Monthly Notices
of the
ROYAL ASTRONOMICAL SOCIETY

MNRAS **493,** 161–170 (2020)
Advance Access publication 2020 January 20

doi:10.1093/mnras/staa122

Characterizing lognormal fractional-Brownian-motion density fields with a convolutional neural network

M. L. Bates,[1]★ A. P. Whitworth[1] and O. D. Lomax[1,2]

[1]*School of Physics and Astronomy, Cardiff University, Cardiff CF24 3AA, UK*
[2]*ESTEC, Keplerlaan 1, NL-2201 AZ Noordwijk, the Netherlands*

Accepted 2020 January 9. Received 2019 December 18; in original form 2019 May 28

ABSTRACT

In attempting to quantify statistically the density structure of the interstellar medium, astronomers have considered a variety of fractal models. Here, we argue that, to properly characterize a fractal model, one needs to define precisely the algorithm used to generate the density field, and to specify – at least – three parameters: one parameter constrains the spatial structure of the field, one parameter constrains the density contrast between structures on different scales, and one parameter constrains the dynamic range of spatial scales over which self-similarity is expected (either due to physical considerations, or due to the limitations of the observational or numerical technique generating the input data). A realistic fractal field must also be noisy and non-periodic. We illustrate this with the exponentiated fractional Brownian motion (xfBm) algorithm, which is popular because it delivers an approximately lognormal density field, and for which the three parameters are, respectively, the power spectrum exponent, β, the exponentiating factor, $\mathcal{S}$, and the dynamic range, $\mathcal{R}$. We then explore and compare two approaches that might be used to estimate these parameters: machine learning and the established Δ-Variance procedure. We show that for $2 \leq \beta \leq 4$ and $0 \leq \mathcal{S} \leq 3$, a suitably trained Convolutional Neural Network is able to estimate objectively both β (with root-mean-square error $\epsilon_\beta \sim 0.12$) and $\mathcal{S}$ (with $\epsilon_\mathcal{S} \sim 0.29$). Δ-variance is also able to estimate β, albeit with a somewhat larger error ($\epsilon_\beta \sim 0.17$) and with some human intervention, but is not able to estimate $\mathcal{S}$.

Key words: methods: data analysis – methods: statistical – stars: formation – ISM: clouds.

1 INTRODUCTION

The interstellar medium is chaotic, due to the non-linear nature of the processes involved in its evolution (supersonic non-ideal magnetohydrodynamics, self-gravity, radiation transport, non-LTE chemistry, and heat transfer, etc.). Consequently, the overall structure of the interstellar medium must be described using statistical metrics. Since in the interstellar medium there exist structures spanning a large dynamic range of spatial scales, and since there is an evidence for self-similarity across parts of this dynamic range, there have been several attempts to characterize the interstellar medium, and in particular star-forming clouds, with fractal or multifractal parameters (e.g. Beech 1987; Bazell & Desert 1988; Falgarone, Phillips & Walker 1991; Hetem & Lepine 1993; Stutzki et al. 1998; Bensch, Stutzki & Ossenkopf 2001; Chappell & Scalo 2001; Sánchez, Alfaro & Pérez 2005; Ossenkopf, Krips & Stutzki 2008a; Kauffmann et al. 2010; Schneider et al. 2013; Elia et al. 2014; Rathborne et al. 2015; Elia et al. 2018). Such characterizations can, in principle, allow one (a) to constrain the 3D structures and dynamics that underlie the observed 2D projections; (b) to evaluate whether two observed regions might be statistically similar, even if their detailed structures are quite different; and (c) to compare the results of numerical simulations with observations, and with one another.

A variety of fractal metrics has been deployed. Of these, the conceptually simplest are the perimeter–area dimension, $\mathcal{D}_{\rm PA}$ (e.g. Beech 1987; Bazell & Desert 1988; Falgarone et al. 1991; Hetem & Lepine 1993; Sánchez et al. 2005; Federrath, Klessen & Schmidt 2009; Rathborne et al. 2015), and the box-counting dimension, $\mathcal{D}_{\rm BC}$ (e.g. Sánchez et al. 2005; Federrath et al. 2009; Elia et al. 2018); $\mathcal{D}_{\rm PA}$ is usually preferred to $\mathcal{D}_{\rm BC}$ because it tends to give less noisy results. A second group of metrics derive from structure, or structure-like, functions (e.g. Sánchez et al. 2005; Federrath et al. 2009; Kritsuk, Lee & Norman 2013); we include in this group the Δ-Variance metric (Stutzki et al. 1998; Ossenkopf, Krips & Stutzki 2008b; Federrath et al. 2009), which is the metric used here to compare with our convolutional neural network (CNN) procedure. A third group of metrics involves evaluation of the mass–length scaling relation (e.g. Chappell & Scalo 2001; Sánchez et al. 2005; Federrath

★ E-mail: matthew.bates@astro.cf.ac.uk

et al. 2009; Kauffmann et al. 2010; Kritsuk et al. 2013; Beattie, Federrath & Klessen 2019a; Beattie et al. 2019b). A fourth group involves estimating the size and/or mass spectra (e.g. Elmegreen & Falgarone 1996), or the density spectrum (e.g. Federrath et al. 2009; Konstandin et al. 2016).

There are two commonly used procedures for calibrating these metrics, and they are quite distinct. One procedure is based on idealized models of fractals generated using recursive algorithms. Hetem & Lepine (1993) describe three possible recursive fractal models, but do not identify a preferred model. Sánchez et al. (2005) use a model proposed by Soneira & Peebles (1978), but have to adjust this model for high fractal dimensions. Stutzki et al. (1998), Elmegreen (2002), and Shadmehri & Elmegreen (2011) use models based on fractal Brownian motion (fBm) and exponentiated fractal Brownian motion (xfBm), and these are the models that we use here.

Recursive fractal models have the problem that they are numberless, in the sense that there is no obvious limit to the possibility of inventing plausible new ones. One important distinction between different recursive fractal models is that some of them deliver nested fractals (i.e. fractals in which the smaller denser structures tend to be embedded within the larger more diffuse structures) and some do not. We have chosen the xfBm model because it delivers a lognormal density distribution: lognormal column density in 2D, as here, and lognormal volume density in 3D. Lognormal volume density and column density fields are commonly observed or inferred in (relatively) low-density gas (e.g. Schneider et al. 2012, 2013; Kainulainen, Federrath & Henning 2014), and are usually attributed to compressible turbulence (e.g. Vazquez-Semadeni 1994; Federrath et al. 2010). However, xfBm does not yield a nested fractal; structures on different scales are positioned randomly with respect to one another. In a future paper, we will explore whether nested fractal models provide a better model of the interstellar medium.

Recursive fractal models for the interstellar medium require the specification of at least three parameters. One parameter reflects the relative frequency and spatial distribution of structures on different scales; here, this parameter is the power-law exponent, β, but it might equally be the fractal dimension, $\mathcal{D}$, or the Hurst parameter, $\mathcal{H}$. A second parameter reflects the way in which the density varies with physical scale; here this parameter is the exponentiating factor, $\mathcal{S}$ (as defined in Section 2.2) – but in other algorithms it is the Larson scaling exponent (i.e. $\mathrm{dln}\,[\rho]/\mathrm{dln}\,[L]$, where ρ is the density, and L is a generic length-scale). A third parameter reflects the dynamic range of spatial scales, $\mathcal{R}$, over which the model is applied. This dynamic range might be determined by physical considerations, as, for example, in the theory of turbulence, which spans an inertial range from the large scales on which turbulent energy is injected to the small scales on which it is dissipated (e.g. Frisch 1995; Federrath 2013). Alternatively, the dynamic range of spatial scales might simply be determined by the limitations of the observations (between the field of view and the resolution of the telescope), or the limitations of the numerical technique (between the size of the computational domain and the smallest cell or particle); this is the case here.

The other procedure for calibrating fractal metrics is quite different, and is based on simulations of turbulent – and usually non-self-gravitating – interstellar gas (e.g. Federrath et al. 2009; Kritsuk et al. 2013; Konstandin et al. 2016; Beattie et al. 2019a,b). The turbulence is maintained in isotropic statistical equilibrium with a random forcing term. Turbulent simulations are normally multifractal, first because the simulations have a limited dynamic range of spatial scales (and hence the turbulence has a limited inertial range), and secondly because the balance of solenoidal and compressive modes tends to depend on scale (with a shift from solenoidal to compressive modes as the turbulent energy cascades to smaller scales). The validity of this procedure depends on the fidelity of the simulations, on whether all the appropriate physics has been included, and on whether the real interstellar medium subscribes to isotropic statistical equilibrium.

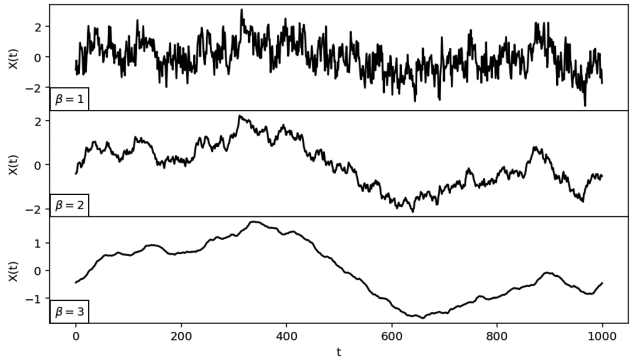


Figure 1. 1D ($\mathcal{E} = 1$) pure fBm curves for $\beta = 1.0,\ 2.0$ and 3.0.

Turbulent fractal simulations are usually characterized by just two parameters, the mean Mach Number, $\mathcal{M}$ of the turbulent velocity field, and the resolved dynamic range, $\mathcal{R}$. However, such simulations are also influenced by the way in which turbulent energy is continuously injected (e.g. the mix of solenoidal and compressive modes), the thermal and chemical behaviour of the gas, and the importance of self-gravity. Indeed, de Vega, Sánchez & Combes (1996) argue that fractal structure could be the natural product of self-gravity, rather than turbulence.

In all cases (both recursive fractal models and turbulent fractal simulations), different realizations of the same model (with the same model parameters) are obtained using different random number seeds.

The plan of this paper is as follows. In Section 2, we describe how 2D xfBm fields are constructed. In Section 3, we apply Δ-variance to the analysis of such fields. In Section 4, we train a CNN to analyse the same fields. In Section 5, we compare the two approaches and summarize our conclusions.

2 CONSTRUCTING XFBM FIELDS

xfBm fields are based on pure fBm fields, which we generate using the spectral synthesis method described by Peitgen & Saupe (1988). The same methods have been used by Stutzki et al. (1998), to create artificial molecular clouds, and by Lomax, Bates & Whitworth (2018), to create artificial star clusters.

Pure fBm fields (*un*-exponentiated fBm fields) are a generalised form of Brownian Motion and are characterized by a power-law spectrum with exponent $\beta = \mathcal{E} + 2\mathcal{H}$, where $\mathcal{E}$ is the Euclidean dimension and $\mathcal{H}$ is the Hurst parameter. Fig. 1 demonstrates how a 1D ($\mathcal{E} = 1$) pure fBm field, $X(t)$, depends on β. Each field has been realized with the same random seed, in order to preserve the general shape, but larger β means more power on larger scales, and hence a smoother field. Irrespective of the value of β, the mean of the field over a sufficiently long t-interval, μ_X, is normally much smaller in magnitude than its standard deviation, σ_X. The case $\beta = 2.0$ corresponds to a 1D random walk, which is also sometimes described as classical Brownian motion.

For the rest of this paper, we will work in two dimensions ($\mathcal{E} = 2$), and hence we will be considering surface density fields. However, the procedures we discuss can easily be adjusted to treat other Euclidean dimensions. The methodology we use to create xfBm fields comprises five distinct stages; Stages 2 through 4 can be implemented in any order, but Stage 1 is always implemented first, and Stage 5 is always implemented last.

2.1 Stage 1, generating a pure fBm field

A pure fBm field, $f_\beta(\boldsymbol{r})$, is constructed by first generating a power spectrum $\hat{f}_\beta(\boldsymbol{k})$. Here, $\boldsymbol{r} \equiv (r_1, r_2)$ is a 2D grid of integers with values of $1 \leq r_i \leq N_{\rm PIX}$, along each Cartesian axis, and $\boldsymbol{k} \equiv (k_1, k_2)$ is a 2D grid of wave-vectors with integer values of $-N_{\rm PIX}/2 \leq k_j \leq N_{\rm PIX}/2$ along each Cartesian axis. For each $\boldsymbol{k}$, the contribution to the power spectrum is given by

$$\hat{f}_\beta(\boldsymbol{k}) = A_\beta(\boldsymbol{k}) \left\{ \cos(\varphi\boldsymbol{k}) + i \sin(\varphi\boldsymbol{k}) \right\}; \tag{1}$$

$$A_\beta(\boldsymbol{k}) = \begin{cases} 0, & \text{if } \boldsymbol{k} = \boldsymbol{0}; \\ \mathcal{K}^{-1/2} ||\boldsymbol{k}||^{-\beta/2}, & \text{if } \boldsymbol{k} \neq \boldsymbol{0}; \end{cases} \tag{2}$$

$$\mathcal{K} = \sum_{\boldsymbol{k}} \left\{ ||\boldsymbol{k}||^{-\beta} \right\}; \tag{3}$$

$$\varphi(\boldsymbol{k}) = \chi(\boldsymbol{k}) - \chi(-\boldsymbol{k}); \tag{4}$$

$A_\beta(\boldsymbol{k})$ and $\varphi(\boldsymbol{k})$ are, respectively, the amplitude and phase of the contribution. The normalization factor $\mathcal{K}$ scales the total power of the field to unity. $\chi(\boldsymbol{k})$ is a random variate sampled from a uniform distribution on the interval $0 \leq \chi(\boldsymbol{k}) \leq 2\pi$. The pure fBm field, $f_\beta(\boldsymbol{r})$, is obtained by taking the inverse Fourier Transform of $\hat{f}_\beta(\boldsymbol{k})$.

2.2 Stage 2, the exponentiated fBm field

A pure fBm field has a roughly Gaussian distribution with a mean of around zero, $\langle f_\beta(\boldsymbol{r}) \rangle \approx 0$. This means that roughly half of the field has negative values. In order to make the field everywhere positive, so that it can be used to model surface density, we follow Elmegreen (2002) and exponentiate $f_\beta(\boldsymbol{r})$ to obtain an xfBm field,

$$g_{\mathcal{HS}}(\boldsymbol{r}) = \exp\left\{ \frac{\mathcal{S}\, f_\beta(\boldsymbol{r})}{\left\langle f_\beta^2(\boldsymbol{r}) \right\rangle^{1/2}} \right\}, \tag{5}$$

using a scaling parameter, $\mathcal{S}$. $\mathcal{S} = 0$ gives uniform density, and, as $\mathcal{S}$ is increased, the range of densities widens, and hence the structures become more sharply defined. This process transforms the roughly Gaussian field into one with a roughly lognormal distribution.

2.3 Stage 3, the non-periodic xfBm field

The xfBm field, $g_{\mathcal{HS}}(\boldsymbol{r})$, is periodic, but observed fields are not. Therefore, we initially generate a pure fBm field, $g'_{\mathcal{HS}}(\boldsymbol{r}')$, with $1 \leq r'_i \leq 4N_{\rm PIX}$, along each Cartesian axis. Then, we cut out an $N_{\rm PIX} \times N_{\rm PIX}$ section, located so that its geometric centre coincides with its centre of mass.

2.4 Stage 4, the noisy xfBm field

Since real observations are noisy, we add white noise to $g_{\mathcal{HS}}(\boldsymbol{r})$. The white noise field is scaled to be a fraction $\eta = 0.05\mathcal{B}$ of the standard deviation, σ_g, of $g_{\mathcal{HS}}(\boldsymbol{r})$, where $\mathcal{B}$ is a linear random deviate on the interval [0, 1]; hence the noise always lies between 0 per cent and 5 per cent of σ_g. This is only intended to be illustrative, but it is worth noting that higher noise levels will compromise Δ-Variance more than the CNN.

2.5 Stage 5, Adjusting rogue pixels

Finally, in order to filter out rogue pixels (for example, which in a real map might represent cosmic ray strikes), we compute the mean, μ_g, and standard deviation, σ_g, for all pixels. Any pixels with $g > \mu_g + 2.5\sigma_g$ are replaced with $\mu_g + 2.5\sigma_g$, and similarly any pixels with $g < \mu_g - 2.5\sigma_g$ are replaced with $\mu_g - 2.5\sigma_g$. This cull of the most extreme pixels helps to stabilize the training and implementation of the CNN. Moreover, in observed clouds, the lognormal part of the column density Probability Density Field (PDF) is seldom well defined outside $\pm 2.5\sigma_g$, due to incompleteness on the low side, and a power-law tail (usually attributed to self-gravity; Girichidis et al. 2014) on the high side (e.g. Schneider et al. 2012).

2.6 xfBm fields

Fig. 2 shows how the appearance of an xfBm field, generated by the procedure outlined in the preceding sections, depends on β and $\mathcal{S}$. These fields have all been generated from the same random seed in order that they all have the same large-scale pattern.

For the three fields on the top row, $\beta = 4.0$ (equivalently $\mathcal{H} = 1.0$), the power is strongly concentrated in long-wavelength modes, and there is little small-scale structure; the same contours could be overlaid on all three images, albeit at different column densities, and these contours would tend to be very smooth. For the three fields on the bottom row, $\beta = 2.0$ (equivalently $\mathcal{H} = 0.0$), there is a lot of power at short wavelengths, and hence lots of small-scale structure; again, the same contours could be overlaid on all three images on the bottom row, albeit at different column densities, and these contours would tend to be very twisted.

For the three fields in the left-hand column, $\mathcal{S} = 0.5$, the range of densities is the same and rather small. The only difference is that at the top ($\beta = 4.0$) the density peaks and troughs are quite extended, and at the bottom ($\beta = 2.0$) they are more compact. For the three fields in the right-hand column, $\mathcal{S} = 3.0$, the range of densities is also the same, but now it is rather big. Once again the density peaks and troughs at the top ($\beta = 4.0$) are quite extended, and those at the bottom ($\beta = 2.0$) are more compact. The range $0.5 \leq \mathcal{S} \leq 3.0$ is chosen because this covers the range of variances in the column density PDFs of observed clouds (e.g. Schneider et al. 2012, 2013; Kainulainen et al. 2014).

Stutzki et al. (1998) show that the corresponding box-counting fractal dimension should be $\mathcal{D}_{\rm BC} \simeq (3\mathcal{E} + 2 - \beta)/2$, and the corresponding perimeter–area fractal dimension should be $\mathcal{D}_{\rm PA} \simeq (3\mathcal{E} - \beta)/2 = \mathcal{D}_{\rm BC} - 1$, where $\mathcal{E}$ is the Euclidean dimension, and we have used '$\simeq$' because our xfBm fields are not pure. Substituting $\mathcal{E} = 2$, we obtain $\mathcal{D}_{\rm BC} \simeq (8 - \beta)/2$ and $\mathcal{D}_{\rm AP} \simeq (6 - \beta)/2$.

In Fig. 2, and in the rest of the paper, we use $N_{\rm PIX} = 128$, so the dynamic range of spatial scales is $\mathcal{R} \lesssim 64$. In the next two sections, we explore two techniques for characterizing xfBm fields constructed in this way: Δ-Variance (Section 3) and CNNs (Section 4).

3 Δ-VARIANCE

The Δ-variance, $\sigma^2_\Delta(L)$, of a 2D field, $g(x, y)$ is the variance after the field has been convolved with a circular filter function, $\odot_L$, characterized by length-scale L:

$$\sigma^2_\Delta(L) = \frac{1}{2\pi} \left\langle (g * \odot_L)^2 \right\rangle_{x,y}. \tag{6}$$

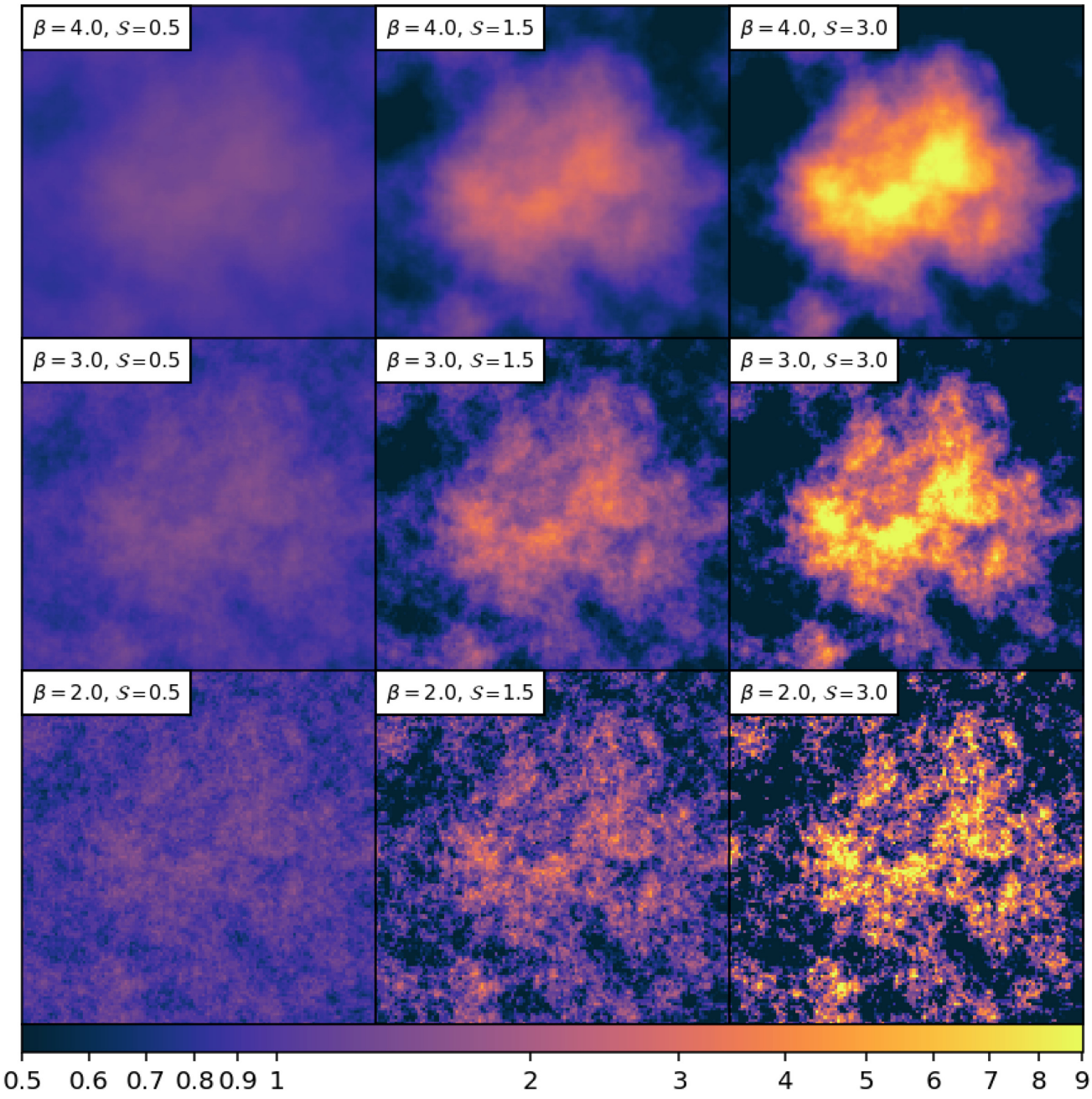


Figure 2. 2D fBm fields generated using the same random seed, but with different β and $\mathcal{S}$, as shown in the top left-hand corner of each panel. The fields are periodic and consist of 128 × 128 pixels. The logarithmic colour scale is the same in each plot, and shows the relative surface density, in arbitrary units.

$\sigma^2_\Delta(L)$ must be evaluated for many different values of L, spanning the full dynamic range of spatial scales being modelled. The power-law exponent, β, is then given by

$$\beta = \mathcal{E} + \frac{\mathrm{d}\ln\left(\sigma^2_\Delta\right)}{\mathrm{d}\ln(L)}. \tag{7}$$

In computing this gradient, care must be taken to discount end effects, i.e. where L is either close to the scale of the whole field, or close to the resolution limit; this issue is discussed further in Section 3.1 next.

In the original formulation (Stutzki et al. 1998), the French Hat filter function has been used, but Ossenkopf et al. (2008b) show that better results are obtained with the Mexican Hat filter function:

$$\odot_L(r) = \odot_{\mathrm{CORE}.L}(r) - \odot_{\mathrm{ANN}.L}(r), \tag{8}$$

$$\odot_{\mathrm{CORE}.L}(r) = \frac{4}{\pi L^2}\exp\left(\frac{-4r^2}{L^2}\right), \tag{9}$$

$$\odot_{\mathrm{ANN}.L}(r) = \frac{4}{\pi(\nu^2-1)L^2}\left\{\exp\left(\frac{-4r^2}{\nu^2 L^2}\right) - \exp\left(\frac{-4r^2}{L^2}\right)\right\}, \tag{10}$$

and this is the filter that we use here.

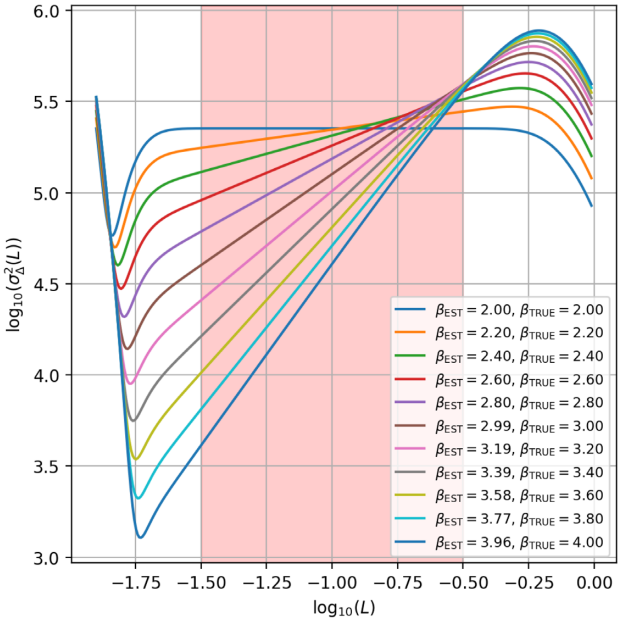


Figure 3. Δ-variance curves for pure (i.e. periodic, un-exponentiated, and noiseless) fBm fields with power-law exponent $\beta_{\rm TRUE} =$ 2.0, 2.2, 2.4, 2.6, 2.8, 3.0, 3.2, 3.4, 3.6, 3.8, and 4.0 (hence $\mathcal{H}_{\rm TRUE} =$ 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, and 1.0). The pink shading shows the range used to estimate the slope, and hence (using equation 7) to obtain $\beta_{\rm EST}$. Values of $\beta_{\rm TRUE}$ and $\beta_{\rm EST}$ are tabulated in the corner of the frame.

3.1 The power-law exponent, β, for periodic fields

As noted by Ossenkopf et al. (2008b), for periodic fields (but only for periodic fields), the Δ-variance can be computed more quickly by integrating the product of the power spectrum of g (denoted $\mathcal{P}_g(\boldsymbol{k})$) and the power spectrum of the filter function (denoted $\tilde{\odot}_L$) over $\boldsymbol{k}$-space:

$$\sigma^2_\Delta(L) = \frac{1}{2\pi}\int \mathcal{P}_g(\boldsymbol{k})\,\left|\tilde{\odot}_L\right|^2\,{\rm d}^2\boldsymbol{k}. \tag{11}$$

Fig. 3 shows the Δ-variance curves obtained in this way for 11 pure (i.e. periodic, un-exponentiated, and noiseless) fBm fields with $\beta_{\rm TRUE} =$ 2.0, 2.2, 2.4, 2.6, 2.8, 3.0, 3.2, 3.4, 3.6, 3.8, and 4.0. If we limit consideration to the range $-1.50 \le \log_{10}(L) \le -0.50$ (the shaded pink on Fig. 3), the slope is in all cases well defined, and can be used to estimate β from equation (7). The values estimated in this way, $\beta_{\rm EST}$, are tabulated in the corner of Fig. 3, and agree well with the input values, $\beta_{\rm TRUE}$.

3.2 The power-law exponent, β, for non-periodic fields

Ossenkopf et al. (2008b) also note that for non-periodic fields, a more convoluted procedure is required. First, the map is zero-padded to twice the linear size. Next, the convolution is performed, using the original filter size, but only on the pixels that constitute the original map, in order to prevent the filter from wrapping around the edges of the map. This involves four convolution integrals:

$$G_{{\rm CORE}.L}(\boldsymbol{r}) = g_{\rm PAD}(\boldsymbol{r}') * \odot_{{\rm CORE}.L}(\boldsymbol{r}), \tag{12}$$

$$G_{{\rm ANN}.L}(\boldsymbol{r}) = g_{\rm PAD}(\boldsymbol{r}') * \odot_{{\rm ANN}.L}(\boldsymbol{r}), \tag{13}$$

$$W_{{\rm CORE}.L}(\boldsymbol{r}) = w(\boldsymbol{r}') * \odot_{{\rm CORE}.L}(\boldsymbol{r}), \tag{14}$$

$$W_{{\rm ANN}.L}(\boldsymbol{r}) = w(\boldsymbol{r}') * \odot_{{\rm ANN}.L}(\boldsymbol{r}), \tag{15}$$

where $g_{\rm PAD}(\boldsymbol{r}')$ is the zero-padded map, and $w(r')$ is a normalization map that takes values of 1 within the region of the original map, and 0 in the zero-padded region. The fully convolved map is then computed using

$$F_L(\boldsymbol{r}) = \frac{G_{{\rm CORE}.L}(\boldsymbol{r})}{W_{{\rm CORE}.L}(\boldsymbol{r})} - \frac{G_{{\rm ANN}.L}(\boldsymbol{r})}{W_{{\rm ANN}.L}(\boldsymbol{r})}, \tag{16}$$

and the Δ-variance is given by

$$\sigma^2_\Delta(L) = \frac{\sum\left\{(F_L(\boldsymbol{r}) - \langle F_L(\boldsymbol{r})\rangle)^2\, W_{{\rm TOT}.L}(\boldsymbol{r})\right\}}{\sum\left\{W_{{\rm TOT}.L}(\boldsymbol{r})\right\}}. \tag{17}$$

Here, $W_{{\rm TOT}.L}(\boldsymbol{r}) = W_{{\rm CORE}.L}(\boldsymbol{r})\, W_{{\rm ANN}.L}(\boldsymbol{r})$ acts as a map of weights, which, when applied to the variance calculation, gives less significance to the pixels that are most heavily distorted by edge effects due to the zero-padding.

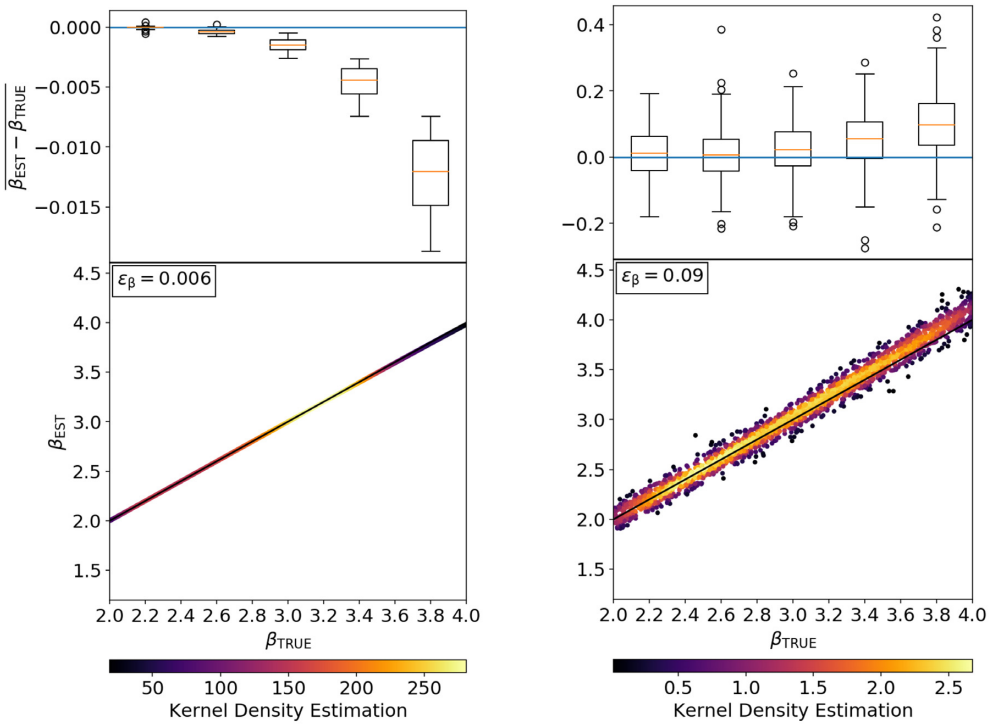
3.3 Evaluating the performance of Δ-variance

To test the above procedures, we use the methodology outlined in Section 2 to construct 2000 different artificial xfBm fields, each measuring 128 × 128 pixels, and each with a random value of $\beta_{\rm TRUE}$ on the interval [2.0, 4.0], and a random value of $\mathcal{S}_{\rm TRUE}$ on the interval [0, 3]. At each stage in the construction, we apply Δ-variance to estimate $\beta_{\rm EST}$, and compare the result with $\beta_{\rm TRUE}$. The results are presented in Fig. 4. Note that for this exercise we have reversed the order of Stages 2 and 3 (Sections 2.2 and 2.3).

For the pure fBm fields generated in Stage 1 (Section 2.1), we are able to use the procedure for periodic fields outlined in Section 3.1, and the same range ($-1.50 \le \log_{10}(L) \le -0.50$). The results are presented in Fig. 4(a). In this case, there is almost exact correspondence between $\beta_{\rm EST}$ and $\beta_{\rm TRUE}$. The root-mean-square error is $\epsilon_\beta \simeq 0.006$.

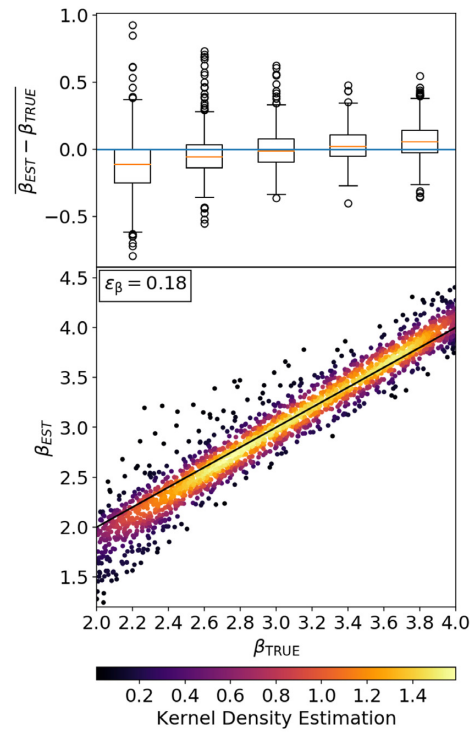
For the non-periodic fields generated in the subsequent stages (Sections 2.2–2.4), we have to use the more convoluted procedure for treating non-periodic fields, as outlined in Section 3.2, and consequently the estimates of the power-law exponent deteriorate. Fig. 4(b) shows the results obtained with non-periodic fBm fields; in this case $\epsilon_\beta \simeq 0.09$, and there is a tendency to overestimate $\beta_{\rm EST}$ for high values of $\beta_{\rm TRUE}$. Fig. 4(c) shows the results obtained for exponentiated non-periodic fields; in this case $\epsilon_\beta \simeq 0.18$, and there is still a tendency to overestimate $\beta_{\rm EST}$ for high values of $\beta_{\rm TRUE}$, but also a tendency to underestimate $\beta_{\rm EST}$ for low values of $\beta_{\rm TRUE}$. The addition of noise does not change the error significantly, i.e. it is $\epsilon_\beta \simeq 0.17$.

In order to explore the interplay between the parameters β and $\mathcal{S}$ and how this is reflected in the values of $\beta_{\rm EST}$ obtained using Δ-variance, we consider discrete values of $\beta_{\rm TRUE} =$ 2.0, 2.2, 2.4, 2.6, 2.8, 3.0, 3.2, 3.4, 3.6, 3.8, and 4.0 and $\mathcal{S}_{\rm TRUE} =$ 0.5, 1.0, 2.0, and 3.0. Then, for each combination of $\beta_{\rm TRUE}$ and $\mathcal{S}_{\rm TRUE}$, we generate 400 different artificial xfBm fields (i.e. non-periodic, exponentiated, and noisy); estimate their individual $\beta_{\rm EST}$ using Δ-variance; and hence determine the mean, $\mu_\beta = \langle\beta_{\rm EST}\rangle$, and standard deviation, $\sigma_\beta = \langle(\beta_{\rm EST} - \langle\beta_{\rm EST}\rangle)^2\rangle^{1/2}$. Fig. 5 displays the results. In general, as $\mathcal{S}_{\rm TRUE}$ increases, the mean, μ_β, falls increasingly far below $\beta_{\rm TRUE}$, and the standard deviation, σ_β, increases. These trends are particularly strong for low values of $\beta_{\rm TRUE}$. Inspection of Fig. 2 suggests that these trends arise because increasing $\mathcal{S}$ and reducing β both have the effect of amplifying the

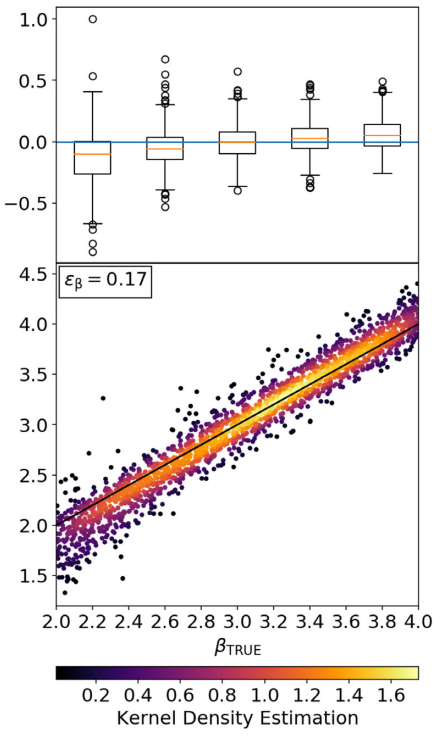


(a) Pure

(b) Non-Periodic



(c) Exponentiated



(d) Noisy

Figure 4. Comparison of the input values of the power-law exponent, $\beta_{\rm TRUE}$, with the values estimated using Δ-variance, $\beta_{\rm EST}$. The box-and-whisker plots in the top row show the distribution of $(\beta_{\rm EST} - \beta_{\rm TRUE})$ in bins of width $\Delta\beta_{\rm TRUE} = 0.4$. In each bin, the orange line marks the median, and the box spans from the lower quartile, Q_1, to the upper quartile, Q_3. If the interquartile range is $\Delta Q = Q_3 - Q_1$, the upper whisker extends to the highest point less than $Q_3 + 1.5\Delta Q$, the lower whisker extends to the lowest point greater than $Q_1 - 1.5\Delta Q$, and all points outside this range are plotted individually as the open circles. The blue line marks exact correspondence. The kernel density estimates on the bottom row show the correspondence between $\beta_{\rm EST}$ and $\beta_{\rm TRUE}$, and ϵ_β is given, for each stage, in the top lefthand corner of the panel. Reading from left to right and top to bottom, the plots correspond to (a) *pure* fBm fields, (b) *non-periodic* fBm fields, (c) *exponentiated*, non-periodic fBm fields, and (d) *noisy*, exponentiated, non-periodic fBm fields.

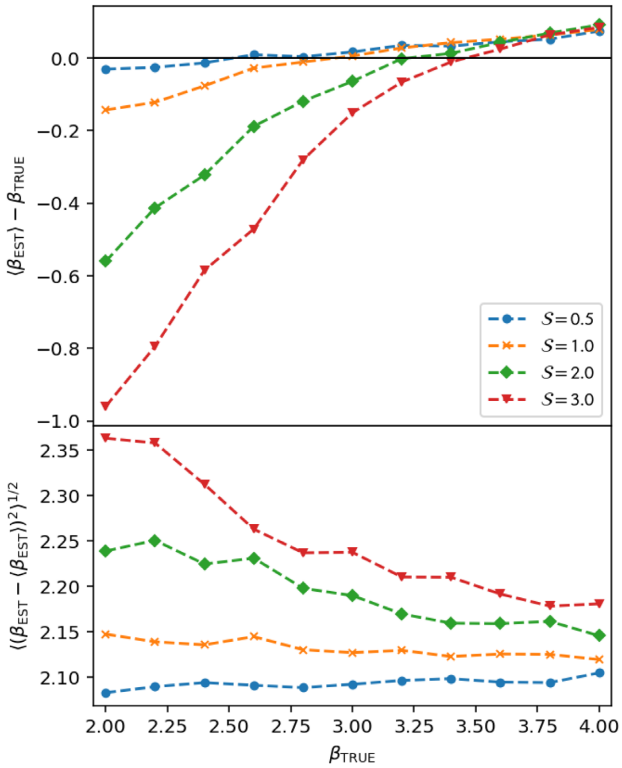


Figure 5. Values of $\langle\beta_{EST}\rangle - \beta_{TRUE}$ (top panel) and $\langle(\beta_{EST} - \langle\beta_{EST}\rangle)^2\rangle^{1/2}$ (bottom panel) for discrete values of $\beta_{TRUE} =$ 2.0, 2.2, 2.4, 2.6, 2.8, 3.0, 3.2, 3.4, 3.6, 3.8, and 4.0, and discrete values of $\mathcal{S}_{TRUE} = 0.5$ (the filled blue circles), 1.0 (the orange crosses), 2.0 (the filled green diamonds), and 3.0 (the filled red triangles). For each combination of β_{TRUE} and $\mathcal{S}_{TRUE}$, β_{EST} has been estimated using Δ-variance. The mean and standard deviation of β_{EST} are based on 400 different artificial xfBm fields (i.e. non-periodic, exponentiated, and noisy fields).

visibility of small-scale structure in the field. Δ-variance is unable to distinguish these two effects, as noted previously by Lomax et al. (2018).

4 CONVOLUTIONAL NEURAL NETWORKS

The use of neural networks for classification and regression has expanded rapidly in recent years. A large variety of different types of network has emerged, most notably the CNN, which is used extensively in problems involving image recognition. A notable example is handwritten digit recognition (Ciresan et al. 2011; Ciresan, Meier & Schmidhuber 2012). Several competitions have also served to push the boundaries of CNNs, for instance, the annual ImageNet Large Scale Visual Recognition Challenge, which in 2012 established the usefulness of Graphic Processing Units when combined with deep CNNs (Krizhevsky, Sutskever & Hinton 2017).

More recently machine learning techniques have started to be applied to problems in astronomy. Examples of the use of CNNs include galaxy classification (Khalifa et al. 2017), gamma-ray astronomy (Dieleman, Willett & Dambre 2015; Postnikov et al. 2018), supernova classification (Kimura et al. 2017), astronomical image reconstruction (Flamary 2016), denoising of images (Remez et al. 2017), and star cluster analysis (Bialopetravicius, Narbutis & Vansevicius 2019).

Table 1. The architecture of the CNN. The initial 128 × 128 × 1 input layer is the 2D field to be analysed, and the final 1 × 1 × 2 output layer gives the estimated β and $\mathcal{S}$. In between, there are five convolutional layers, each followed by a max pooling function, and then five flattened, fully connected, dense layers. The output size column follows the format: `width × height × channels`. The total number of parameters is 11 545 090.

Layer	Output size	Operation
Input	128 × 128 × 1	Input layer
Conv.1	126 × 126 × 512	3 × 3 kernel
MaxPool.1	63 × 63 × 512	2 × 2 max pooling
Conv.2	61 × 61 × 512	3 × 3 kernel
MaxPool.2	30 × 30 × 512	2 × 2 max pooling
Conv.3	28 × 28 × 512	3 × 3 kernel
MaxPool.3	14 × 14 × 512	2 × 2 max pooling
Conv.4	12 × 12 × 512	3 × 3 kernel
MaxPool.4	6 × 6 × 512	2 × 2 max pooling
Conv.5	4 × 4 × 512	3 × 3 kernel
MaxPool.5	2 × 2 × 512	2 × 2 max pooling
Flatten	1 × 1 × 2048	Flattens into 1D layer
Dense.1	1 × 1 × 512	Fully connected
Dense.2	1 × 1 × 512	Fully connected
Dense.3	1 × 1 × 512	Fully connected
Dense.4	1 × 1 × 512	Fully connected
Dense.5	1 × 1 × 512	Fully connected
Output	1 × 1 × 2	One channel each for β and $\mathcal{S}$

4.1 Architecture of the CNN

A CNN consists of a collection of artificial neurons, with each neuron taking a vector of inputs $\boldsymbol{x}$, and producing a scalar output, $y = f(c + \boldsymbol{w} \cdot \boldsymbol{x})$. Here, $\boldsymbol{w}$ is a vector of weights, c is a bias, and $f(\cdot)$ is an activation function; the activation function used here is the rectified linear unit, $f(x) = \text{MAX}[0, x]$ (Nair & Hinton 2010).

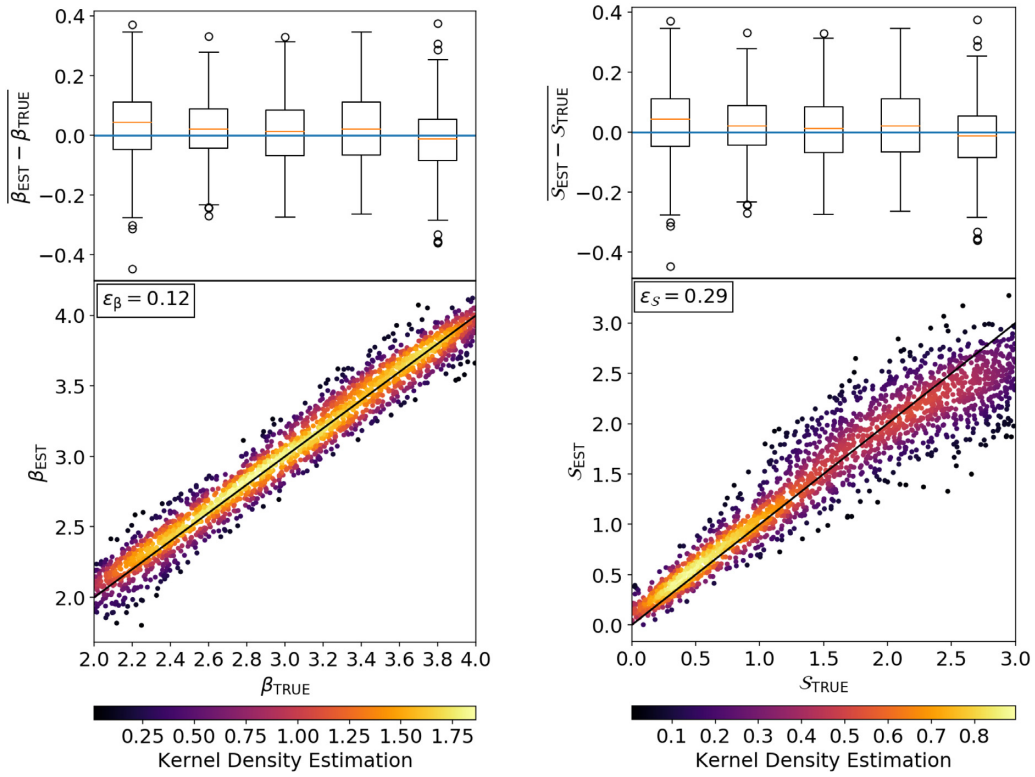
Neurons are arranged in multiple groupings called layers, and each neuron in the layer takes all the outputs from the previous layer as its inputs. In general, a layer delivers a vector of outputs $\boldsymbol{y}$; and a sequence of layers forms a neural network. Table 1 shows the structure of the CNN developed here, using the `Tensorflow` package. It consists of five convolutional layers (Conv.N), each followed by a max pooling layer (MaxPool.N). These are then flattened into a 1D layer that is then followed by five fully connected layers (Dense.N).

The weights and biases of the neurons comprise the parameters of the network, and are refined using multiple sets of input data ($\boldsymbol{x}_{INPUT}$) and their corresponding known statistical parameters ($\boldsymbol{y}_{KNOWN}$). Gradient descent is then used to minimize a loss function $\mathcal{L}$, which we set to the mean square error:

$$\mathcal{L} = \langle(F(\boldsymbol{x}_{INPUT}) - \boldsymbol{y}_{KNOWN})^2\rangle. \quad (18)$$

Here, $F(\boldsymbol{x}_{INPUT})$ is the estimate of $\boldsymbol{y}$ delivered by the CNN.

The convolutional layers of the CNN consist of 2D grids of multiple, learnable convolutional filters. Each filter comprises a 3 × 3 window, made up of nine parameters. The window is moved across the map in steps, producing an output at each step by computing the dot product between the filter and the local subsection of the map. The CNN used here has five convolutional layers, each using 512 different filters (so that it produces 512 different feature maps) and a step of 1 (so that it reduces the size of the layer by 2 in each dimension). The nine parameters for each filter are refined by minimization of the loss function.



(a) β (b) $\mathcal{S}$

Figure 6. *Left-hand panels:* comparison of the input values of the power-law exponent, β_{TRUE} with the values returned by the CNN, β_{EST}. *Right-hand panels:* comparison of the input values of the scaling factor, $\mathcal{S}_{TRUE}$ with the values returned by the CNN, $\mathcal{S}_{EST}$. The box-and-whisker plots on the top row show the distribution of $(\beta_{EST}-\beta_{TRUE})$ in bins of width $\Delta\beta_{TRUE} = 0.4$, and $(\mathcal{S}_{EST} - \mathcal{S}_{TRUE})$ in bins of width $\Delta\mathcal{S}_{TRUE} = 0.6$. In each bin, the orange line marks the median, and the box spans from the lower quartile, Q_1, to the upper quartile, Q_3. If the interquartile range is $\Delta Q = Q_3 - Q_1$, the upper whisker extends to the highest point less than $Q_3 + 1.5\Delta Q$, the lower whisker extends to the lowest point greater than $Q_1 - 1.5\Delta Q$, and all points outside this range are plotted as the open circles. The blue line marks exact correspondence. The kernel density estimates on the bottom row show the correspondence between β_{EST} and β_{TRUE}, and $\mathcal{S}_{EST}$ and $\mathcal{S}_{TRUE}$; the values of ϵ_β and $\epsilon_\mathcal{S}$ are given in the top left-hand corner of each panel.

Each convolutional layer is followed by a max pooling layer, using a 2×2 window and a step of 2. Max pooling outputs the maximum value of a 2×2 subsection of the layer. The step of 2 means the window moves 2 pixel before outputting the next maximum, thereby halving the image size.

The input to the CNN is a single channel, 128×128 pixel xfBm field. The first convolutional layer (Conv.1) produces 512 different feature maps, and these are then carried through the network, until they are condensed into 512 single neurons at the Dense.1 layer, and finally into two singular neurons at the output layer, i.e. the values of β_{EST} and $\mathcal{S}_{EST}$.

4.2 Training the CNN

To train the CNN, we generate 20 000 artificial xfBm fields (using the procedures described in Section 2), each with a random value of β on the interval [2.0, 4.0], a random value of $\mathcal{S}$ on the interval [0, 3], and 128×128 pixels. The CNN's parameters start out with random values. The artificial xfBm fields are then input to the network in batches of 32, the input β_{TRUE}, and $\mathcal{S}_{TRUE}$ are compared with the values estimated by the network, β_{EST} and $\mathcal{S}_{EST}$, and the parameters updated using the `RMSprop` gradient-descent optimizer, so as to minimize the loss function, $\mathcal{L}$. For a comprehensive review of different optimizers and their applicability, see Ruder (2016). We train the CNN for 100 epochs with a random 70–30 train-test cross-validation split. For details of this cross-validation split, see Appendix A.

4.3 Evaluating the performance of the CNN

We test the performance of the CNN using the same 2000 artificial fBm fields that were used in Section 3.3 to test the performance of Δ-variance. Fig. 6(a) shows that the CNN tends to overestimate the power-law exponent, β, but the error is small, $\epsilon_\beta = 0.12$. Fig. 6(b) shows that the CNN also tends to overestimate the scaling factor, $\mathcal{S}$, except for large values ($\mathcal{S} > 2$), which it tends to underestimate; the error is $\epsilon_\mathcal{S} = 0.29$.

5 DISCUSSION AND CONCLUSIONS

It appears that the CNN developed here is able to estimate the power-law exponent, β, of an xfBm field (i.e. an fBm field that has been exponentiated, and is non-periodic and noisy) more accurately (rms error $\epsilon_\beta = 0.12$) than Δ-variance ($\epsilon_\beta = 0.18$). In addition, the CNN can also evaluate the scaling factor ($\mathcal{S}$) with reasonable accuracy ($\epsilon_\mathcal{S} = 0.29$).

Training and cross-validating a CNN takes about 4 h on a GPU cluster, but applying the CNN to a single, 128×128 pixel xfBm field then takes $\lesssim 0.1$ s. In contrast, Δ-variance requires no training, but applying it to a single, 128×128 pixel xfBm field takes ~ 2 s (on

the same computer architecture), because it entails the computation of several convolution integrals over the whole field. It may also require human intervention to identify the range over which the plot of $\log_{10}(\sigma^2_\Delta(L))$ against $\log_{10}(L)$ is linear.

The CNN developed here can only be applied to 128×128 pixel fields. Given a field with $N_{\rm PIX} \neq 128$, we have three choices. (i) We can convert the field to 128×128 pixels. (ii) If $N_{\rm PIX} > 128$, we can divide the field up into 128×128 pixel subfields, analyse each subfield separately, and combine the results with appropriate weights. (iii) We can develop a new CNN. In contrast, Δ-variance can be applied immediately to a field with any number of pixels.

The disadvantage of both approaches is that they return parameter values irrespective of whether the fields being analysed are actually well approximated by fBm. This is particularly true for the CNN, which is a black box with no demonstrable relation to underlying physical structures. Δ-variance can at least provide some (necessary but not sufficient) evidence for an underlying fBm structure, if the plot of $\log_{10}(\sigma^2_\Delta(L))$ against $\log_{10}(L)$ displays a linear portion (as demonstrated for the pure fBm fields analysed in Fig. 3), but this may require human intervention. It might therefore be appropriate to combine the two approaches: use the CNN to estimate $\beta_{\rm CNN}$ and $\mathcal{S}_{\rm CNN}$, and then re-estimate $\beta_{\Delta\text{-VAR}}$ using Δ-variance, and check whether it falls below $\beta_{\rm CNN}$ in accordance with the results of Fig. 5.

ACKNOWLEDGEMENTS

MLB gratefully acknowledges the receipt of a PhD studentship from the UK Science and Technology Facilities Council (STFC) through the Centre for Doctoral Training (CDT) in Data Intensive Science (ST/P006779/1). APW and ODL gratefully acknowledge the support of an STFC Consolidated Grant (ST/K00926/1). ODL also gratefully acknowledges the support of an ESA Fellowship. This work was performed using the computational facilities of the Advanced Research Computing at Cardiff Division, Cardiff University. We thank the referee for their careful report on the original version of this paper, which we found very helpful.

REFERENCES

Bazell D., Desert F. X., 1988, ApJ, 333, 353
Beattie J. R., Federrath C., Klessen R. S., 2019a, MNRAS, 487, 2070
Beattie J. R., Federrath C., Klessen R. S., Schneider N., 2019b, MNRAS, 488, 2493
Beech M., 1987, Ap&SS, 133, 193
Bensch F., Stutzki J., Ossenkopf V., 2001, A&A, 366, 636
Bialopetravičius J., Narbutis D., Vansevičius V., 2019, A&A, 621, A103
Chappell D., Scalo J., 2001, ApJ, 551, 712
Cireşan D. C., Meier U., Masci J., Gambardella L. M., Schmidhuber J., 2011, in International Joint Conference on Artificial Intelligence, p. 1237
Ciresan D., Meier U., Schmidhuber J., 2012, in 2012 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, Providence, RI, p. 3642
de Vega H. J., Sánchez N., Combes F., 1996, Nature, 383, 56
Dieleman S., Willett K. W., Dambre J., 2015, MNRAS, 450, 1441
Elia D. et al., 2014, ApJ, 788, 3
Elia D. et al., 2018, MNRAS, 481, 509
Elmegreen B. G., 2002, ApJ, 564, 773
Elmegreen B. G., Falgarone E., 1996, ApJ, 471, 816
Falgarone E., Phillips T. G., Walker C. K., 1991, ApJ, 378, 186
Federrath C., 2013, MNRAS, 436, 1245
Federrath C., Klessen R. S., Schmidt W., 2009, ApJ, 692, 364
Federrath C., Roman-Duval J., Klessen R. S., Schmidt W., Mac Low M.-M., 2010, A&A, 512, A81
Flamary R., 2016, eprint (arXiv:1612.04526)
Frisch U., 1995, Turbulence. The Legacy of A.N. Kolmogorov. Cambridge Univ. Press, Cambridge
Girichidis P., Konstandin L., Whitworth A. P., Klessen R. S., 2014, ApJ, 781, 91
Hetem A., Jr, Lepine J. R. D., 1993, A&A, 270, 451
Kainulainen J., Federrath C., Henning T., 2014, Science, 344, 183
Kauffmann J., Pillai T., Shetty R., Myers P. C., Goodman A. A., 2010, ApJ, 716, 433
Khalifa N. E. M., Taha M. H. N., Hassanien A. E., Selim I. M., 2017, pr eprint (arXiv:1709.02245 [cs])
Kimura A., Takahashi I., Tanaka M., Yasuda N., Ueda N., Yoshida N., 2017, 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW), p. 354
Konstandin L., Schmidt W., Girichidis P., Peters T., Shetty R., Klessen R. S., 2016, MNRAS, 460, 4483
Kritsuk A. G., Lee C. T., Norman M. L., 2013, MNRAS, 436, 3247
Krizhevsky A., Sutskever I., Hinton G. E., 2017, Commun. ACM, 60, 84
Lomax O., Bates M. L., Whitworth A. P., 2018, MNRAS, 480, 371
Nair V., Hinton G. E., 2010, in Proceedings of the 27th International Conference on International Conference on Machine Learning, Omnipress, Madison, WI, p. 807
Ossenkopf V., Krips M., Stutzki J., 2008a, A&A, 485, 719
Ossenkopf V., Krips M., Stutzki J., 2008b, A&A, 485, 917
Peitgen H.-O., Saupe D., 1988, The Science of Fractal Images. Springer, New York
Postnikov E. B. et al., 2018, preprint (arXiv:1812.01551)
Rathborne J. M. et al., 2015, ApJ, 802, 125
Remez T., Litany O., Giryes R., Bronstein A. M., 2017, preprint (arXiv:1701.01687)
Ruder S., 2016, preprint (arXiv:1609.04747)
Sánchez N., Alfaro E. J., Pérez E., 2005, ApJ, 625, 849
Schneider N. et al., 2012, A&A, 540, L11
Schneider N. et al., 2013, ApJ, 766, L17
Shadmehri M., Elmegreen B. G., 2011, MNRAS, 410, 788
Soneira R. M., Peebles P. J. E., 1978, AJ, 83, 845
Stutzki J., Bensch F., Heithausen A., Ossenkopf V., Zielinsky M., 1998, A&A, 336, 697
Vazquez-Semadeni E., 1994, ApJ, 423, 681

APPENDIX A: OPTIMIZATION

The CNN was initially trained for 500 full passes, or epochs, of the input data set (the 20 000 artificial fBm fields). At each epoch, a random 70 per cent of the artificial fields (i.e. 14 000 fields) were selected and used to train the network, by minimizing the associated loss function, $\mathcal{L}_{\rm TRAIN.70\ per\ cent}$. The remaining 30 per cent (6000 fields) were set aside and used to cross-validate the network, by computing its loss function, $\mathcal{L}_{\rm VALID.30\ per\ cent}$, separately. This cross-validation is designed to check that the network is not overfitting the data set. If it is, $\mathcal{L}_{\rm VALID.30\ per\ cent}$ will tend to increase systematically with successive passes, while $\mathcal{L}_{\rm TRAIN.70\ per\ cent}$ will generally continue to decrease. We train for a large number of epochs (500) in order to determine the point at which the CNN starts to overfit. Fig. A1 shows the evolution of the TRAIN. 70 PER CENT and VALID. 30 PER CENT loss functions. Separate plots are given for the contributions to the loss functions from β and $\mathcal{S}$, and for their sum. We see by eye that $\mathcal{L}_{\rm VALID.30\ per\ cent}$ starts to increase at $\sim$100 epochs. Therefore, we restrict the CNN to 100 epochs for the analyses described in Section 4.

After $\sim$400 epochs, $\mathcal{L}_{\rm TRAIN.70\ per\ cent}$ also starts to increase. This suggests that the gradient-descent optimizer (here, `RMSProp`) is taking too large a step and thus moving away from the loss-function minimum.

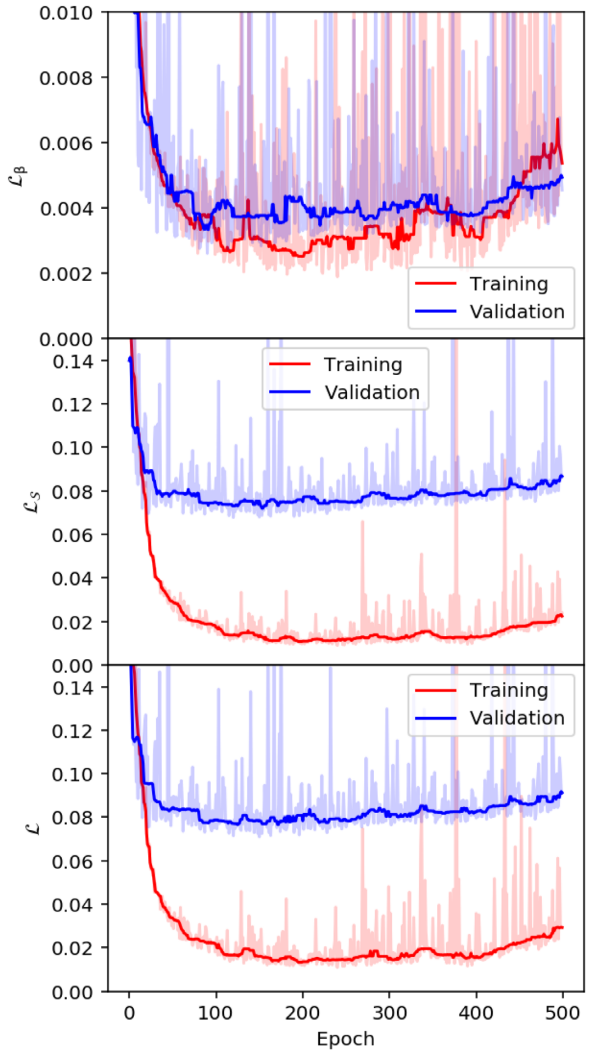


Figure A1. The loss function for β ($\mathcal{L}_\beta$, top panel), $\mathcal{S}$ ($\mathcal{L}_\mathcal{S}$, middle panel), and the total ($\mathcal{L} = \mathcal{L}_\beta + \mathcal{L}_\mathcal{S}$, bottom panel). The pale red (blue) curves show how the loss function for the training (testing) set evolves with epoch, and the dark red (blue) curves are smoothed versions obtained by taking the median of the 20 surrounding points.

We tested the dependence on image size by repeating the analyses in Sections 3 and 4 with 100 × 100 pixel xfBm images. Using a CNN, there was no significant change in the accuracy, with $\epsilon_\beta = 0.12$ and $\epsilon_\mathcal{S} = 0.31$. Using Δ-variance, the accuracy was somewhat worse, with $\epsilon_\beta = 0.18$.

We also tested several distinct CNN architectures, and different numbers of layers and different numbers of nodes. The architecture described in the text (Table 1) appears to deliver reasonable accuracy using modest computation time.

This paper has been typeset from a TEX/LATEX file prepared by the author.