Monthly Notices
of the
ROYAL ASTRONOMICAL SOCIETY

MNRAS **504,** 2603–2613 (2021)
Advance Access publication 2021 April 12

doi:10.1093/mnras/stab1011

A generative model of galactic dust emission using variational autoencoders

Ben Thorne ,★ Lloyd Knox and Karthik Prabhu
Department of Physics, University of California, One Shields Avenue, Davis, CA 95616, USA

Accepted 2021 April 8. Received 2021 April 8; in original form 2021 January 27

ABSTRACT

Emission from the interstellar medium can be a significant contaminant of measurements of the intensity and polarization of the cosmic microwave background (CMB). For planning CMB observations, and for optimizing foreground-cleaning algorithms, a description of the statistical properties of such emission can be helpful. Here, we examine a machine learning approach to inferring the statistical properties of dust from observational data. In particular, we apply a type of neural network called a variational autoencoder (VAE) to maps of the intensity of emission from interstellar dust as inferred from *Planck* sky maps and demonstrate its ability to (i) simulate new samples with similar summary statistics as the training set, (ii) provide fits to emission maps withheld from the training set, and (iii) produce constrained realizations. We find VAEs are easier to train than another popular architecture: that of generative adversarial networks, and are better suited for use in Bayesian inference.

Key words: methods: statistical – ISM: general – cosmic background radiation.

1 INTRODUCTION

Among the many research enterprises stimulated by the detection of large-scale anisotropies in the cosmic microwave background (CMB) by the COsmic Background Explorer (*COBE*) with its differential microwave radiometer (Smoot et al. 1992), is the hunt for signatures of primordial gravitational waves (PGWs). To date, only upper limits have been set, most commonly expressed as limits on the ratio of primordial tensor perturbation power to scalar perturbation power, r. Soon after the *COBE* detection, it was realized that reliably detecting levels below $r \sim 0.1$ could not be done with temperature anisotropies alone (Knox & Turner 1994), and that proceeding further would require highly sensitive measurements of the polarization of the CMB on angular scales of about a degree, or larger (Kamionkowski, Kosowsky & Stebbins 1997; Seljak & Zaldarriaga 1997).

Polarized emission from the interstellar medium (ISM) of the Milky Way, in the cleanest parts of the sky at the cleanest observing frequencies, is comparable to the CMB signal generated by PGWs, if the PGW signal is near the current 95 per cent confidence upper limit of $r < 0.06$ (BICEP2 Collaboration 2018). So-called Stage III CMB experiments, such as the Simons Observatory (Ade et al. 2019), and BICEP Array (Hui et al. 2018) combined with SPT-3G (Benson et al. 2014) are designed to have sufficient sensitivity and systematic error control to tighten the 95 per cent confidence upper limits by a factor of about 20. The Stage IV experiments LiteBIRD and CMB-S4 are targeting upper limit factors of 2 and 5 times more stringent still, respectively. Thus, we are rapidly moving into a regime where the foreground contamination is up to two orders of magnitude larger [1] than the signal of interest.

The most exciting possibility is that there will be a detection of PGWs, as opposed to improved upper limits. A detection claim would essentially be a claim that there is power remaining in the map that cannot be explained as a residual instrumental systematic or residual foreground emission. Detection therefore requires not only foreground cleaning, but the capability to quantify the probability distribution of residual foreground power. Such capability is hampered by our lack of prior knowledge of the probability distribution of the non-Gaussian and non-isotropic Galactic foreground emission.

The state of the art in analysis of such observations either implicitly or explicitly has the Galactic emission, or their residuals, modelled as Gaussian isotropic fields (BICEP2 Collaboration 2018; Aiola et al. 2020; Planck Collaboration VI 2020). They are modelled as such not because they are, but strictly for convenience. The most appropriate underlying statistical description of the Galactic emission is unknown. Progress has been made in simulating observations of synthetic galaxies with properties similar to ours using magnetohydrodynamic codes (Kim, Choi & Flauger 2019), however, it is unclear how best to apply these simulations to simulate or analyse our particular galaxy. Other recent efforts have focused on defining new summary statistics that go beyond the assumption of Gaussianity, and which can be used to generate novel samples, or de-noise observations (Regaldo-Saint Blancard et al. 2021). Other work has utilized machine learning techniques to extrapolate the non-Gaussian structure from high signal-to-noise observations of dust intensity to inform the noisier small-scale structure of dust polarization (Krachmalnicoff & Puglisi 2021), however, this method is entirely deterministic.

In this paper, we propose the use of a variational autoencoder (VAE) to learn the underlying distribution of dust images, and use this as a prior on the spatial distribution of interstellar dust emission. This distribution could then be used in any downstream Bayesian analysis task. For example, sampling from such a prior to create maps of Galactic emission with the appropriate statistical

★ E-mail: bn.thorne@gmail.com

[1] This is for fluctuation power. The rms level of contamination in the map is up to one order of magnitude larger than the signal of interest.

properties for testing analysis algorithms to be used on real data, or in reconstructing incomplete observations of foreground images by conditioning the prior on the partial data. An ambitious goal would be to perform a complete Bayesian analysis of the CMB observations with incorporation of this prior for the spatial distribution of interstellar emission. Groundbreaking progress towards such a Bayesian analysis has been made recently, with the development of analysis methodologies by Millea, Anderes & Wandelt (2020a), and the recent application to real data (Millea et al. 2020b).

The analysis framework in Millea et al. (2020a) was developed for 'de-lensing' of the CMB; i.e. taking into account the impact of gravitational lensing on the statistical properties of CMB polarization. Although it has not been applied to multifrequency data, or used for foreground cleaning, at a conceptual level the framework can be straightforwardly extended to analysis of foreground-contaminated multifrequency data. Although this extension could be implemented with isotropic Gaussian priors for foreground emission, it also presents the opportunity to incorporate more realistic priors – priors that more accurately reflect what we know about such emission from other data, or from physics-based simulations.

Here, we report on progress towards accomplishing these tasks with the use of neural networks. Aylor et al. (2021) studied the use of generative adversarial networks (GANs) for learning how to simulate new emission maps with statistical properties similar to those from a training set. Here, we present a similar study, this time using a different neural network architecture and training program, that of VAEs.

VAEs and GANs are examples of deep generative models. These models have had recent success in accurately modelling complicated, high-dimensional, data sets, and generating realistic novel samples (van den Oord et al. 2016b; Brock, Donahue & Simonyan 2018; Razavi, van den Oord & Vinyals 2019). Generative models can be divided into two main categories: likelihood-based models that seek to optimize the log likelihood of the data, these include the VAE (Kingma & Welling 2013; Jimenez Rezende, Mohamed & Wierstra 2014), *flow* based methods (Dinh, Krueger & Bengio 2014; Jimenez Rezende & Mohamed 2015; Dinh, Sohl-Dickstein & Bengio 2016; Kingma & Dhariwal 2018), and *autoregressive* models (van den Oord, Kalchbrenner & Kavukcuoglu 2016a); and implicit models, such as GANs (Goodfellow et al. 2014), which train a generator and discriminator in an adversarial game scenario. There are many trade-offs to consider when selecting a likelihood-based approach (Kingma & Dhariwal 2018), but here we choose to explore the use of VAEs due to their explicit likelihood interpretation compared to GANs, and their computational efficieny on high-dimensional data sets compared to normalizing flows.

We find some advantages of VAEs over GANs trained in previous works. The adversarial training process does not produce an explicit inference model, and it is hard to consistently compare model performance against some test set. Furthermore, it is also a common problem that samples from GANs do not represent the full diversity of the underlying distribution (Grover, Dhar & Ermon 2017). In contrast, VAEs optimize the log likelihood of the data. This means both that it is possible to directly compare models, and trained models should support the entire data set, which is crucial when applying a trained model to real data. VAEs also tend to be easier to train in that training success is more stable to variation of hyperparameters. As a downside, VAEs are well known for loss of resolution. We see this in our results and discuss adaptations one could make to avoid this degradation of angular resolution.

Although our work is motivated by the PGW-driven desire to understand the statistical properties of polarized foreground emission, in this paper, as was the case in Aylor et al. (2021), we restrict ourselves to intensity. Observations of polarized dust emission with high signal-to-noise ratio over a large fraction of sky do not currently exist, which precludes the training of similar models on real data. However, in ongoing work, we are exploring the use of magnetohydrodynamical (MHD; Kim et al. 2019) simulations to train generative models of polarized emission. In this scenario, a trained model would provide a 'compression' of the information available in MHD simulations into a single statistical model, which could then be used either in inference or to augment real low-resolution observations with physically motivated small-scale realizations.

The rest of this paper is structured as follows. In Section 2, we introduce VAEs, and the objective for their optimization. We then describe the network architecture we used, the training data set we produced to train the network, and how hyperparameter values were set. In Section 3, we present the results of applying the trained VAE to test set images. Finally, in Section 4 we summarize our findings and discuss areas of current and future work.

2 VARIATIONAL AUTOENCODERS

In this section, we will introduce the idea of VAEs, the specific model we implement, and the details of how we train that model. This derivation closely follows the literature on this subject (e.g. Kingma et al. 2016; Kingma & Welling 2019), but we reproduce it here for pedagogical purposes.

Our goal here is to take a set of images of thermal emission from interstellar dust $\mathbf{x}^{(i)} = (x_1^{(i)}, \ldots, x_N^{(i)}) \in \mathbf{R}^N$, and infer from them an underlying distribution, $p(x)$ from which they could have been drawn, using the techniques of *generative modelling*. VAEs are a type of generative machine learning model, which provide a framework by which we may infer the parameters of a joint distribution over our original data, and some *latent variables*, $\mathbf{z}$, representing the unobserved part of the model. We can factorize the joint distribution of the data and latent variables into two terms representing the generative process of the data, and the latent space, responsible for the variance in the observed data:

$$p(\mathbf{x}, \mathbf{z}) = \underbrace{p(\mathbf{x}|\mathbf{z})}_{\text{Generative}} \; \underbrace{p(\mathbf{z})}_{\text{Variance}}. \tag{1}$$

The VAE approach is to model the conditional distribution with an appropriate family of functions with some unknown weights, θ: $p_\theta(\mathbf{x}|\mathbf{z}) \approx p(\mathbf{x}|\mathbf{z})$. This conditional model encodes the generative process by which $\mathbf{x}$ depends on the latent set of variables $\mathbf{z}$. The choice of $p(\mathbf{z})$ can then be a simple, perhaps Gaussian, prior probability distribution $p(\mathbf{z})$, which encodes the data set variation in a simple latent space. This can be seen as a type of regularization by which we separate out different sources of variation within the data set, a process that is quite natural for physical processes, and often makes the resulting model interpretable.

The goal of training is thus to find a transformation that delivers an acceptable approximation $p_\theta(\mathbf{x}) \approx p(\mathbf{x})$, that is optimal (in some sense), given the training set data. Towards that end, we consider the parametrized joint distribution of $\mathbf{x}$ and $\mathbf{z}$:

$$p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z}), \tag{2}$$

which leads to our object of interest via marginalization over $\mathbf{z}$:

$$p_\theta(\mathbf{x}) = \int d\mathbf{z}\, p_\theta(\mathbf{x}, \mathbf{z}). \tag{3}$$

Our tasks are thus to choose a parametrization – this is referred to as a choice of *architecture* – and then find a means of optimizing

these parameters (θ) with resepect to a chosen *objective*, via a process referred to as *training*.

2.1 Objective

In principle, we could determine θ by maximizing the training set's joint likelihood $\prod_i p_\theta(\mathbf{x}^i)$. In practice, however, this would involve evaluating the integral in equation (3) for each data point individually, which is intractable for even moderately high-dimensional latent spaces. The VAE framework provides an objective function that bounds the maximum likelihood value, and is computationally tractable.

Let a data set $\mathbb{D}$ be made up of samples $\mathbf{x}^{(i)} = (x_1^{(i)}, \ldots, x_N^{(i)}) \in \mathbf{R}^N$, which we will assume to be independent and identically distributed samples from some true underlying distribution $p_D(\mathbf{x})$. Absent an analytical model for $p_D(\mathbf{x})$, we can instead take it to be a member of an expressive family of functions parametrized by $\boldsymbol{\theta}$: $p_D(\mathbf{x}) = p_{\boldsymbol{\theta}}(\mathbf{x})$. This can be done by introducing an unobserved set of latent variables, $\mathbf{z} = (z_1, \ldots, z_d) \in \mathbf{R}^d$, and considering the joint distribution $p(\mathbf{x}, \mathbf{z})$. This joint distribution is specified by: the prior over the latent space, $p(\mathbf{z})$, which is assumed to be some simple distribution (typically Gaussian), and the conditional distribution $p(\mathbf{x}|\mathbf{z})$, which is intended to represent most of the complexity in the true underlying distribution $p_D(\mathbf{x})$. We model this distribution as a neural network with weights θ: $p_\theta(\mathbf{x}|\mathbf{z})$. The marginal likelihood is then:

$$p_\theta(\mathbf{x}) = \int \mathrm{d}\mathbf{z}\, p(\mathbf{z}) p_\theta(\mathbf{x}|\mathbf{z}) = \mathbf{E}_{p(\mathbf{z})}[p_\theta(\mathbf{x}|\mathbf{z})], \tag{4}$$

where we have introduced the notation $\mathbf{E}_Y[h(y)]$ to indicate the expectation of the function $h(y)$ with respect to the distribution $y \sim Y$. In principle, we could determine the conditional model by fixing θ to a value that maximizes the marginal likelihood. In practice, however, the integral in equation (4) is intractable, due to the dimensionality of the latent space, and in any case would require a per-data point optimization process. As a result, the posterior $p_\theta(\mathbf{z}|\mathbf{x}) = p_\theta(\mathbf{z}, \mathbf{x}) / p_\theta(\mathbf{x})$ is also intractable.

We make progress by introducing a second approximation, this time to the posterior: $q_\varphi(\mathbf{z}|\mathbf{x}) \approx p_\theta(\mathbf{z}|\mathbf{x})$, where $q_\varphi(\mathbf{z}|\mathbf{x})$ is often referred to as an *inference* network. For any choice of $q_\varphi(\mathbf{z}|\mathbf{x})$, including any choice of its weights φ, we can write the log likelihood of the data as

$$\log p_\theta(\mathbf{x}) = \mathbf{E}_{q_\varphi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x})]. \tag{5}$$

Applying the chain rule of probability: $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{z}) p_\theta(\mathbf{x}|\mathbf{z})$, and inserting an identity, this can be split into two terms:

$$\log p_\theta(\mathbf{x}) = \mathbf{L}_{\theta,\varphi}(\mathbf{x}) + \mathbf{D}_{\mathrm{KL}}(q_\varphi(\mathbf{z}|\mathbf{x}) || p_\theta(\mathbf{z}|\mathbf{x})), \tag{6}$$

where $\mathbf{L}_{\theta,\varphi}$ is referred to as the *evidence lower bound* (ELBO):

$$\mathbf{L}_{\theta,\varphi}(\mathbf{x}) \equiv \mathbf{E}_{q_\varphi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\varphi(\mathbf{z}|\mathbf{x})}\right], \tag{7}$$

and the second term is the Kullback–Leibler (KL) divergence:

$$\mathbf{D}_{\mathrm{KL}}(q_\varphi(\mathbf{z}|\mathbf{x}) || p_\theta(\mathbf{z}|\mathbf{x})) = \mathbf{E}_{q_\varphi(\mathbf{z}|\mathbf{x})}\left[\log \frac{q_\varphi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}\right], \tag{8}$$

which is a measure of the 'distance' between two distributions, and is always positive.

From equation (6), we see that the bound $\mathbf{L}_{\theta,\varphi}(\mathbf{x})$ will become tightest when $\mathbf{D}_{\mathrm{KL}}(q_\varphi(\mathbf{z}|\mathbf{x}) || p_\theta(\mathbf{z}|\mathbf{x})) \to 0$, such that our approximation to the posterior $q_\varphi(\mathbf{z}|\mathbf{x}) \approx p_\theta(\mathbf{z}|\mathbf{x})$, becomes exact. However, due to the presence of the $p_\theta(\mathbf{z}|\mathbf{x})$ term, $\mathbf{D}_{\mathrm{KL}}(q_\varphi(\mathbf{z}|\mathbf{x}) || p_\theta(\mathbf{z}|\mathbf{x}))$ cannot be evaluated directly, and so we are not able to directly optimize the likelihood in equation (6). Instead, we seek to maximize the ELBO, thereby achieving an 'optimum' set of weights θ, φ.

The ELBO and its gradient with respect to θ can be computed straightforwardly. The gradients with respect to φ appear more problematic, since the expectation we are calculating is taken over a distribution parametrized by φ. The typical Monte Carlo estimates of this expectation, and its derivatives, are unbiased, but tend to have a high variance, often making the training process unstable. Through a reparametrization presented in Kingma & Welling (2013), it is possible to rewrite this expectation such that the source of randomness is not dependent on φ, and gradients with respect to φ may be calculated with standard Monte Carlo techniques. We are therefore able to optimize $\mathbf{L}_{\theta,\varphi}(\mathbf{x})$ by stochastic gradient descent, and approximately optimize the marginal log likelihood.

2.2 Architecture

In this section, we describe the architecture of the networks $p_\theta(\mathbf{x}|\mathbf{z})$ and $q_\varphi(\mathbf{z}|\mathbf{x})$, and the latent prior $p(\mathbf{z})$. We adopt a convolutional architecture for both the encoder and decoder network.

2.2.1 Latent space

We choose to use a d-dimensional latent space, with a multivariate normal prior, $\mathbf{z} \sim N(0, \mathbf{1}^{d\times d})$.

2.2.2 Encoder

The encoder maps input images $\mathbf{x} \in \mathbf{R}^{256\times 256}$ to latent space distribution parameters, $[\boldsymbol{\mu}^d, \boldsymbol{\sigma}^d] \in \mathbf{R}^{2d}$. It is worth emphasizing the point that, since we are modelling the distribution $p(\mathbf{z}|\mathbf{x})$, the output of the encoder is not a single point in the latent parameter space, but rather a distribution, parametrized by the mean and variance $[\boldsymbol{\mu}^d, \boldsymbol{\sigma}^d]$. The mapping from image to latent space parameters requires both a dimensionality reduction, and a reshaping. We achieve these goals by using a *convolutional neural network*. In the following, we will describe the precise network that we implemented, using the language of neural networks. For details on the motivation for these choices, and their technical meaning, we refer to introductory texts on machine learning and convolutional neural networks such as Goodfellow, Bengio & Courville (2016).

The encoder reduces the dimension of the input image by applying a series of strided convolutions with a rectified linear unit activation function, and then flattens the image for input to a final dense layer connected to the output latent space distribution parameters. Each convolution is characterized by a kernel shape with a number of pixels, k_i, where i indicates the layer, and a stride length, which we set to 2. The values k_i are set during the hyperaparameter optimization stage described in Section 2.3.3. We apply a batch normalization with momentum parameter equal to 0.9 after each convolution. This regularizes the weights, and leads to more stable training. A summary of the encoder model is given in Table 1.

2.2.3 Decoder

The decoder is essentially the reverse process to the encoder, mapping a latent vector $\mathbf{z} \in \mathbf{R}^d$ to an image $\mathbf{x} \in \mathbf{R}^{256\times 256}$. We denote a decoder g, with weights φ as $g_\varphi : \mathbf{z} \to \mathbf{x}$. The primary difference to the structure of the encoder is that we use transverse convolutions as opposed to convolutions, in order to increase the size of each dimension. A summary of the decoder model is given in Table 2.

Table 1. This table shows the structure of the encoder network, $q_\varphi(\mathbf{z}|\mathbf{x})$.

Layer	Layer output shape	Hyperparameters
Input	(256, 256, 1)	
Conv2D	(128, 128, 256)	Stride = 2
ReLu	(128, 128, 256)	
BatchNorm	(128, 128, 256)	Momentum = 0.9
Conv2D	(64, 64, 128)	Stride = 2
ReLu	(64, 64, 128)	
BatchNorm	(64, 64, 128)	Momentum = 0.9
Conv2D	(32, 32, 64)	Stride = 2
ReLu	(32, 32, 64)	
BatchNorm	(32, 32, 64)	Momentum = 0.9
Dense	(1024)	
Dense	(512)	

Table 2. This table shows the structure of the decoder network, $p_\theta(\mathbf{x}|\mathbf{z})$.

Layer	Layer output shape	Hyperparameters
Input	(256, 1)	
Dense	(8192)	
Reshape	(16, 16, 32)	
BatchNorm	(16, 16, 32)	Momentum = 0.9
TransposeConv2D	(32, 32, 128)	Stride = 2
ReLu	(32, 32, 128)	
BatchNorm	(32, 32, 128)	Momentum = 0.9
TransposeConv2D	(64, 64, 64)	Stride = 2
ReLu	(64, 64, 64)	
BatchNorm	(64, 64, 64)	Momentum = 0.9
TransposeConv2D	(128, 128, 32)	Stride = 2
ReLu	(128, 128, 32)	
BatchNorm	(128, 128, 32)	Momentum = 0.9
TransposeConv2D	(256, 256, 16)	Stride = 2
ReLu	(256, 256, 16)	
BatchNorm	(256, 256, 16)	Momentum = 0.9
TransposeConv2D	(256, 256, 1)	Stride = 1

2.3 Training

In this section, we detail the process by which we optimize the weights of the VAE model described in Section 2.2 with respect to the ELBO objective introduced in Section 2.1. The training process requires us to specify the training data set, D, the training *strategy* by which we make updates to the weights θ, φ, and the process of hyperparameter optimization by which we make concrete selections of meta parameters of the model (such as kernel shapes and training parameters).

2.3.1 Data

Machine learning techniques are notoriously data hungry, and will perform best for larger data sets. Standard computer vision data sets on which algorithms are tested (e.g. ImageNet Russakovsky et al. 2015) contain tens of thousands, sometimes millions, of images. However, we have only one sky from which to obtain observations of Galactic dust. As such, we are forced to partition the sky into patches, which we treat as separate images in the training process. In order to obtain ~ thousands of images, the natural linear scale of an individual patch is ~ 10°. Such a small patch size has the advantage that we are then justified in projecting the cutouts on to the flat sky, and applying standard machine learning techniques to the resulting two-dimensional images, sidestepping the issue of defining neural networks that operate on spherical images (for such implementations, see Krachmalnicoff & Tomasi 2019; Perraudin et al. 2019).

We use the *Planck* GNILC-separated thermal dust intensity map at 545 GHz,[2] which we download from the Planck Legacy Archive. In order to extract cutout images from this map, we follow a similar procedure to Aylor et al. (2021). We mask the Galactic plane by excluding all regions at latitudes below 15°. Then we lay down a set of centroids $(l_{i+1}, b_{i+1}) = (l_i + s, b_i + s/\cos(l_i))$, where s is a step size parameter, and $s/\cos(l)$ is a step between longitudes for a given latitude, which ensures the same angular separation in the latitudinal direction. Each centroid is then rotated to the equator, and an $8^\circ \times 8^\circ$ square region around the centroid is projected on to a Cartesian grid with 256 pixels along each size. For $s = 4^\circ$, this results in a data set, D, of 2254 maps. We then shuffle and split D into three groups: a 70 per cent training set, $\mathbf{x}^{\text{train}}$, a 15 per cent validation set, $\mathbf{x}^{\text{val}}$, and a 15 per cent test set, $\mathbf{x}^{\text{test}}$.

Data sets with large dynamic ranges can be difficult to use directly with neural networks due to correspondingly large fluctuations in the updates to weights during the backpropagation of the loss. Therefore, all images were standardized to have unit variance, and zero mean.

In order to artificially increase the diversity of images in our limited sample, we employ two standard data augmentation techniques. During the data preprocessing stage of training, we randomly flip each image along the horizontal and vertical directions, and rotate each image by an integer multiple of 90°. These transformations are not invariant under convolution; however, the transformed image would constitute a perfectly realistic foreground image. The transformed images exist only during the training process, and are not stored or used later.

2.3.2 Strategy

Here, we discuss the training strategy used to learn the weights θ, φ.

As discussed in Section 2, to train a VAE we maximize the lower bound on the log likelihood of the data given in equation (7) with respect to the weights θ, φ. In practice, at each step we compute a Monte Carlo estimate of this quantity:

$$\mathbf{E}_{q_\varphi(\mathbf{z}|\mathbf{x})}\left[\frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\varphi(\mathbf{z}|\mathbf{x})}\right] \approx \log p_\theta(\mathbf{x}|\mathbf{z}) + \log p(\mathbf{z}) - \log q_\varphi(\mathbf{z}|\mathbf{x}), \tag{9}$$

where $\mathbf{x}$ on the RHS is now a minibatch of the data, the size of which is a hyperparameter of the training process. The analysis we present in Section 2.3.3 shows that a batch size of 8 is preferred. For each batch, we then calculate the gradients of this quantity with respect to the weights θ, φ and backpropagate the errors through the network, adjusting θ, φ in accordance with the learning schedule. For this schedule, we used the Adam optimizer (Kingma & Ba 2014) with hyperparameters determined through the optimization process described in Section 2.3.3.

The training was performed by passing over the entire data set 100 times, and in each pass splitting the data into batches of eight images. To guard against overfitting, we evaluated $\mathbf{L}_{\theta,\varphi}(\mathbf{x}^{\text{train}})$ and $\mathbf{L}_{\theta,\varphi}(\mathbf{x}^{\text{val}})$ every five epochs and checked for divergence between these quantities at late epochs. If the network had begun to overfit on the training data, its predictions for the validation set would deteriorate, which would be reflected in a worsening $\mathbf{L}_{\theta,\varphi}(\mathbf{x}^{\text{val}})$. We found that the $\mathbf{L}_{\theta,\varphi}(\mathbf{x}^{\text{train}})$ plateaued after 50 epochs, and saw no divergence between $\mathbf{L}_{\theta,\varphi}(\mathbf{x}^{\text{train}})$ and $\mathbf{L}_{\theta,\varphi}(\mathbf{x}^{\text{val}})$ after training for an additional 50 epochs.

[2] http://pla.esac.esa.int/pla/aio/product-action?MAP.MAP_ID=COM_CompMap_Dust-GNILC-F545_2048_R2.00.fits

Models were built using the TENSORFLOW software package (Abadi et al. 2015), and trained using a Tesla V100 GPU on the Cori supercomputer at NERSC. The time taken to train an individual model depends on the complexity of the architecture (e.g. number of layers, kernel sizes, and number of features). For the nominal architecture shown in Tables 1 and 2, training for 100 epochs on a single Tesla V100 GPU took an average of 12 min over 10 trials, with a standard deviation between trials of 3 s.

2.3.3 Hyperparameter optimization

In this section, we provide motivation for our selection of the model hyperparameters. It is not possible to optimize model hyperparameters such as batch size, or model architecture, using the same stochastic gradient descent technique that is used to optimize model weights and biases. Instead, a limited number of hyperparameter combinations can be trained, and the corresponding model that achieves the best loss after a certain amount of training time, or certain number of epochs, is used. The space of hyperparameters is high dimensional, and so cannot be uniformly densely sampled due to computational cost. Instead, we employed a Bayesian optimization approach in which a few random combinations of hyperparameters are chosen, and trained for 20 epochs each. From this set of hyperparameters, a Gaussian process (GP) model of the loss as a function of hyperparameters is built. From this GP model, new trial candidates are selected, and trained, with the resulting loss then being incorporated into the GP weights. We allowed this process to continue for 100 different trials, and used the hyperparameters that achieved the lowest loss after 20 epochs of training.

3 RESULTS

3.1 Reconstructions

In this section, we present reconstructions of test set images, and compare their pixel value distribution and power spectra.

For a given image, $\mathbf{x}_{\rm test}$, we can sample the posterior as $\mathbf{z}^{(i)}_{\rm test} \sim q_\varphi(\mathbf{z}|\mathbf{x})$, and push these through the decoder to get a reconstructed image $\mathbf{x}^{(i)}_{\rm test} = \mathbf{g}_\theta(\mathbf{z}^{(i)}_{\rm test})$. To summarize the distribution of reconstructed images, we draw L samples and calculate their average:

$$\bar{\mathbf{x}} \approx \frac{1}{L}\sum_{l=1}^{L} \mathbf{g}_\theta(\mathbf{z}^{(l)}_{\rm test}). \tag{10}$$

For the remainder of this section, a 'reconstruction' refers to the calculation of equation (10) with $L = 100$. For a given reconstruction, we can straightforwardly calculate two statistics: (i) the histogram of its pixel values and (ii) the power spectrum. We calculate the histogram of pixel values in 20 bins from -3 to 5, and normalize the count such that the area under the histogram is equal to unity. To calculate the power spectrum, we apply a cosine apodization with a characteristic scale of 1° to the image, such that it smoothly tapers to zero at the edge of the map. We then calculate the mode coupling matrix for this mask, and calculate the uncoupled power spectrum using the NAMASTER code (Alonso et al. 2019). For reasons that will become clear later, we are primarily interested in comparing ranges of multipoles in the signal-dominated regime, well within the resolution limit of the original maps, and so we do not make any efforts to noise debias or account for the beam present in the original maps.

First, we present the reconstructions of three randomly selected test set images, and show the resulting maps, along with the residuals, in Fig. 1. We can see that the network does very well in reconstructing the large-scale features in these test set maps, and the visual quality is sufficient to appear 'real', if lower resolution. Features are well recovered up to $\sim$ degree scales, with features below that scale being smoothed out by the calculation of the expectation in equation (10). The residuals shown in the bottom row of Fig. 1 do not show any visual biases correlated with features in the input maps. To quantify this, we take each image in the test set, calculate the residuals, and calculate the cross power spectrum of the two maps. We take the mean and standard deviation across the test set at each bandpower, and plot the result in Fig. 2.

In Fig. 3, we take a single randomly selected test set image, and show its reconstruction, the pixel value histograms of each image, and their power spectra. As was the case for the three examples shown in Fig. 1, there is excellent visual agreement between the original image and its reconstruction. This is enforced by the excellent agreement between the distribution of pixel values in the two images, shown in the bottom left panel of Fig. 3. The reconstructed power spectrum in the bottom right panel of Fig. 3 also shows excellent agreement up to $\sim$ 400, and suppression of power in the reconstructed image going to smaller scales, consistent with the visual blurriness of the reconstructed image.

In order to compare reconstructions for the whole test set, we now calculate the pixel value distribution and power spectrum for each of the 339 images in the test set and their reconstructions. In order to represent the distribution of pixel value histograms across this test set, we calculate the quartiles and median in each bin, across the test set. In Fig. 4, we plot the 25th percentile, median, and 75th percentile as a function of bin centre, for both the original test set images, and their reconstructions. There is excellent agreement between the two sets of images, with no evidence of any aggregate bias in the reconstructions. In Fig. 5, we compare the power spectra of the test set images and their reconstructions. Fig. 5 shows that the same behaviour as was seen in Fig. 3 is displayed for the entire test set. Spectra are generally well recovered for < 400, with power being increasingly suppressed for > 400, relative to the real image power spectra. Note that we have standardized all test set maps, as described in Section 2.3.1, and so the spread of amplitudes does not reflect the variation one would expect from a random sample of non-standardized foreground spectra.

Here, we are encountering a known issue with VAEs: reconstructed images are often blurry (Kingma et al. 2016; Kingma & Dhariwal 2018; Kingma & Welling 2019). The blurriness can be understood by considering the objective function in equation (7), and inspecting the term $\mathbf{E}_{q_\varphi(\mathbf{z}|\mathbf{x})}[p_\theta(\mathbf{x}, \mathbf{z})]$. Since this expectation is taken with respect to the distribution $q_\varphi(\mathbf{z}|\mathbf{x})$, it will strongly penalize points $(\mathbf{x}, \mathbf{z})$ that are likely under q_φ, but unlikely under p_θ. On the other hand, points that are likely under p_θ, but are not present in the empirical data distribution, will suffer a much smaller penalty. The result is that, if the model is not sufficiently flexible to fit the data distribution exactly, it will compensate by widening the support of $p_\theta(\mathbf{x}, \mathbf{z})$ beyond what is present in the data distribution, inflating the variance of $p_\theta(\mathbf{x}|\mathbf{z})$. Since we have assumed a Gaussian distribution for the decoder model that is independent from pixel to pixel, and given that the signal in the training images is red tilted (as is the case for most natural images containing extended recognizable structures), the increased variance leads to a degradation of small-scale features through the averaging process of equation (10) (Zhao, Song & Ermon 2017). A corollary of the extended support of $p_\theta(\mathbf{x}, \mathbf{z})$ is that sampling the prior in order to generate novel images will not necessarily produce realistic samples (Kingma & Welling 2019).

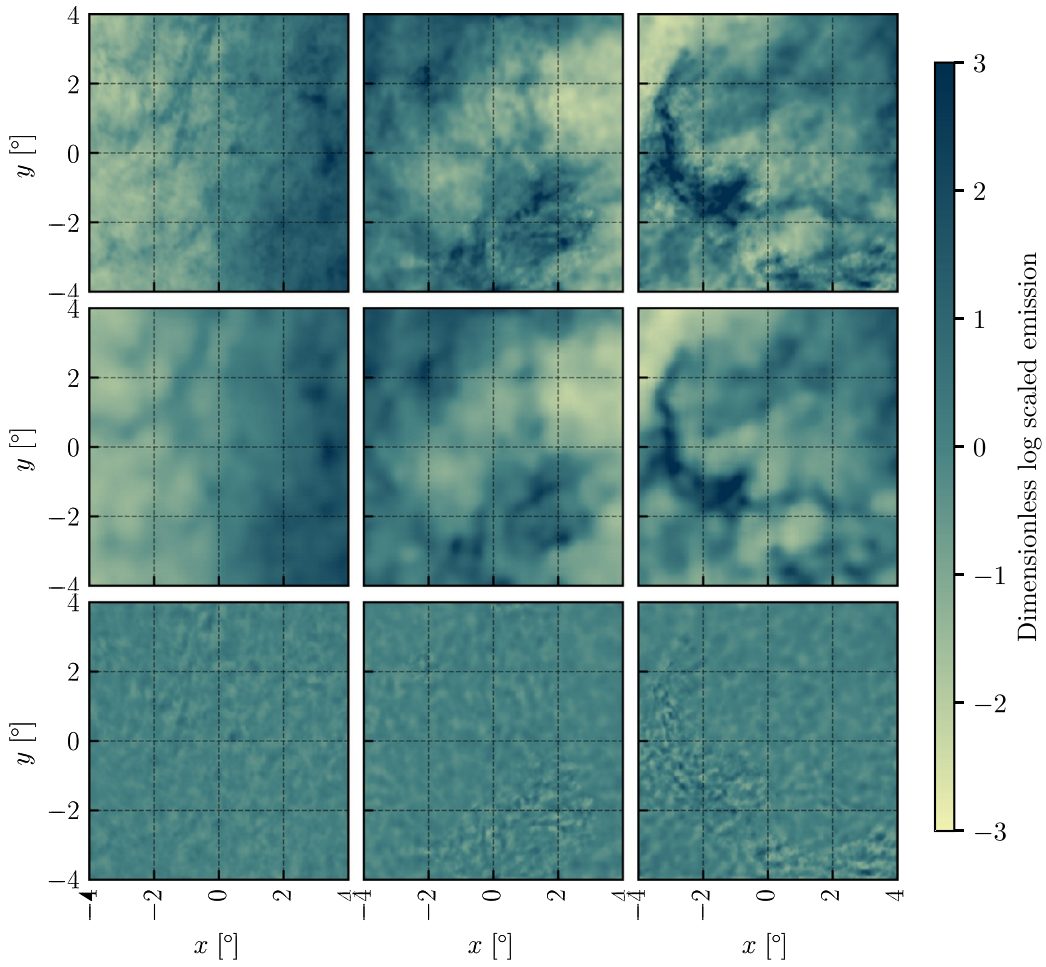

Figure 1. This figure shows the reconstruction of three randomly selected images from the test set, not used during the training or validation of the network. The top row are the original images, the second row are the reconstructions. and the third row are the residuals of the reconstructions. The reconstructions clearly lose small-scale details, but manage to recover the large-scale variations well.

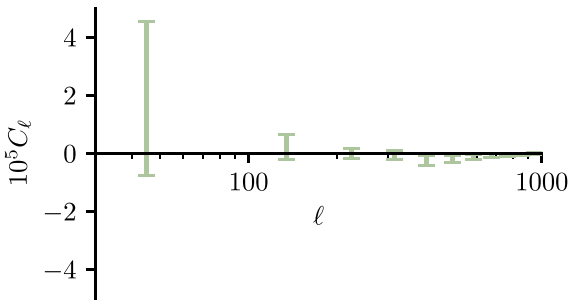

Figure 2. In this figure, we plot a summary of the cross-correlation between residual maps and input maps, calculated across the test set. For each map, we calculate the residuals, and correlate them with the input map. We then plot the mean and standard deviation of each bandpower across the test set, and plot them above. We do not see any evidence for systematic correlations between residual maps and input maps.

One way in which the flexibility of VAEs may be enhanced is through the use of *normalizing flows* (Jimenez Rezende & Mohamed 2015). As the name suggests, the idea here is to start with a simple distribution, such as a multivariate normal, and 'stack' layers of invertible transformations, such that the output may be significantly more complex. There are certain requirements placed on these transformations such that they remain computationally efficient, for example, they must have tractable Jacobians (Jimenez Rezende & Mohamed 2015). Expanding the VAE model presented here by introducing normalizing flows could be expected to improve both the reconstruction quality, and the quality of novel samples, and is the subject of this work.

3.2 Interpolation in the latent space

As a means of investigating the structure of the encoding that has been learned, we study the 'interpolation' between real images, $\mathbf{x}_1$ and $\mathbf{x}_2$, by performing the interpolation between their latent encodings, $\mathbf{z}_1$ and $\mathbf{z}_2$. From the smooth nature of the changes in the resulting continuum of maps, we will see that smooth variations in the latent space result in smooth variations in the map space. This study also demonstrates that the VAE has learned generalizable information about foreground images, as each image generated by sampling along the trajectory has the visual appearance of a foregound map, and the correct statistics, but does not correspond to any real image.

The probability mass in high-dimensional distributions tends to concentrate in a shell relatively far from the modal probability density. Therefore, traversing the latent space in a straight line (in

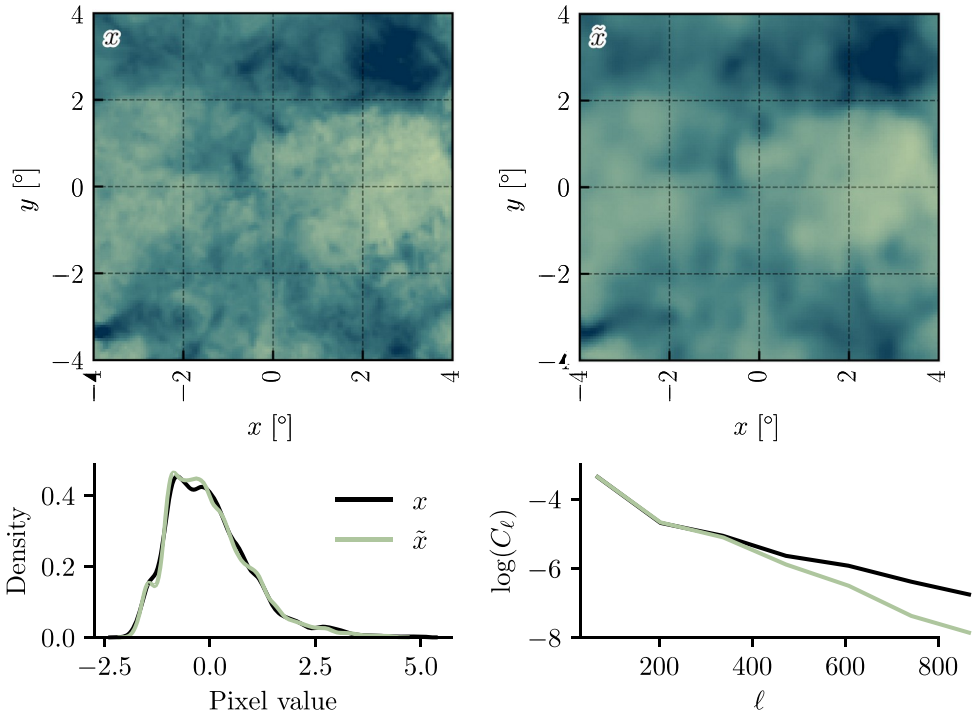

Figure 3. *Top left*: A randomly selected test set image, $\mathbf{x}$. *Top right*: The reconstruction of the test set image, $\tilde{\mathbf{x}}$, as computed using equation (10). *Bottom left*: Kernel density estimate of the distribution of pixel values of the original image, and its reconstruction. *Bottom right*: The log power spectra of the test set image and its reconstruction. Note that the test set images are standardized, as described in Section 2.3.1 therefore these quantities are unitless.

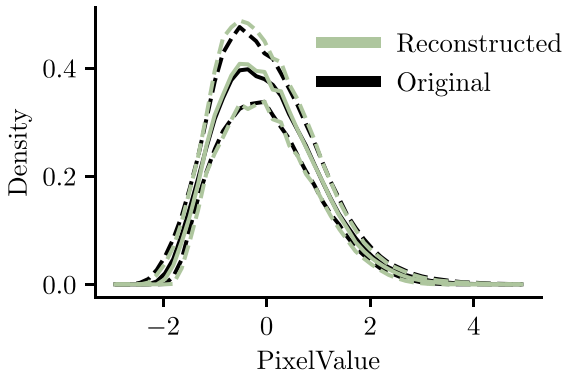

Figure 4. In this figure, we compare the pixel value distributions of the 339 test set images (black), and their reconstructions (green). We calculate quantiles across the test set, and plot the 25th and 75th percentiles (the dashed lines), and the median (the solid lines) as functions of pixel value.

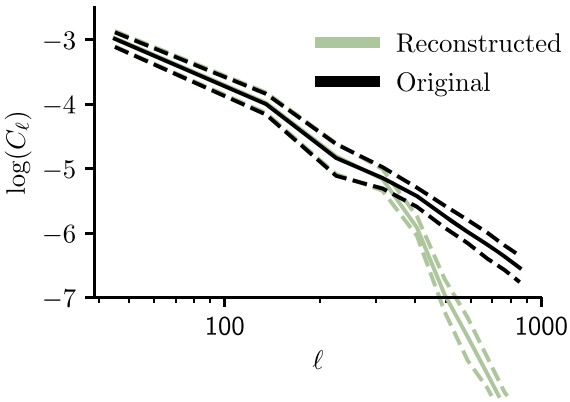

Figure 5. In this figure, we compare the power spectra of the 339 test set images (black) and their reconstructions (green). We calculate quantiles in each bandpower across the test set and plot the 25th and 75th percentiles (the dashed lines), and the median (solid lines) as a function of bandpower.

the Euclidean sense), does not necessarily pass through areas of high probability mass. In order to keep the interpolated points within areas of high probability mass, we interpolate from $\mathbf{z}_1$ to $\mathbf{z}_2$ using spherical trajectories that traverse great circles in the latent space, as the distance from the origin smoothly changes from $|\mathbf{z_1}|$ to $|\mathbf{z_2}|$. Specifically, we follow this continuous trajectory parametrized by some factor λ:

$$\mathbf{z}_{1,2}(\lambda) = \frac{\sin((1-\lambda)\theta)}{\sin\theta}\mathbf{z}_1 + \frac{\sin(\lambda\theta)}{\sin\theta}\mathbf{z}_2, \tag{11}$$

where $\cos(\theta) = \hat{\mathbf{z}}_1 \cdot \hat{\mathbf{z}}_2$. We then take N points along this line corresponding to $\lambda = [1/(N+1), 2/(N+1), \ldots, N/(N+1)]$, and decode to obtain the corresponding map $\mathbf{x}_{1,2}(\lambda) = g_\varphi(\mathbf{z}_{1,2}(\lambda))$.

Fig. 6 shows the smooth transition in image space between the two real images (the top left panel and the bottom right panel) randomly selected from the test set, calculated using the interpolation described above. Features, such as the strong filamentary structures in the centre of the image, transition smoothly in and out of the image, demonstrating that small perturbations in the latent space result in small perturbations in decoded images.

3.3 Data imputation

In this section, we consider a possible application of our trained model to the reconstruction of corrupted data. During the analysis of CMB data, there are many possible reasons that data may

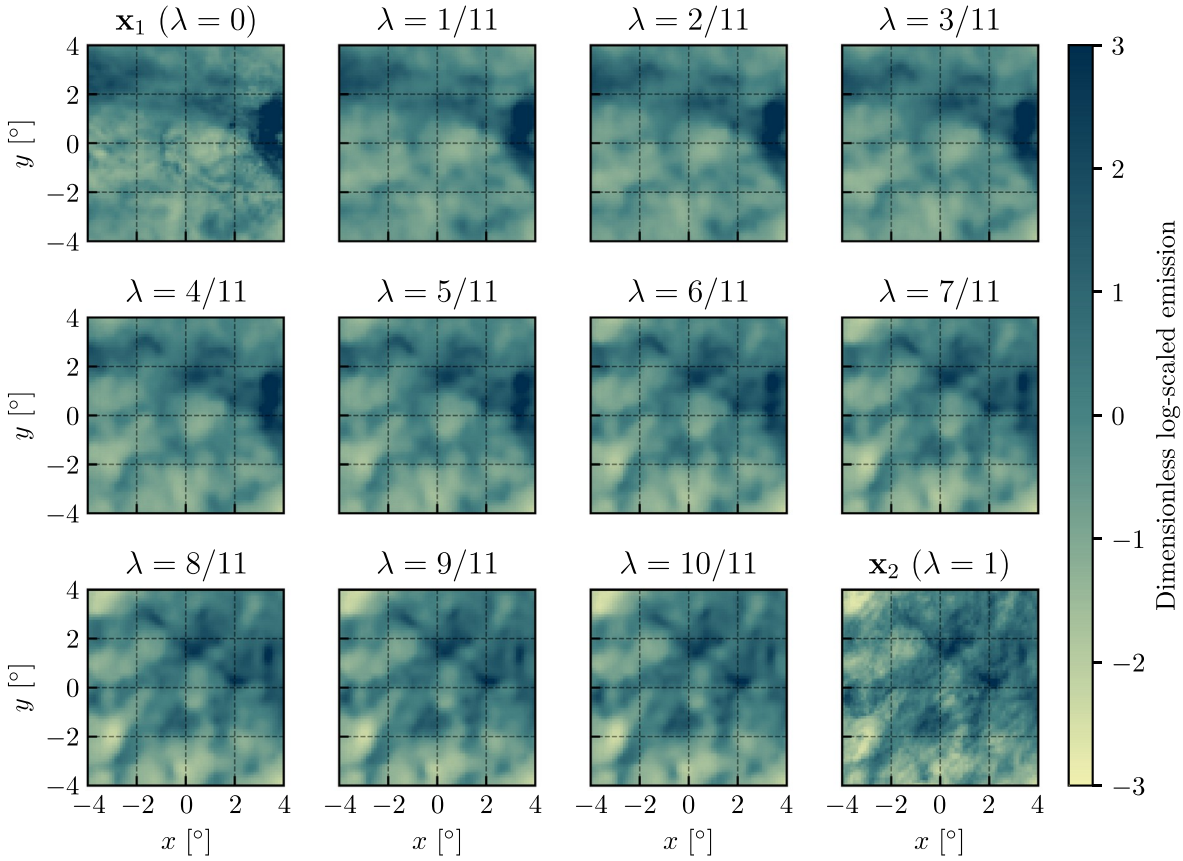

Figure 6. This figure presents synthetic images generated by interpolating between real images, $\mathbf{x}_1$ and $\mathbf{x}_2$, shown in the top left and bottom right panels, respectively. The interpolation is carried out in the latent space using equation (11), and is parametrized by a continuous variable λ. The intermediate panels show the interpolation evaluated at $N = 10$ points along the trajectory.

be incomplete, from masking of point sources, to corruption by uncontrolled systematics.

If trying to estimate the power spectrum from a corrupted map, troublesome regions can be masked and in principle an optimal estimate can still be made by using the correct inverse-variance weighting of the pixels (Gorski 1994; Tegmark & Bunn 1995; Borrill 1999). In practice, however, such a calculation is prohibitively computationally expensive, and pseudo-C_ℓ techniques are relied on (Wandelt, Hivon & G órski 2001; Hivon et al. 2002; Alonso et al. 2019). In the pseudo-C_ℓ approach, near-optimal results can still be obtained, given a careful apodization of the masked region.

When computing higher order estimates, the typical approach is to inpaint the desired region (Gruetjen et al. 2017). This task is simple when the missing emission is well described by Gaussian statistics, as is the case for the CMB (Bucher & Louis 2012). The lack of a similarly simple approach for the non-Gaussian foreground signal means that previous efforts have relied on empirically validated, simple, algorithms, such as diffusive filling (Bucher, Racine & van Tent 2016), and more recently targeted machine learning applications (Puglisi & Bai 2020).

Future surveys will have ever lower noise floors, and will be increasingly contaminated by polarized point sources (Datta et al. 2019). If particularly aggressive masking is required, this could lead to the biasing of standard power spectrum techniques, as well as biasing of higher order estimators (Puglisi & Bai 2020).

The statistical foreground model presented here allows us to take a Bayesian approach to foreground inpainting, in which we may compute a posterior distribution for the missing data, conditioned on the observed data (B öhm, Lanusse & Seljak 2019). This has the advantage of conserving the foregrounds' statistical properties, while also taking into account all of the contextual information in the image, unlike methods such as diffusive inpainting. In the rest of this section, we will present a toy model for corrupted data, and show that we are able to perform inpainting by optimizing the posterior distribution in the latent space.

Representing the contamination as a linear operator A, we can write down a model for the observed data $\mathbf{d}$: $\mathbf{d} = \mathsf{A}\,\mathbf{x} + \mathbf{n}$, where $\mathbf{n}$ is a possible noise term. The posterior distribution of $\mathbf{z}$ is given by Bayes' theorem:

$$\log p(\mathbf{z}|\mathbf{d}) = \log p(\mathbf{z}) + \log p_\theta(\mathbf{d}|\mathbf{z}) - \log p(\mathbf{d}). \tag{12}$$

For a given statistical model of the noise, we have a complete description of the term $\log p(\mathbf{d}|\mathbf{z})$, and we can work with the posterior distribution in the latent space.

As a concrete example, we will consider the case of a binary $N \times N$ masking operator, A, with elements equal to one (zero) where pixels are (un)observed. To form simulated 'corrupted' images, we take random images from the test data set, apply A, and add white Gaussian noise $\mathbf{n}$, characterized by a pixel standard deviation σ: $\mathbf{d}_{\rm test} = \mathsf{A}\,\mathbf{x}_{\rm test} + \mathbf{n}$. The posterior distribution in the latent space is

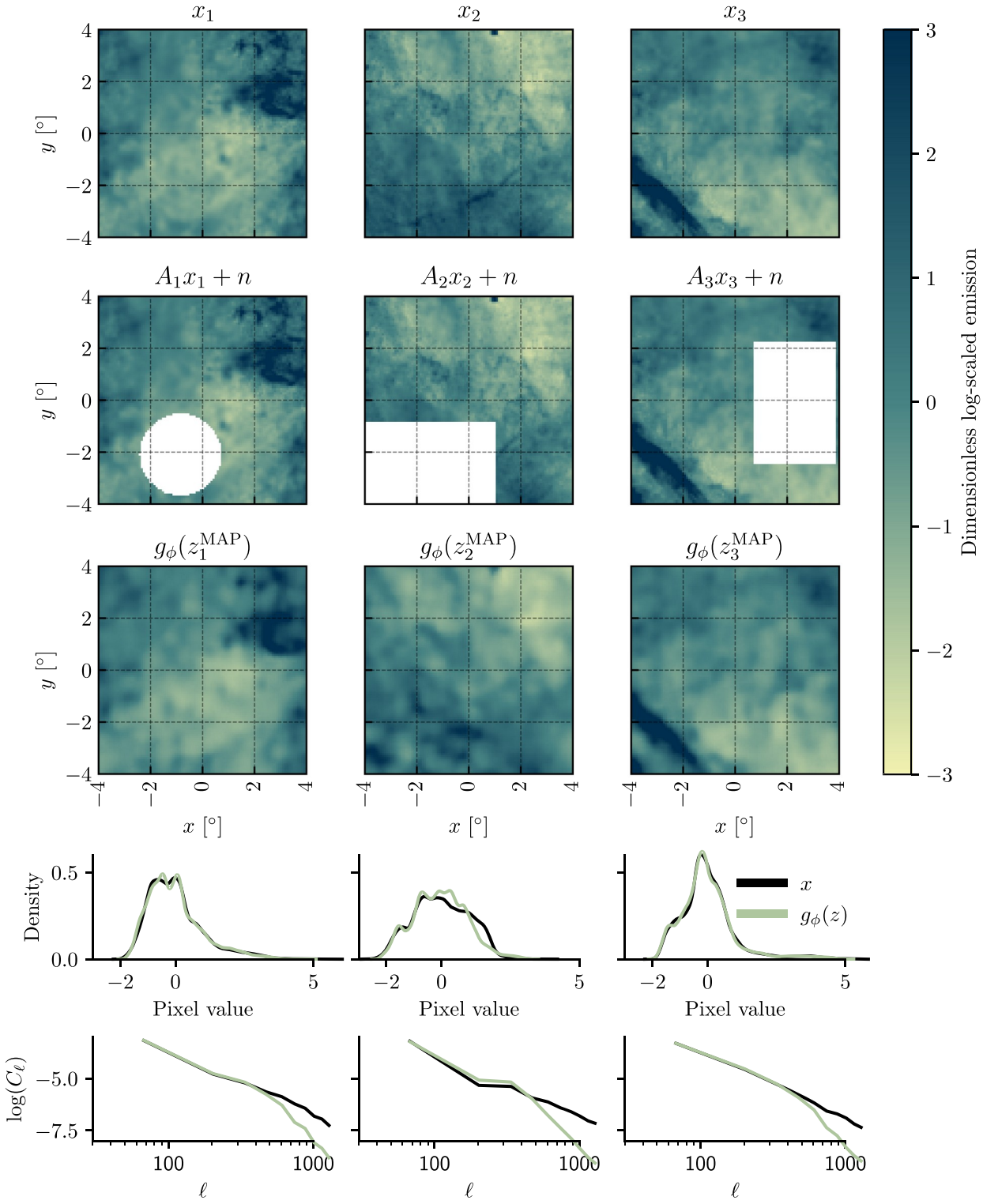

Figure 7. This figure shows three randomly selected test set images, $\mathbf{x}_{1,2,3}$ in the top row. As described in Section 3.3, these images are corrupted with a binary mask $\mathsf{A}_{1,2,3}$ and white noise. The corrupted images are shown in the second row. The third row shows the reconstructed images obtained by maximizing the latent space posterior in equation (13) for each of the three corrupted images, and decoding the resulting points in the latent space. The fourth and fifth rows show the pixel value histograms and power spectra of the original and reconstructed maps.

then:

$$-2\log p(\mathbf{z}|\mathbf{d}_{\rm test}) \propto \mathbf{z}^T\mathbf{z} + \frac{\boldsymbol{\mu}_\theta(\mathbf{z})^T\boldsymbol{\mu}_\theta(\mathbf{z})}{\sigma^2}, \quad (13)$$

where we have written the residual vector as $\boldsymbol{\mu}_\theta(\mathbf{z}) = \mathsf{A}\,\mathbf{g}_\theta(\mathbf{z}) - \mathbf{d}_{\rm test}$.

Fully sampling equation (13) can be computationally expensive due to the dimensionality of $\mathbf{z}$, and is made more challenging by the possibility of $\log p(\mathbf{z}|\mathbf{d}_{\rm test})$ being multimodal. For these reasons, applying standard Markov chain Monte Carlo techniques can often fail to fully explore the posterior (Böhm et al. 2019), and we leave a sampling approach for future work, here taking only a single representative sample by maximizing $\hat{\mathbf{z}}_{\rm test} = \mathrm{argmax}_{\mathbf{z}} \log p(\mathbf{z}|\mathbf{d}_{\rm test})$.

In the following, we will take A to be a masking operator that applies a binary mask to a map. However, as long as a forward model for the corruption operation can be written down (e.g. a Gaussian convolution), the same technique could be applied. We take three randomly selected test set images, $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, and apply three different binary masks, A_1, A_2, A_3. To each corrupted image, we add a white noise realization with a pixel standard deviation of 0.2. For each corrupted, noisy image, we then maximize the posterior in equation (13) to find $\mathbf{z}_i^{\rm MAP}$ using the limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm. In Fig. 7, we show the randomly selected test set images in the first row, the corrupted images in the second row, and the reconstructed map $g(\mathbf{z}_i^{\rm MAP})$ in the third row. We also calculate the pixel value histograms and power spectra of the input and reconstructed maps and show these in the bottom two rows of Fig. 7.

One can see from Fig. 7 that all the images are well reconstructed, and there is no visible effect of the masking remaining in the reconstructions. Comparing the regions in the first and third rows corresponding to the masked areas, we see that the network does not reproduce the exact features in the masked region, for any of the $\mathbf{x}_i$, as expected. However, the network does reconstruct plausible inpaintings, with the correct statistics, given the context in the rest of the image. For example, the reconstruction $g_\varphi(\mathbf{z}_2^{\rm MAP})$ does not replicate the true high-intensity filamentary structure in the input image, $\mathbf{x}_2$, which would be impossible. However, it does recognize from the context that intensity is increasing towards the masked area in the bottom left of the image, and populates that area with high-variance, high-intensity features. Correspondingly, such high-intensity features are not seen in the reconstructed regions of $g_\varphi(\mathbf{z}_{1,3}^{\rm MAP})$, which correspond to relatively low emission regions. The pixel value histograms and power spectra in the last two rows of Fig. 7 show similar behaviour. We see good agreement between the original and reconstructed histograms and power spectra for both the $\mathbf{x}_1$ and $\mathbf{x}_3$ maps, up to the suppression at > 400 common to all reconstructions. On the other hand, we see a disagreement between the original and reconstructed statistics of $\mathbf{x}_2$, due to the higher variance associated with the filled-in region.

These results show that the network has learned generalizable information about foreground behaviour, and is able to inpaint novel foreground emission with correct statistical properties, based on the context of an image. The forward model used in this inpainting process can be easily extended to maps with multiple masks and different types of filtering and noise found in real data.

4 DISCUSSION AND CONCLUSIONS

In this paper, we have presented a new application of VAEs to images of Galactic thermal dust emission. Using a training set extracted from *Planck* observations of thermal dust emission, this technique allowed us to learn a transformation from a space of uncorrelated latent variables with a multivariate normal prior, to the space of possible dust maps.

The training process was validated by computing and comparing summary statistics, including the distribution of pixel values, and power spectra of reconstructed maps, on a test set withheld during the training process. The applicability of the trained model was also demonstrated by reconstructing data corrupted by noise and masking. This was the first use of a trained generative dust model to perform Bayesian inference, and demonstrates the applicability of this approach in the simulation of foreground images, and the Bayesian modelling of polarized CMB data.

The usefulness of this model is currently limited by the flexibility of the posterior, and its ability to fit the true underlying posterior. As was discussed in Section 3.1, this has two main consequences: (i) a naïve sampling of the prior is not guaranteed to produce realistic samples and (ii) reconstructed images are blurry, limiting accuracy to degree scales. Both of these issues may be tackled by increasing the expressiveness of the model (Kingma & Welling 2019), which we plan to do by introducing a normalizing flow to link the prior and latent space (Kingma et al. 2016).

As discussed in the Section 1, our main goal is to model polarized dust emission. We attempted a similar analysis to that presented here by repeating the training procedure on a network that accepted an additional 'channel' as input, representing a tuple of Stokes Q and U parameters, rather than only Stokes I, and using the Planck 353 GHz polarization observations to form a training set. We found that the network was not able to learn any meaningful information from this set-up, consistent with what similar analyses have found (Petroff et al. 2020). In order to extend our analysis to polarization, we are therefore exploring the use of MHD simulations (Kim et al. 2019) as a training set. Kim et al. (2019) have demonstrated that simulations of a multiphase, turbulent, magnetized ISM produce synthetic observations of the ISM with statistics (such as the ratio of E power to B power, and the tilt of the EE and BB power spectra) matching those of real skies. Our initial results have shown that this is a promising alternative to the use of real data in training generative networks.

ACKNOWLEDGEMENTS

We would like to acknowledge useful conversations with Ethan Anderes and Kevin Aylor in the preparation of this work. This work was supported by an XSEDE start up allocation, PHY180022. This work was supported in part by the National Science Foundation via awards OPP-1852617 and AST-1836010. We also acknowledge the use of the Perlmutter preparedness GPU allocation on the Cori super computer at NERSC.

DATA AVAILABILITY

The data used in this study are available on the Planck Legacy Archive at the URL: http://pla.esac.esa.int/pla/aio/product-action?MAP.MAP_ID=COM_CompMap_Dust-GNILC-F545_2048_R2.00.fits.

REFERENCES

Abadi M. et al., 2015, TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, preprint (arXiv:1603.04467)
Ade P. et al., 2019, J. Cosmol. Astropart. Phys., 2019, 056
Aiola S. et al., 2020, J. Cosmol. Astropart. Phys., 12, 047
Alonso D., Sanchez J., Slosar A., LSST Dark Energy Science Collaboration, 2019, MNRAS, 484, 4127

Aylor K., Haq M., Knox L., Hezaveh Y., Perreault-Levasseur L., 2021, MNRAS, 500, 3889
Benson B. A. et al., 2014, in Holland W. S., Zmuidzinas J., eds, Proc. SPIE Conf. Ser. Vol. 9153, Millimeter, Submillimeter, and Far-Infrared Detectors and Instrumentation for Astronomy VII. SPIE, Bellingham, p. 91531P
BICEP2 Collaboration, 2018, Phys. Rev. Lett., 121, 221301
Böhm V., Lanusse F., Seljak U., 2019, preprint (arXiv:1910.10046)
Borrill J., 1999, preprint(astro-ph/9911389)
Brock A., Donahue J., Simonyan K., 2018, preprint (arXiv:1809.11096)
Bucher M., Louis T., 2012, MNRAS, 424, 1694
Bucher M., Racine B., van Tent B., 2016, J. Cosmol. Astropart. Phys., 2016, 055
Datta R. et al., 2019, MNRAS, 486, 5239
Dinh L., Krueger D., Bengio Y., 2014, accepted as contribution to 2015 International Conference for Learning Representations, preprint (arXiv:1410.8516)
Dinh L., Sohl-Dickstein J., Bengio S., 2016, accepted as contribution to 2017 International Conference for Learning Representations, preprint (arXiv:1605.08803)
Goodfellow I. J., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y., 2014, Neural Information Processing Systems, preprint (arXiv:1406.2661)
Goodfellow I., Bengio Y., Courville A., 2016, Deep Learning. MIT Press, Cambridge, Massachusetts
Gorski K. M., 1994, ApJ, 430, L85
Grover A., Dhar M., Ermon S., 2017, preprint (arXiv:1705.08868)
Gruetjen H. F., Fergusson J. R., Liguori M., Shellard E. P. S., 2017, Phys. Rev. D, 95, 043532
Hivon E., Górski K. M., Netterfield C. B., Crill B. P., Prunet S., Hansen F., 2002, ApJ, 567, 2
Hui H. et al., 2018, in Zmuidzinas J., Gao J.-R., eds, Proc. SPIE Conf. Ser. Vol. 10708, Millimeter, Submillimeter, and Far-Infrared Detectors and Instrumentation for Astronomy IX. SPIE, Bellingham, p. 1070807
Jimenez Rezende D., Mohamed S., 2015, PMLR, 37, 1530
Jimenez Rezende D., Mohamed S., Wierstra D., 2014, PMLR, 32, 1278
Kamionkowski M., Kosowsky A., Stebbins A., 1997, Phys. Rev. Lett., 78, 2058
Kim C.-G., Choi S. K., Flauger R., 2019, ApJ, 880, 106
Kingma D. P., Ba J., 2014, CoRR, 2015, preprint (arXiv:1412.6980)
Kingma D. P., Dhariwal P., 2018, Advances in Neural Information Processing Systems 31, preprint (arXiv:1807.03039)
Kingma D. P., Welling M., 2013, 2nd International Conference on Learning Representations, 2014, preprint (arXiv:1312.6114)
Kingma D. P., Welling M., 2019, Foundations and Trends in Machine Learning, 12, 307
Kingma D. P., Salimans T., Jozefowicz R., Chen X., Sutskever I., Welling M., 2016, CoRR, preprint (arXiv:1606.04934)
Knox L., Turner M. S., 1994, Phys. Rev. Lett., 73, 3347
Krachmalnicoff N., Puglisi G., 2021, AJ, 911, 1
Krachmalnicoff N., Tomasi M., 2019, A&A, 628, A129
Millea M., Anderes E., Wandelt B. D., 2020a, Phys. Rev. D, 102, 123542
Millea M. et al., 2020b, preprint (arXiv:2012.01709)
Perraudin N., Defferrard M., Kacprzak T., Sgier R., 2019, Astron. Comput., 27, 130
Petroff M. A., Addison G. E., Bennett C. L., Weiland J. L., 2020, ApJ, 903, 104
Planck Collaboration VI, 2020, A&A, 641, A6
Puglisi G., Bai X., 2020, ApJ, 905, 143
Razavi A., van den Oord A., Vinyals O., 2019, Advances in Neural Information Processing Systems, 32
Regaldo-Saint Blancard B., Allys E., Boulanger F., Levrier F., Jeffrey N., 2021, preprint (arXiv:2102.03160)
Russakovsky O. et al., 2015, Int. J. Comput. Vis., 115, 211
Seljak U., Zaldarriaga M., 1997, Phys. Rev. Lett., 78, 2054
Smoot G. F. et al., 1992, ApJ, 396, L1
Tegmark M., Bunn E. F., 1995, ApJ, 455, 1
van den Oord A., Kalchbrenner N., Kavukcuoglu K., 2016a, PMLR, 48, 1747
van den Oord A. et al., 2016b, preprint (arXiv:1609.03499)
Wandelt B. D., Hivon E., Górski K. M., 2001, Phys. Rev. D, 64, 083003
Zhao S., Song J., Ermon S., 2017, CoRR, preprint (arXiv:1702.08658)

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