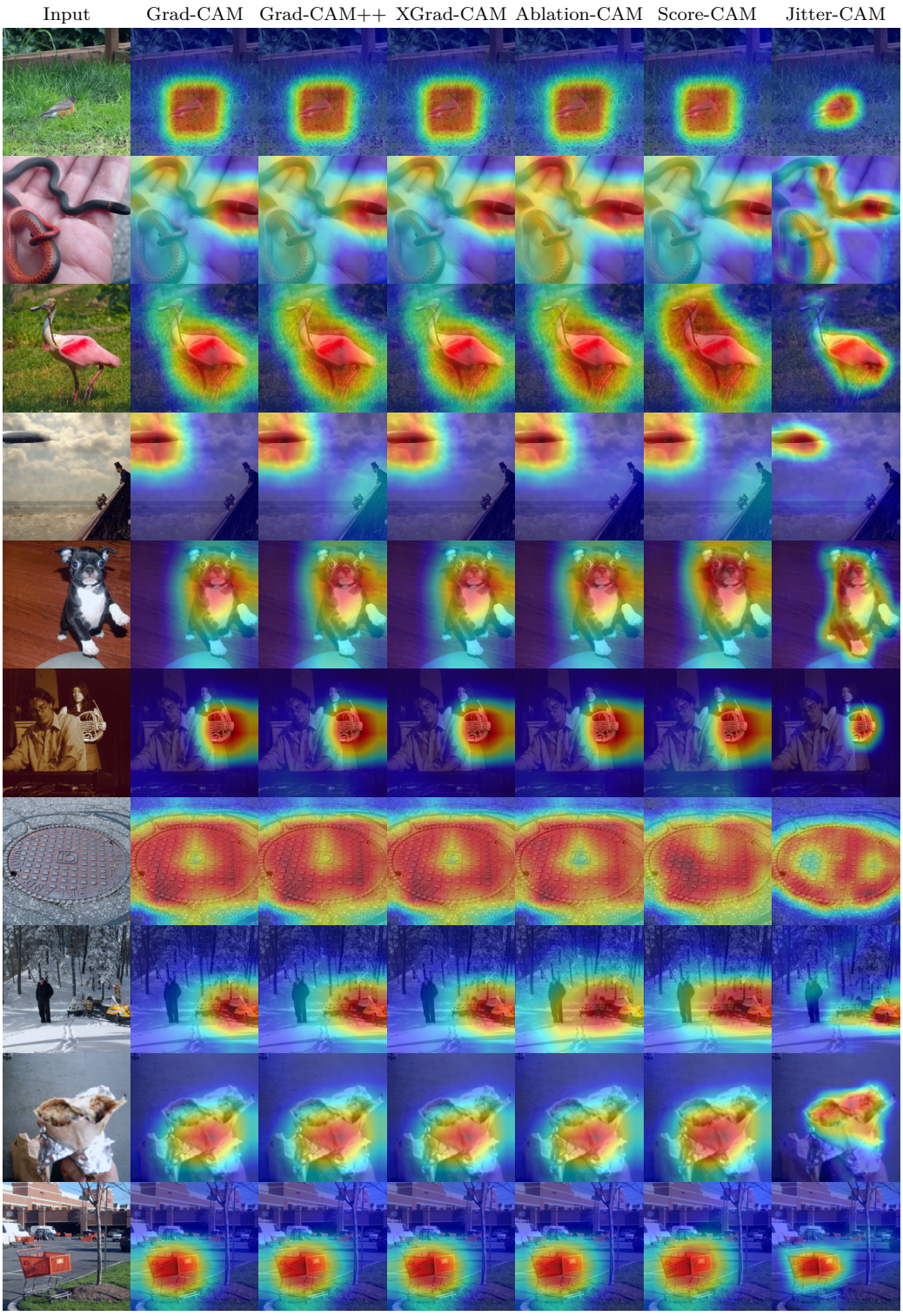

FIGURE C.1: Additional examples for ResNet50 using ImageNet.

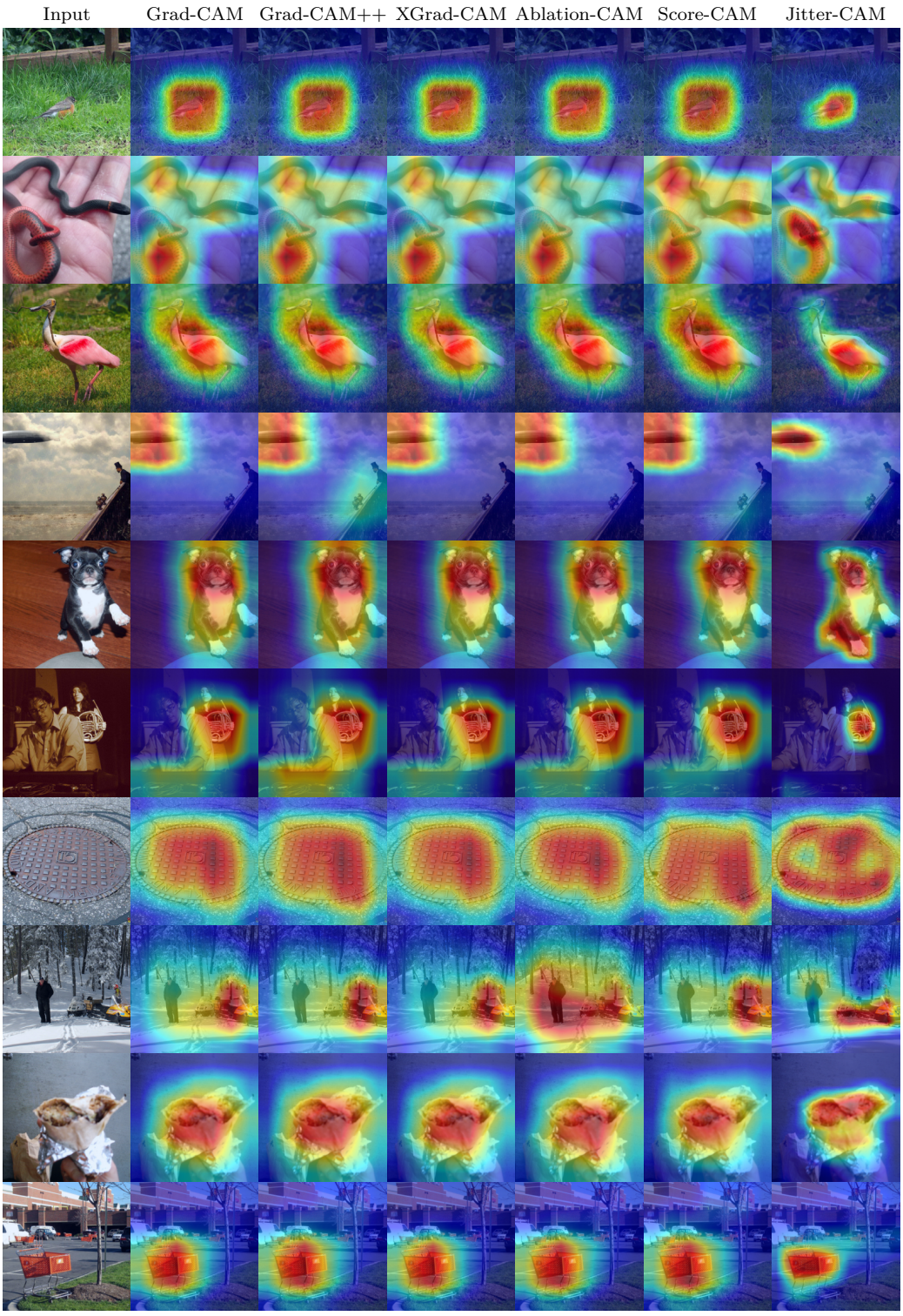

FIGURE C.2: Additional examples for DenseNet121 using ImageNet

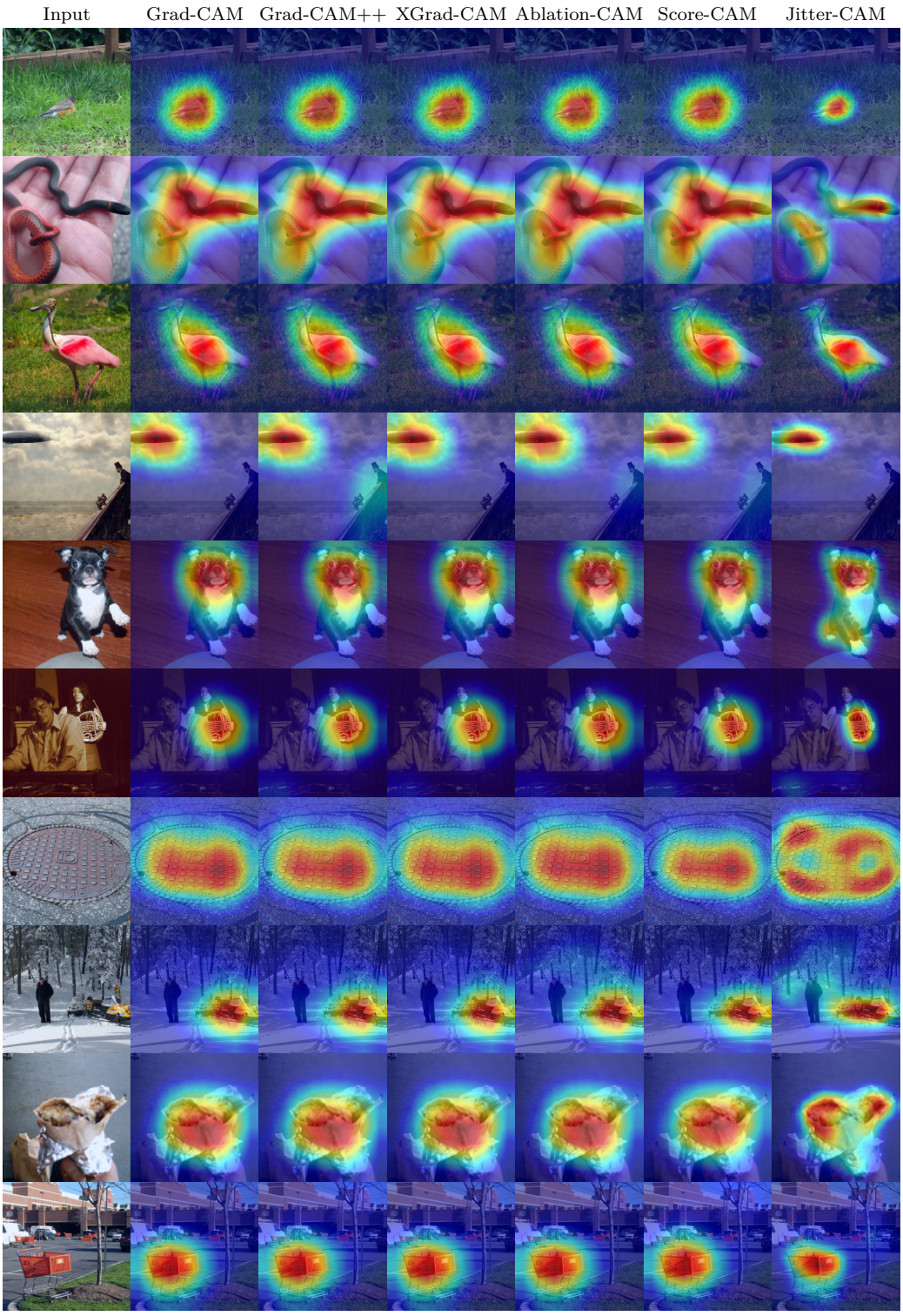

FIGURE C.3: Additional examples for InceptionV3 using ImageNet.