Inpainting CMB maps using Partial Convolutional Neural Networks

Gabriele Montefalcone,[a,1] Maximilian H. Abitbol,[b] Darsh Kodwani[b] and R.D.P. Grumitt[b]

[a]Department of Physics, Princeton University, Princeton, NJ, USA

[b]University of Oxford, Department of Physics, Denys Wilkinson Building, Keble Road, Oxford, OX1 3RH, UK

E-mail: gm9@princeton.edu, maximilian.abitbol@physics.ox.ac.uk, darsh.kodwani@physics.ox.ac.uk, richard.grumitt@physics.ox.ac.uk

Abstract. We present a novel application of partial convolutional neural networks (PCNN) that can inpaint masked images of the cosmic microwave background. The network can reconstruct both the maps and the power spectra to a few percent for circular and irregularly shaped masks covering up to 10% of the image area. By performing a Kolmogorov-Smirnov test we show that the reconstructed maps and power spectra are indistinguishable from the input maps and power spectra at the 99.9% level. Moreover, we show that PCNNs can inpaint maps with regular and irregular masks to the same accuracy. This should be particularly beneficial to inpaint irregular masks for the CMB that come from astrophysical sources such as Galactic foregrounds. The proof of concept application shown in this paper shows that PCNNs can be an important tool in data analysis pipelines in cosmology.

---

[1]Corresponding author.

Contents

1 Introduction

Observations of the cosmic microwave background (CMB) have provided crucial experimental tests for theoretical models describing the origin and evolution of the universe. We currently have a model that describes the observations from the CMB and other cosmological probes extremely well, called the ΛCDM model. Obtaining precision constraints on cosmological parameters requires a thorough understanding of instrumental and astrophysical systematic errors. Of particular importance for CMB measurements, are astrophysical foregrounds; non-CMB sources of microwave emission from our own galaxy and from extra-galactic sources that obfuscate CMB observations [1]. Typically, pixels that are completely dominated by foregrounds are removed from the maps and inferences are made from masked maps using cut-sky power spectrum estimators. One of the most widely-used estimators, largely due to their computational efficiency, are pseudo-C_l estimators [2, 3]. Pseudo-C_l methods construct unbiased, but sub-optimal power spectrum estimators on the cut-sky. One may ask whether the missing values can be replaced by physically or statistically motivated values and whether that can allow for better power spectrum estimation than the cut-sky pseudo-C_l estimators. This question has previously been considered in [4], where it was found that using a linear inpainting scheme could significantly outperform pseudo-C_l methods. In this paper we

demonstrate the ability of our machine learning based inpainting scheme at recovering the correct input power spectra as a proof of concept. However, this is likely to be particularly advantages when attempting to construct potential non-Gaussian features in masked CMB maps.

The general procedure of replacing missing values through inpainting is a common problem in a variety of fields in addition to cosmology such as computer vision, satellite imaging etc. A review of inpainting algorithms can be found in [5]. Machine learning techniques have become ubiquitous in solving inpainting problems as they do not rely on an a-priori model that describes the data. Instead they learn statistical features about the data from training data. Of course, it is possible that a forward model is required to create a training set in the first place - this is indeed the case for CMB maps, as we only have a single realisation of the CMB map.

Simulating a CMB map requires a set of cosmological parameters that will determine the statistical properties of the map. The observed CMB will come from the "true" cosmological parameters[1]. Since we do not have access to the true parameters, in order to infer the actual statistical properties of the map we need to sample a range of cosmological parameters and see which one of them best matches the observations. The training set we make for inpainting CMB maps includes maps that are evaluated for different cosmological parameters.

The traditional approach to inpainting missing values is to use a Gaussian constrained realisation of the map [6]. There have been various studies that have recently attempted to use machine learning techniques to inpaint CMB images. In [7] a variational auto-encoder is used to reconstruct CMB maps. This study compared the difference between the reconstructed image and the original image at the map level, however, in cosmology the more relevant function is the power spectrum as that is the object typically used for inference of cosmological parameters. Further studies have used different machine learning methods and also compared power spectra between the true images and the inpainted images. In [8] three separate machine learning methods are used to inpaint galactic foreground maps. One of the most popular techniques in computer vision is called Generative Adversarial Networks (GANs) [9] and it has been used to inpaint CMB maps in [10]. Moreover, the symmetrical properties of the sphere are taken advantage of by using a neural network architecture that preserves $SO(3)$ symmetry in [11] to inpaint CMB images for foreground removal. Traditional deep learning methods use both the masked (empty) pixels and the surrounding pixels with signals in an image to inpaint the masked region. This is known to lead to bluriness in the inpainted image [12]. In this paper we use a novel concept of partial convolutions in convolutional neural networks [12] to inpaint CMB maps. The strength of using partial convolutions is that they only use the non-masked pixels to draw statistical information and inpaint the masked pixels with that information. This is done by masking the convolution matrix itself. Our model outperforms the other recently developed ML methods that inpaint pure CMB maps. Specifically, we obtain mean square error (MSE), mean absolute error (MAE) and peak signal to noise ratio (PSNR) equal to 0.0005, 0.0044 and 32.881, as opposed to 0.0055, 0.134 and 23.989 obtained in [7] for similar mask sizes[2]. Moreover, we are able to reproduce the CMB patch power spectra to percent accuracy up to $\ell_{\max} = 2048$ for irregularly shaped masks that cover $\lesssim 15\%$ of the total area, compared to $\ell_{\max} \sim 1500$ obtained in [10] for regular masks of

---

[1]This implicitly assumes that the ΛCDM model is a the correct model of the universe.

[2]The numerical values for the three parameters presented here correspond to the average values obtained from comparing 500 ground-truth and inpainted CMB patches from the test set and masked with $\mathbf{M_r}$ (see 2.2.2 for more details on this).

equivalent size.

This paper is structured as follows: Section 2 described the machine learning method of partial convolutions that is used for inpainting. The results, for the pixel distribution and the power spectra of inpainted maps, are shown in section 3 and finally we conclude in section 4 with a summary of our results and potential future avenues of research using partial convolutions.

2 Machine learning algorithm

Inpainting refers to the process of reconstructing lost or deteriorated parts of an image without introducing significant differences. Several algorithms have been developed to solve such tasks. They can be classified into two approaches:

- Patch-based synthesis and image statistics [13–16]

- Machine learning (ML) techniques involving neural networks [17–21].

We focus on the second approach in this paper. Convolutional neural networks (CNNs) have proven particularly effective in image analysis, with their network architecture being specifically designed for the extraction of information from image data through a series of convolutional layers.

Each layer consists of a set of learnable filters which are convolved with the input image, sliding across its pixels and identifying key features in the image. The deeper the network, the more features that can be extracted, such that, by building a CNN with a sufficiently large number of layers, one is able to recognize even the most intricate images. This, however, can also lead to overfitting and thus has to be accounted for by techniques such as dropout [22]. Given that the CNN filter responses are conditioned on both valid pixels as well as the substitute values in the masked holes, a CNN algorithm that fills in masked parts of an image suffers from the dependence on the initial hole values. This dependency on the hole values can result in several imperfections in the output image such as color discrepancy, lack of texture in the hole regions and blurriness [12]. These artifacts are typically removed by downstream processing of the resulting images which increases the computational cost of inpainting [23, 24].

In an attempt to solve such inaccuracies in the image reconstruction, whilst minimising the inpainting time and cost, a deep learning algorithm based on partial convolutional[3] neural networks (PCNNs) has been developed in [12, 25]. This new approach is also motivated by its ability to deal with random irregular holes in place of the traditional rectangular masks which may lead to over-fitting and limit the usefulness of the algorithm. The main extension introduced by PCNNs is that the convolution is masked and re-normalized in order to be conditioned only on valid pixels. This is accomplished with the introduction of the automatic mask update step, which removes any masking where the partial convolution was able to operate on an unmasked value. Thus, with this method, the convolutional results depend only on the non-hole regions at every layer such that, given sufficient successive layers, even the largest masks eventually shrink away, leaving only valid responses in the feature map.

Typically, inpainting in relation to masked CMB maps consists of generating constrained Gaussian realizations to fill in the masked regions [6, 26]. However this approach fails to deal with non-Gaussian components in the map and can be computationally expensive. Recently,

---

[3]Sometimes also called *segmentation-aware convolutions.*

machine learning algorithms, specifically deep learning using CNNs and GANs, have been increasingly applied to this problem with promising results [8, 10, 11]. Nevertheless, all of these approaches have so far only focused on neural networks with pure convolutional layers that fill in square or circular masks. In this paper, we decide to adapt the PCNN algorithm developed in [12] to learn the statistical properties of CMB maps and efficiently fill in irregularly masked areas, avoiding biases caused by the initial hole values and the shape of the masks.

2.1 Network Architecture

The proposed model is U-Net [27] like and characterized by two main elements: the partial convolutional layer and the loss function. The details are given below and we follow the notation in [12].

2.1.1 Partial Convolutional Layer

The key element in this network is the partial convolutional layer. Given the convolutional filter weights $\mathbf{W}$, its corresponding bias b and a binary mask of 0s and 1s $\mathbf{M}$; the partial convolution on the current pixel values $\mathbf{X}$ at each layer, is defined as:

$$x' = \begin{cases} \mathbf{W}^T(\mathbf{X} \odot \mathbf{M})\frac{\text{sum}(\mathbf{1})}{\text{sum}(\mathbf{M})} + b, & \text{if } \text{sum}(\mathbf{M}) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

where x' corresponds to the updated pixel values and $\odot$ is the element-wise multiplication. The scaling factor $\frac{\text{sum}(\mathbf{1})}{\text{sum}(\mathbf{M})}$ applies some appropriate scaling to adjust for the varying amount of valid (unmasked) inputs. After each convolution, the mask is removed at the location where convolution was able to condition its output on the valid input. The updated mask pixel values m' are expressed as:

$$m' = \begin{cases} 1, & \text{if } \text{sum}(\mathbf{M}) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.2}$$

2.1.2 U-Net structure

The model is characterized by a U-Net architecture where all the convolutional layers are replaced with partial convolutional layers using nearest neighbor up-sampling in the decoding stage. The input images taken in are 128×128 pixels. The coding and decoding stages are made of 7 layers where in the coding phase, the number of filters in each convolution steadily increases from 32 to 128 meanwhile the kernel size decreases from 7 to 3. In the decoding stage, the kernel size is kept constant at 3 meanwhile the number of filters in each convolution decreases steadily until the final concatenation with the input image, where it is equal to 3. The last layer's concatenation with the original input image and mask serves to allow the model to copy non-hole pixels. Details of the PCNN architecture can be found in the Appendix Tab.1.

2.1.3 Loss Function

The loss function aims to evaluate both per-pixel reconstruction accuracy and composition. The full form of the loss is taken from [12] and is presented in detail in appendix A.2.

$$\begin{aligned} \mathcal{L}_{\text{total}} = \mathcal{L}_{\text{valid}} + 6\mathcal{L}_{\text{hole}} + 0.05\mathcal{L}_{\text{perceptual}} \\ + 120\mathcal{L}_{\text{style}} + 0.1\mathcal{L}_{\text{tv}}, \end{aligned} \tag{2.3}$$

where $\mathcal{L}_{\text{valid}}$, $\mathcal{L}_{\text{hole}}$ are respectively the losses on the output for the non-hole and the hole pixels; $\mathcal{L}_{\text{perceptual}}$ is based on ImageNet pre-trained VGG-16 (*pool*1, *pool*2 and *pool*3 layers); $\mathcal{L}_{\text{style}}$ is similar to the perceptual loss where it's first performed an auto-correlation on each feature; and finally $\mathcal{L}_{\text{tv}}$ refers to the smoothing penalty on R, where R is the region of 1-pixel dilation of the hole region.

2.2 Training

2.2.1 Dataset

We use CAMB [28] to generate 50,000 Gaussian, temperature-only, random CMB realizations for training, validation, and testing. Maps are generated with Healpix [29] $N_{side} = 1024$ with the Planck 2018 ΛCDM parameters [30]. Images are cut out and projected onto a flat sky, with 128x128, 6.87 arcminute square pixels. Each image is therefore approximately 14.65 degrees on a side. We generate 10,000 images for each of 5 different values of the scalar index n_s, between 0.8 and 0.96. Each set of 10,000 images is divided into training, validation and test sets, with 7,000, 1,500, and 1,500 images respectively. The PCNN architecture takes 3 RGB 8-bit inputs. We convert the pixel temperature values to 8-bit RGB values using a linear mapping (e.g. the images are represented as 128x128x3 8-bit integers). The mapping parameters are saved in order to reconstruct the original temperature scale after inpainting.

2.2.2 Masking

The masking is performed through two independent methods.

- **Irregular masks**: The first generates masks composed of ellipses, circles and lines in random order, amount and size to cover about 10% and 25% of the input 128×128 pixel image. We will refer to them as $\mathbf{M_r}$ and $\mathbf{M_R}$ and they are used, respectively, for the fidelity evaluation and in the training process.

- **Regular masks** The second method generates centered circular masks of various radius that can cover from 0 to roughly 100% of the input 128x128 pixel image. This last masking is instead used in the fidelity evaluation both to facilitate the comparison with previous results and to measure the ability of the algorithm to reconstruct the image in terms of the percentage of area covered. We will refer to these masks as $\mathbf{M_{cR}}$ where R is equal to the radius of the circle in pixel units.

2.2.3 Training Procedure

At first we initialize the weights using the pre-trained VGG 16 weights from ImageNet [31]. This is done for image classification purposes and recognize the different classes of input CMB patches based on the value of the spectral index, n_s. The training is then performed using the Adam optimizer [32] in two stages. The first stage has a learning rate of 0.0001 and enables batch normalization in all layers. The second stage has a learning rate of 0.00005 and only enables the batch normalization in the decoding layers. This is done to avoid the incorrect mean and variance issues that occur with batch normalization in the masked convolutions (as the mean and variance will be computed for the hole pixels as well) and also helps us to achieve faster convergence.

Overall, the program trained for 29 epochs, with batch size of 5 and respectively 10,000 training steps and 1,500 validation steps. The total computational cost was roughly 100 GPU hours.
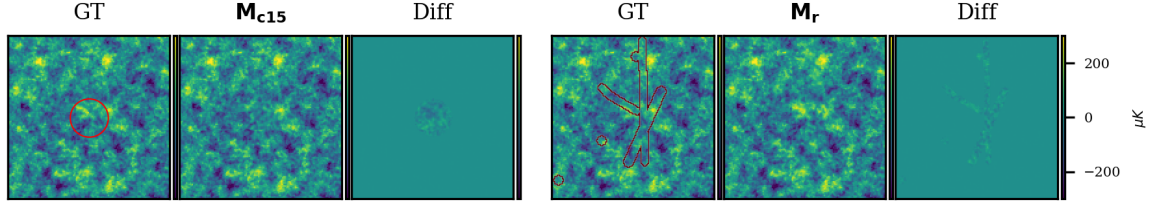
Figure 1: The figure shows a ground truth (GT) thumbnail image from the $n_s = 0.96$ test set and its corresponding predictions given a circular mask of radius 15 pixels ($\mathbf{M_{c15}}$) and an irregular mask of approximately equal size ($\mathbf{M_r}$). For each masking, the difference (Diff) between the inpainted and GT image is also shown. For examples of reconstructed thumbnails from different n_s values and mask size refer to figure 5 and 6 in Appendix B.

3 Results

Figures 1, 5 and 6 show examples of maps extracted from the test set and reconstructed via the PCNN algorithm for the different masking methods outlined in Section 2.2.2.

To determine the efficiency of the algorithm in reconstructing the CMB patches, we follow the approach from [8, 33, 34] and focus on evaluating the ability to replicate the summary statistics of the underlying signal that needs to be reconstructed [8]. Specifically, for each n_s value, we compare 500 ground-truth CMB patches from the test set with their corresponding inpainted maps, through two metrics: (1) the pixel intensity distribution; (2) the angular power spectra. Numerically, we perform this comparison by running the Kolmogorov-Smirnov (KS) two-sample test which assesses the likelihood that the distribution of inpainted CMB patches is drawn from the ground-truth distribution.

3.1 Pixel intensity distribution

To compute the pixel intensity distribution, all the sample maps are scaled with `MinMax` rescaling to [-1, 1] as in [8]. This is done in order to more easily compare the difference between the ground-truth and reconstructed maps and Gaussian distributions. The left plots in Fig.2 show the resulting histograms of the pixel intensities for each n_s value, when masked respectively by $\mathbf{M_{c15}}$ (2a) and $\mathbf{M_r}$ (2b). The differences between the pixel distributions are negligible and $\lesssim 1\%$ for both $\mathbf{M_{c15}}$ and $\mathbf{M_r}$ and all the n_s values. The resulting KS test p-values are $p = 0.999$ for both $\mathbf{M_{c15}}$ and $\mathbf{M_r}$, which shows that the PCNN algorithm was able to accurately reproduce the ground-truth CMB patterns.

3.2 Angular power spectrum

The angular power spectrum in each flat square CMB patch is estimated with `Pixell`[4]. It is binned into equally spaced multipoles with $\Delta\ell = 32$ and maximum multipole $\ell_{\max} = 2048$, chosen to correspond roughly to the available sky fraction and `Pixell` power spectrum accuracy. The right plots in Fig.2 show the resulting average power spectra for each n_s value, when masked respectively by $\mathbf{M_{c15}}$ (2a) and $\mathbf{M_r}$ (2b). The power spectra residuals for both $\mathbf{M_{c15}}$ and $\mathbf{M_r}$ are in the same order as the corresponding pixel distribution residuals. This uniformity between the pixel distribution and power spectra residuals is evidenced in the equivalent KS test p-values obtained, with $p = 0.999$ for both $\mathbf{M_{c15}}$ and $\mathbf{M_r}$. Moreover, at high ℓ, we note that the residuals are always negative, which implies that PCNN underestimates the original power spectrum at this multipole range. This could be caused by additional

---

[4]https://github.com/simonsobs/pixell

smoothing in the inpainted region, leading to reduced small-scale power. That said, it is important to emphasize that the obtained residuals are fully consistent with zero, as shown by the associated error bars, defined as the standard deviation from the mean value of the power spectrum at each $\Delta\ell$ bin.
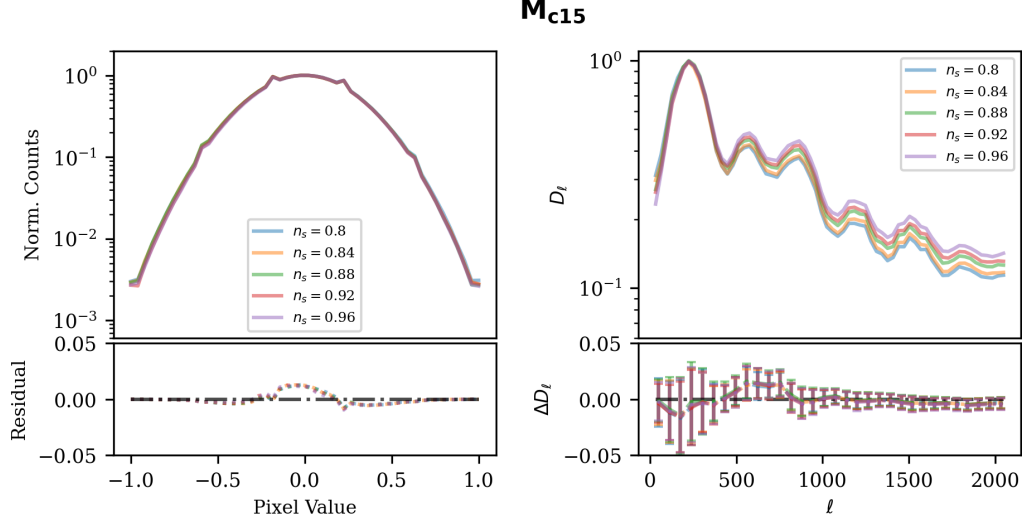

(a) Predictions given a circular mask of radius 15 pixels ($\mathbf{M_{c15}}$).
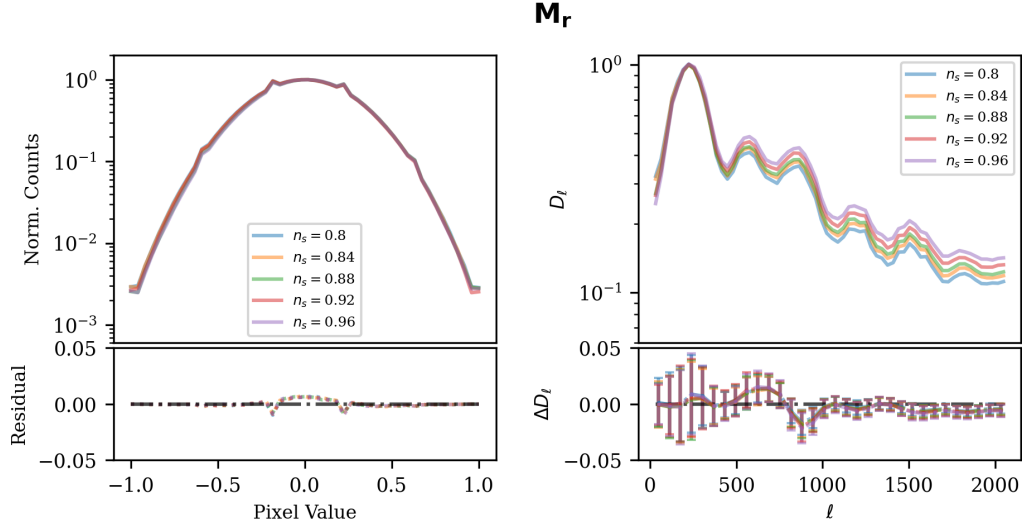
(b) Predictions given an irregular mask of approximately the same size as $\mathbf{M_{c15}}$ ($\mathbf{M_r}$).

Figure 2: The left and right plot of Fig. 2a and 2b contain respectively the pixel intensity distribution and the average angular power spectrum of 500 inpainted maps with PCNN from the test set for each n_s values considered. The distributions are normalized to 1 and set in log scale. At the bottom of each plot, the distributions of the residuals between the ground truth and inpainted maps are shown. As expected, for the pixel intensity distributions they are larger in the center rather than on their tails and overall they are always $\lesssim 1\%$. With regard to the angular power spectra, at high ℓ the residuals, though very small, tend to always underestimate the original value. However, they are still consistent with zero as emphasized by the associated error bars (for clarity, the ΔD_ℓ plot was rebinned with $\Delta\ell = 64$).
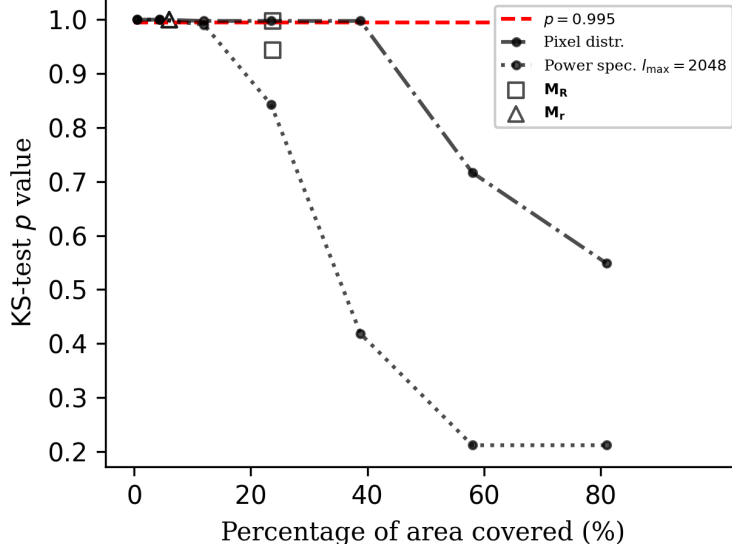
Figure 3: The figure presents the KS test p-values as a function of the percentage of area covered by circular masks $\mathbf{M_{cR}}$ of radius $R_k = 10 \cdot k + 5$ pixels and $k \in [0, 6]$. Respectively, the pixel intensity distribution and the power spectra given $\ell_{\max} = 2048$ are the dot-dashed and dotted line. The plot also contains the KS test p-values given irregular masks $\mathbf{M_r}$ (triangle) and $\mathbf{M_R}$ (square) which present the same values as the corresponding circular masks of equal size, emphasizing the ability of the PCNN algorithm to reproduce ground truth thumbnail images, independently of the mask shape.

3.3 Performance in terms of mask size

To assess the performance of the PCNN algorithm in terms of the percentage of CMB patch area covered, we use the circular masks $\mathbf{M_{cR}}$ with increasing radius R from 5 to 65 pixels, in steps of 10 pixels. We use the same two metrics as for the fidelity evaluation (pixel intensity distribution and angular power spectrum) and compare 500 ground-truth CMB patches from the $n_s = 0.96$ test set with their corresponding inpainted maps, for each $\mathbf{M_{cR}}$ masking. The choice of the $n_s = 0.96$ test over the other n_s values is arbitrary. In fact, as presented in subsections 3.1 and 3.2, the obtained KS test p-values are independent of the n_s value under consideration.

The results of this analysis are plotted in Fig. 3 and show that the KS test p-value for the pixel intensity distribution and the angular power spectrum are approximately equivalent for $\lesssim 15\%$ of masked area and diverge significantly for $\geq 20\%$ of masked area. This discrepancy between the obtained KS test p-values for the pixel intensity distribution and the angular power spectrum is most likely caused by the fact that the loss function we use is used to optimize the pixel values of the images.

Finally, we repeat the same analysis for the irregular masks $\mathbf{M_r}$ and $\mathbf{M_R}$ and note that the shape of the masking has no effect on the algorithm performance. In fact, as emphasized in Fig.3, the KS test p-values obtained for $\mathbf{M_r}$ and $\mathbf{M_R}$ are the same as those for the circular masks of equivalent size for all the metrics under consideration.

4 Summary & Conclusion

In this paper we have found a novel application for the deep learning algorithm, PCNN: to inpaint masked CMB maps. Inpainting CMB maps is normally done on regularly shaped masks. There have been a variety of machine learning and other statistical methods to inpaint these types of shapes (circles, squares etc.). By using PCNNs we can inpaint the CMB maps with regular and irregularly shaped masks.

We train the PCNN by simulating maps of the CMB using known cosmological parameters. Our method can reconstruct the pixel distribution and the power spectrum of a CMB patch to an accuracy respectively of $< 1\%$ and $< 5\%$, for masks covering up to $\sim 15\%$ of the image. Furthermore, for the power spectra we see that the reconstruction error is consistent with zero when the corresponding power spectrum uncertainty is taken into account. PCNNs work *equally well on regularly and irregularly shaped masks*. To quantify how well the PCNN can reconstruct the original images, we perform a KS test and find that the p-value's achieved by the PCNN based on the pixel values of the original and reconstructed images are ~ 0.999 for different mask sizes. Since in cosmology we are typically interested in the correlation function, as opposed to the amplitude of the pixel values, we also perform a KS test on the true and reconstructed power spectrum, obtaining comparable results. This shows that inpainting CMB maps with PCNNs, and computing the power spectra is a viable alternative to standard cut-sky power spectrum estimation using e.g., pseudo-C_l estimators.

The results we have obtained so far have been on a pure CMB map without any noise. In reality, any additional sources of noise and other contaminants would have to be included in the forward model used to generate the training dataset. We expect this to have no significant impact on the performance of the proposed algorithm, which is particularly suited to inpaint highly non-Gaussian patterns, as shown already in [12] on images from the ImageNet [31], Places2 [35] and CelebA-HQ dataset [36].

Overall, our results show that PCNNs can be a powerful method in reconstructing CMB maps to percent level accuracy with irregular masks, in addition to the more common circular and other regular shaped masks. It would be prudent to extend the application of PCNN architectures to inpainting irregular masks on the sphere, as we encounter when performing large sky area CMB analyses. Alongside the map-level reconstruction, we successfully recover the input power spectra from the inpainted maps. It will be interesting to explore further potential biases induced in the power spectrum estimation by the PCNN inpainting process, and studying the use of inpainted power spectrum estimates in cosmological parameter inference. We leave this and attempts to incorporate non-Gaussian foregrounds into our model for future projects.

References

[1] Y. Akrami, M. Ashdown, J. Aumont, C. Baccigalupi, M. Ballardini, A.J. Banday et al., *Planck 2018 results*, *Astronomy & Astrophysics* **641** (2020) A4.

[2] E. Hivon, K. Gorski, C. Netterfield, B. Crill, S. Prunet and F. Hansen, *Master of the cosmic microwave background anisotropy power spectrum: a fast method for statistical analysis of large and complex cosmic microwave background data sets*, *Astrophys. J.* **567** (2002) 2 [astro-ph/0105302].

[3] A. Challinor and G. Chon, *Error analysis of quadratic power spectrum estimates for CMB polarization: Sampling covariance*, *Mon. Not. Roy. Astron. Soc.* **360** (2005) 509 [astro-ph/0410097].

[4] H. Gruetjen, J. Fergusson, M. Liguori and E. Shellard, *Using inpainting to construct accurate cut-sky cmb estimators*, *Physical Review D* **95** (2017) .

[5] R.T. Pushpalwar and S.H. Bhandari, *Image inpainting approaches - a review*, in *2016 IEEE 6th International Conference on Advanced Computing (IACC)*, pp. 340–345, 2016.

[6] M. Bucher and T. Louis, *Filling in cosmic microwave background map missing data using constrained gaussian realizations*, *Monthly Notices of the Royal Astronomical Society* **424** (2012) 1694.

[7] K. Yi, Y. Guo, Y. Fan, J. Hamann and Y.G. Wang, *CosmoVAE: Variational Autoencoder for CMB Image Inpainting*, 2001.11651.

[8] G. Puglisi and X. Bai, *Inpainting Galactic Foreground Intensity and Polarization maps using Convolutional Neural Network*, 2003.13691.

[9] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair et al., *Generative Adversarial Networks*, *arXiv e-prints* (2014) arXiv:1406.2661 [1406.2661].

[10] A.V. Sadr and F. Farsian, *Inpainting via Generative Adversarial Networks for CMB data analysis*, 2004.04177.

[11] M.A. Petroff, G.E. Addison, C.L. Bennett and J.L. Weiland, *Full-sky Cosmic Microwave Background Foreground Cleaning Using Machine Learning*, 2004.11507.

[12] G. Liu, F.A. Reda, K.J. Shih, T. Wang, A. Tao and B. Catanzaro, *Image inpainting for irregular holes using partial convolutions*, *CoRR* **abs/1804.07723** (2018) [1804.07723].

[13] C. Barnes, E. Shechtman, A. Finkelstein and D.B. Goldman, *Patchmatch: A randomized correspondence algorithm for structural image editing*, *ACM Trans. Graph.* **28** (2009) .

[14] Levin, Zomet and Weiss, *Learning how to inpaint from global image statistics*, .

[15] S.M. Muddala, R. Olsson and M. Sjöström, *Spatio-temporal consistent depth-image-based rendering using layered depth image and inpainting*, *EURASIP Journal on Image and Video Processing* **2016** (2016) .

[16] Q. Fan and L. Zhang, *A novel patch matching algorithm for exemplar-based image inpainting*, *Multimedia Tools and Applications* **77** (2017) 10807.

[17] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu and T.S. Huang, *Generative image inpainting with contextual attention*, 1801.07892.

[18] S. Iizuka, E. Simo-Serra and H. Ishikawa, *Globally and locally consistent image completion*, *ACM Trans. Graph.* **36** (2017) .

[19] J. Zhao, Z. Chen, L. Zhang and X. Jin, *Unsupervised learnable sinogram inpainting network (sin) for limited angle ct reconstruction*, 1811.03911.

[20] N. Cai, Z. Su, Z. Lin, H. Wang, Z. Yang and B.W.-K. Ling, *Blind inpainting using the fully convolutional neural network*, *The Visual Computer* **33** (2015) 249.

[21] Y. Chen and H. Hu, *An improved method for semantic image inpainting with GANs: Progressive inpainting*, *Neural Processing Letters* **49** (2018) 1355.

[22] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, *Dropout: A simple way to prevent neural networks from overfitting*, *J. Mach. Learn. Res.* **15** (2014) 1929–1958.

[23] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva and A. Torralba, *Places: A 10 million image database for scene recognition*, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **40** (2018) 1452.

[24] P. Pérez, M. Gangnet and A. Blake, *Poisson image editing*, *ACM Trans. Graph.* **22** (2003) 313–318.

[25] A.W. Harley, K.G. Derpanis and I. Kokkinos, *Segmentation-aware convolutional networks using local attention masks*, `1708.04607`.

[26] Y. Hoffman and E. Ribak, *Constrained realizations of gaussian fields - a simple algorithm*, *The Astrophysical Journal* **380** (1991) L5.

[27] O. Ronneberger, P. Fischer and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, 2015.

[28] A. Lewis, A. Challinor and A. Lasenby, *Efficient computation of CMB anisotropies in closed FRW models*, *Astrophys. J.* **538** (2000) 473 [`astro-ph/9911177`].

[29] K.M. Gorski, E. Hivon, A.J. Banday, B.D. Wandelt, F.K. Hansen, M. Reinecke et al., *Healpix: A framework for high-resolution discretization and fast analysis of data distributed on the sphere*, *The Astrophysical Journal* **622** (2005) 759–771.

[30] PLANCK collaboration, *Planck 2018 results. VI. Cosmological parameters*, `1807.06209`.

[31] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, *arXiv 1409.1556* (2014) .

[32] D.P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017.

[33] M. Mustafa, D. Bard, W. Bhimji, Z. Lukić, R. Al-Rfou and J.M. Kratochvil, *Cosmogan: creating high-fidelity weak lensing convergence maps using generative adversarial networks*, *Computational Astrophysics and Cosmology* **6** (2019) .

[34] K. Aylor, M. Haq, L. Knox, Y. Hezaveh and L. Perreault-Levasseur, *Cleaning our own dust: Simulating and separating galactic dust foregrounds with neural networks*, 2019.

[35] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva and A. Torralba, *Places: A 10 million image database for scene recognition*, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **40** (2018) 1452.

[36] Z. Liu, P. Luo, X. Wang and X. Tang, *Deep learning face attributes in the wild*, in *Proceedings of International Conference on Computer Vision (ICCV)*, December, 2015.

A   Details of PCNN Algorithm

A.1   Network architecture

Module Name	Filter Size	# Channels	Stride Size	BatchNorm	Nonlinearity
PConv1	7×7	32	2	-	ReLu
PConv2	5×5	64	2	-	ReLu
PConv3	5×5	128	2	Y	ReLu
PConv4	3×3	128	2	Y	ReLu
PConv5	3×3	128	2	Y	ReLu
PConv6	3×3	128	2	Y	ReLu
PConv7	3×3	128	2	Y	ReLu
PConv8	3×3	128	2	Y	ReLu
NearestUpSample1		128	2	-	-
Concat1 (w/ PConv7)		128+128		-	-
PConv9	3×3	128	1	Y	LeakyReLu (0.2)
NearestUpSample2		128	2	-	-
Concat2 (w/ PConv6)		128+128		-	-
PConv10	3×3	128	1	Y	LeakyReLu(0.2)
NearestUpSample3		128	2	-	-
Concat3 (w/ PConv5)		128+128		-	-
PConv11	3×3	128	1	Y	LeakyReLu(0.2)
NearestUpSample4		128	2	-	-
Concat4 (w/ PConv4)		128+128		-	-
PConv12	3×3	128	1	Y	LeakyReLu(0.2)
NearestUpSample5		128	2	-	-
Concat5 (w/ PConv3)		128+128		-	-
PConv13	3×3	128	1	Y	LeakyReLu(0.2)
NearestUpSample6		128	2	-	-
Concat6 (w/ PConv2)		128+64		-	-
PConv14	3×3	64	1	Y	LeakyReLu(0.2)
NearestUpSample7		64	2	-	-
Concat7 (w/ PConv1)		64+32		-	-
PConv15	3×3	32	1	Y	LeakyReLu(0.2)
NearestUpSample8		32	2	-	-
Concat8 (w/ Input)		32+3		-	-
PConv16	3×3	3	1	Y	-

Table 1: The architecture of the PConv neural network: PConv1-8 are in encoder stage, whereas PConv9-16 are in decoder stage. The BatchNorm column indicates whether PConv is followed by a Batch Normalization layer. The Nonlinearity column shows whether and what nonlinearity layer is used. For further details see [12].

A.2 Loss Function Terms

Given ground truth image $\mathbf{I}_{gt}$, initial binary mask $\mathbf{M}$, PCNN prediction $\mathbf{I}_{out}$, we can express the L^1 losses on the output for the non-hole ($\mathcal{L}_{valid}$) and hole pixels ($\mathcal{L}_{hole}$) as:

$$\mathcal{L}_{valid} = \frac{1}{N_{\mathbf{I}_{gt}}} \|M \odot (\mathbf{I}_{out} - \mathbf{I}_{gt})\|_1 ; \qquad \mathcal{L}_{hole} = \frac{1}{N_{\mathbf{I}_{gt}}} \|(1 - M) \odot (\mathbf{I}_{out} - \mathbf{I}_{gt})\|_1 ,$$

where $N_{\mathbf{I}_{gt}} = C \cdot H \cdot W$ and C, H and W are the channel size, height and width of $\mathbf{I}_{gt}$ [12]. We then define $\mathbf{I}_{comp}$ equal to the raw output image $\mathbf{I}_{out}$ with the non-hole pixels set to ground truth. The perceptual loss ($\mathcal{L}_{perceptual}$) computes the L^1 distances between both $\mathbf{I}_{out}$, $\mathbf{I}_{gt}$ and $\mathbf{I}_{comp}$, after projecting these images into higher level feature spaces using an ImageNet-pretrained VGG-16 [31]:

$$\mathcal{L}_{perceptual} = \sum_{p=0}^{P-1} \frac{\left\|\Psi_p^{\mathbf{I}_{out}} - \Psi_p^{\mathbf{I}_{gt}}\right\|_1}{N_{\Psi_p^{I_{gt}}}} + \sum_{p=0}^{P-1} \frac{\left\|\Psi_p^{\mathbf{I}_{comp}} - \Psi_p^{\mathbf{I}_{gt}}\right\|_1}{N_{\Psi_p^{I_{gt}}}}.$$

$\Psi_p^{\mathbf{I}_*}$ is the activation map of the pth selected layer given input $\mathbf{I}_*$, with $N_{\Psi_p^{\mathbf{I}_*}}$ equal to its total number of elements. The layers used in this loss are $pool1$, $pool2$ and $pool3$ [12]. The style loss term ($\mathcal{L}_{style}$) is equivalent to the perceptual loss on which it's first performed an autocorrelation (Gram matrix) on each feature map [12].

$$\mathcal{L}_{style} = \sum_{p=0}^{P-1} \frac{1}{C_p C_p} \Big\{ \left\| K_p \left[(\Psi_p^{\mathbf{I}_{out}})^\intercal (\Psi_p^{\mathbf{I}_{out}}) - (\Psi_p^{\mathbf{I}_{gt}})^\intercal (\Psi_p^{\mathbf{I}_{gt}}) \right] \right\|_1 +$$
$$\left\| K_p \left[(\Psi_p^{\mathbf{I}_{comp}})^\intercal (\Psi_p^{\mathbf{I}_{comp}}) - (\Psi_p^{\mathbf{I}_{gt}})^\intercal (\Psi_p^{\mathbf{I}_{gt}}) \right] \right\|_1 \Big\},$$

where K_p is the normalization factor $1/C_p H_p W_p$ for the pth selected layer. The final loss term is the total variation loss ($\mathcal{L}_{tv}$) expressed as:

$$\mathcal{L}_{tv} = \sum_{(i,j)\in R,(i,j+1)\in R} \frac{\left\|\mathbf{I}_{comp}^{i,j+1} - \mathbf{I}_{comp}^{i,j}\right\|_1}{N_{\mathbf{I}_{comp}}} + \sum_{(i,j)\in R,(i+1,j)\in R} \frac{\left\|\mathbf{I}_{comp}^{i+1,j} - \mathbf{I}_{comp}^{i,j}\right\|_1}{N_{\mathbf{I}_{comp}}},$$

where R is the region of 1-pixel dilation of the hole region [12].

To conclude, as expressed in eq.2.3, the total loss ($\mathcal{L}_{total}$) is a combination of all of the above terms, according to:

$$\mathcal{L}_{total} = \mathcal{L}_{valid} + 6\mathcal{L}_{hole} + 0.05\mathcal{L}_{perceptual} + 120\mathcal{L}_{style} + 0.1\mathcal{L}_{tv}.$$

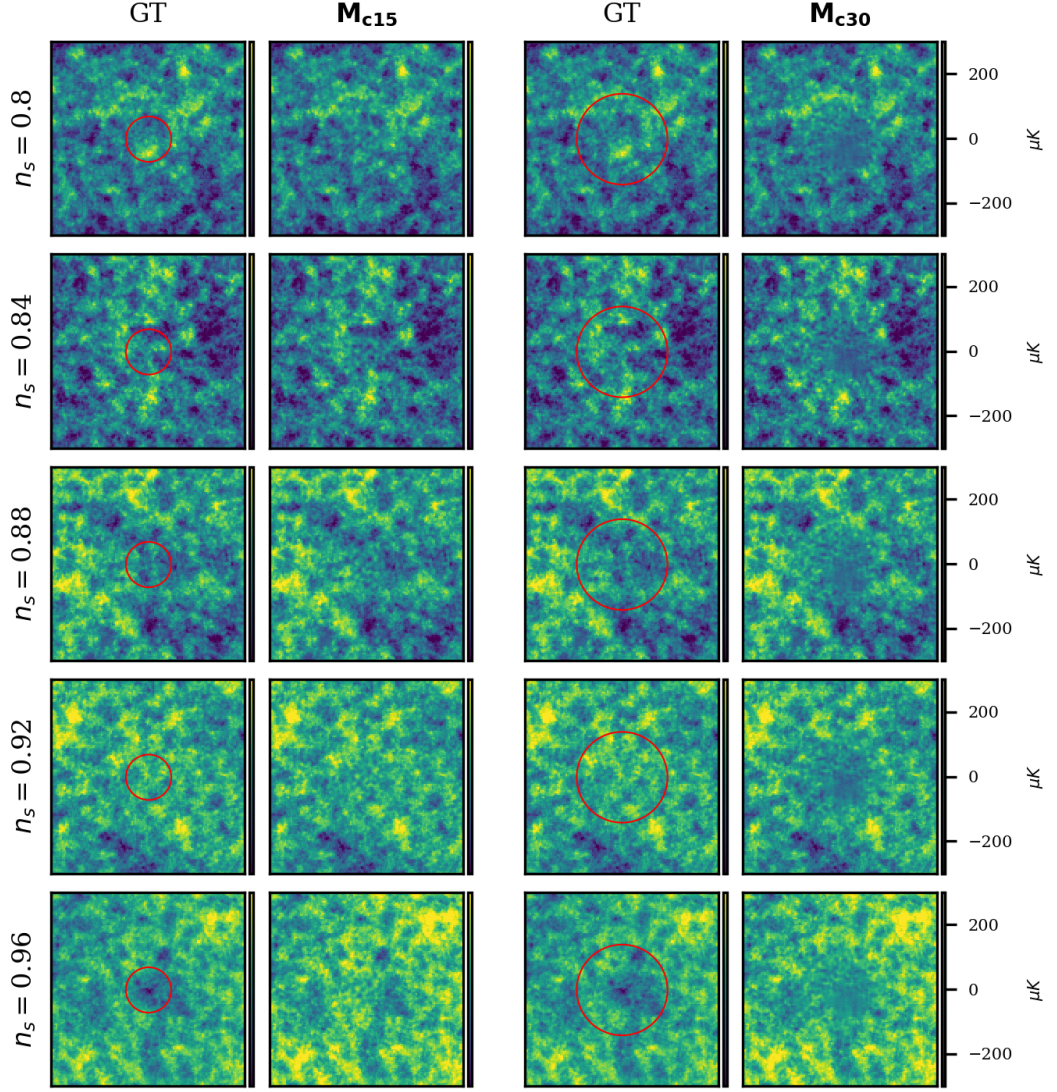were the loss term weights were determined by performing a hyperparameter search on 100 validation images [12].

A.3 PSNR and $\mathcal{L}_{total}$ evolution
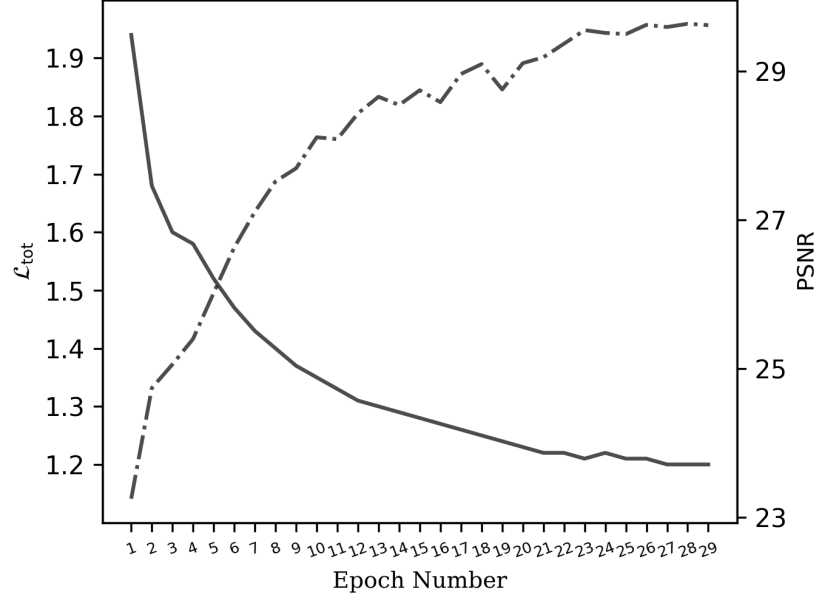
Figure 4: The plot shows the evolution of the Peak Signal to Noise Ratio - PSNR (dashed) and the total loss - $\mathcal{L}_{\text{total}}$ (solid) as a function of the epoch number.

Figure 5: The Figure shows ground truth (GT) thumbnail images and their corresponding predictions given a circular mask, respectively of radius 15 ($\mathbf{M_{c15}}$) and 30 ($\mathbf{M_{c30}}$) pixels. The masks contours are drawn in red over the GT images. From top to bottom, each row corresponds to a different value of n_s.
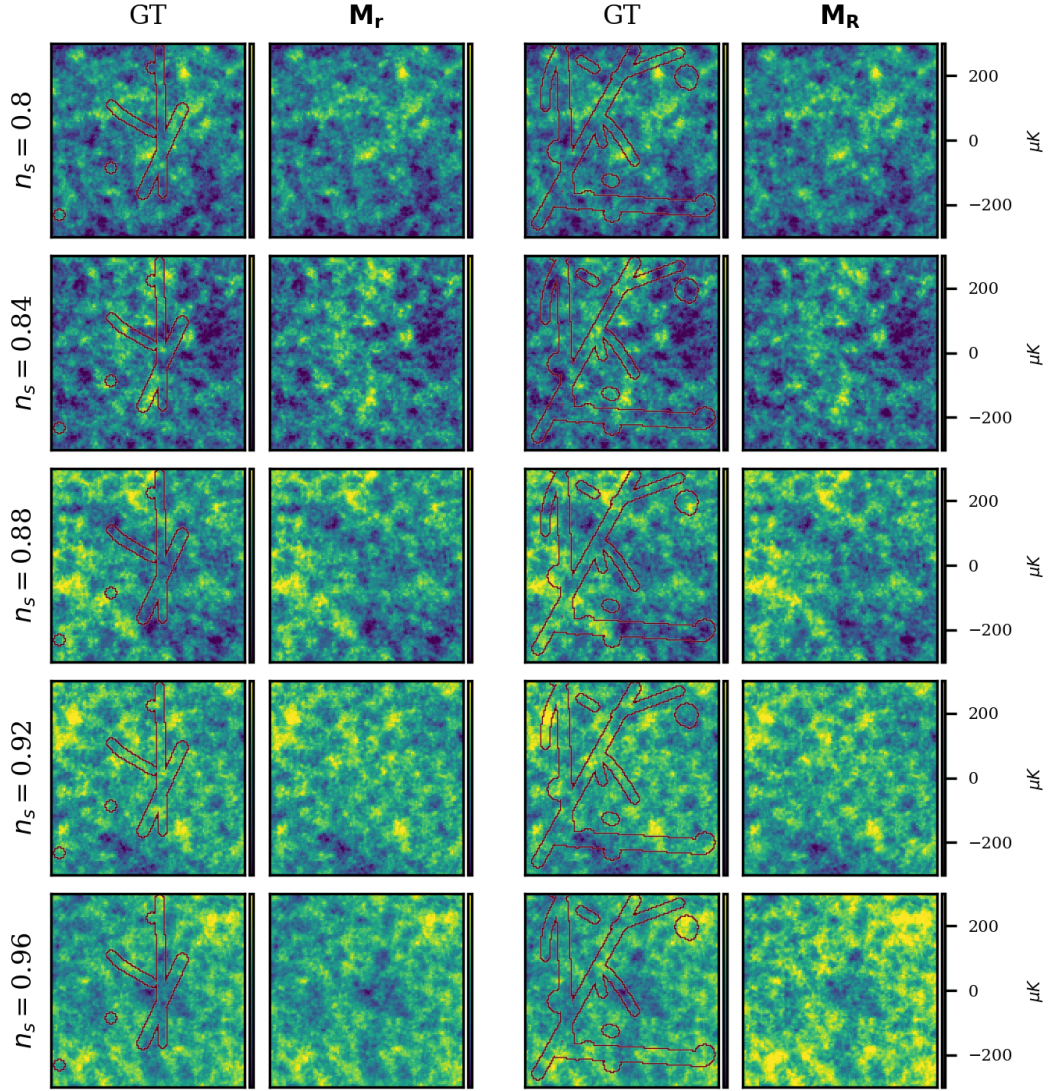
Figure 6: Similarly to Fig.5, the Figure shows ground truth (GT) thumbnail images and their corresponding predictions. In this case the masks $\mathbf{M_r}$ and $\mathbf{M_R}$ are irregular and are respectively approximately equal in size to ($\mathbf{M_{c15}}$) and ($\mathbf{M_{c30}}$). As in fig.5, the masks contours are drawn in red over the GT images and from top to bottom, each row corresponds to a different value of n_s.