Monthly Notices
of the
ROYAL ASTRONOMICAL SOCIETY

MNRAS **499,** 379–390 (2020)
Advance Access publication 2020 October 2

doi:10.1093/mnras/staa2724

Deep learning improves identification of Radio Frequency Interference

Alireza Vafaei Sadr,[1,2]★ Bruce A. Bassett,[3,4,5,6] Nadeem Oozeer ,[3,5] Yabebal Fantaye[5] and Chris Finlay[3,4,5]

[1]*Institute for Research in Fundamental Sciences (IPM), PO Box 19395-5531 Tehran, Iran*
[2]*Département de Physique Théorique and Center for Astroparticle Physics, University of Geneva, 1205 Geneva, Switzerland*
[3]*South African Radio Astronomy Observatory (SARAO), 2 Fir Street, Observatory, Cape Town 7925, South Africa*
[4]*Department of Maths and Applied Maths, University of Cape Town, Cape Town, 7700, South Africa*
[5]*African Institute for Mathematical Sciences, 6 Melrose Road, Muizenberg 7945, South Africa*
[6]*South African Astronomical Observatory, Observatory, Cape Town 7925, South Africa*

Accepted 2020 August 31. Received 2020 August 19; in original form 2020 May 29

ABSTRACT

Flagging of Radio Frequency Interference (RFI) in time–frequency visibility data is an increasingly important challenge in radio astronomy. We present R-Net, a deep convolutional ResNet architecture that significantly outperforms existing algorithms – including the default MeerKAT RFI flagger, and deep U-Net architectures – across all metrics including AUC, F1-score, and MCC. We demonstrate the robustness of this improvement on both single dish and interferometric simulations and, using transfer learning, on real data. Our R-Net model's precision is approximately 90 per cent better than the current MeerKAT flagger at 80 per cent recall and has a 35 per cent higher F1-score with no additional performance cost. We further highlight the effectiveness of transfer learning from a model initially trained on simulated MeerKAT data and fine-tuned on real, human-flagged, KAT-7 data. Despite the wide differences in the nature of the two telescope arrays, the model achieves an AUC of 0.91, while the best model without transfer learning only reaches an AUC of 0.67. We consider the use of phase information in our models but find that without calibration the phase adds almost no extra information relative to amplitude data only. Our results strongly suggest that deep learning on simulations, boosted by transfer learning on real data, will likely play a key role in the future of RFI flagging of radio astronomy data.

Key words: methods: data analysis – software: data analysis.

1 INTRODUCTION

Radio astronomy observatories are driven to the quietest regions on Earth in an attempt to escape the relentless contamination from man-made radio emission including satellites, television, radio, cell phones, and aircraft. Despite this, contamination from Radio Frequency Interference (RFI) is often orders of magnitude stronger than the astronomical signals of interest and hence must be carefully removed from data. RFI will increasingly be a limiting factor for high-quality science observations with current and planned radio telescopes, such as the MeerKAT, the South African precursor of the Square Kilometre Array (SKA; Ellingson 2005).

Despite international and local regulations to limit RFI contamination in areas where radio telescopes operate, or adopting observational strategies to avoid known RFI sources, RFI cleaning, flagging or masking procedures are needed to obtain robust science results. Exploring the best way to remove post-correlation data significantly contaminated by RFI is the subject of this paper.

Excising RFI is complicated by the fact that RFI can take an exceptionally wide range of forms, with diverse effects on different science goals such as mapping the H I spectral line as a function of redshift, continuum emission surveys, and detection of new transients. The latter is particularly affected by RFI since it can appear on a wide range of characteristic time-scales from nanoseconds to hours; for a review see Fridman & Baan (2001).

There have been numerous techniques proposed to address the excision or mitigation of RFI from observed data. These include Singular Vector Decomposition (Offringa et al. 2010), Principal Component Analysis (Zhao, Zou & Weng 2013); CUMSUM (Baan, Fridman & Millenaar 2004), SUMTHRESHOLD (Raza, Boonstra & Van der Veen 2002; Offringa et al. 2010), methods exploiting polarization information (Yatawatta 2020), and finally supervised machine learning where the algorithm learns from classified examples of RFI; e.g. Wolfaardt (2016) and Mosiane, Oozeer & Bassett (2016).

Machine leaninrg techniques have been proposed to address different problems in astronomy (Connor & van Leeuwen 2018; Harp et al. 2019; Nieto et al. 2019; Vafaei Sadr et al. 2019). This paper presents a method in the machine learning category, exploiting the powerful ability of deep neural networks – and convolutional neural networks (CNN) in particular – to learn the relevant features used to do the classification.

In the context of RFI flagging, deep learning has been used on simulated single-dish data (Akeret et al. 2017b), where a U-Net architecture delivered better results than traditional methods. Our paper extends this result by exploring new deep architectures and, in particular, now demonstrates superiority also on simulated interferometric data sets. Other studies have also looked to deep learning to help with RFI (Burd et al. 2018; Czech, Mishra &

★ E-mail: alireza.vafaeisadr@unige.ch

Inggs 2018; Sclocco, Vohl & van Nieuwpoort 2020), including in the context of modern facilities such as FAST (Yang et al. 2020) and HERA (Kerrigan et al. 2019), including the use of generative models (Vos et al. 2019).

There are two big challenges to using supervised deep learning for RFI: the first is that, we need lots of data with labels (RFI or non-RFI). It is extremely tedious for humans to accurately label many different baselines (in the case of MeerKAT, over 2000 baselines when the full array of 64 dishes is being used). In addition, different humans will disagree on the boundary of the RFI and hence we do not have access to the ground truth in the data.

On the other hand, a simulator for RFI for radio interferometers was not available, so we had to create it. This solves both the problem of the ground truth (we know what signals were put into the data) and the volume of data (we can simulate arbitrarily large amounts). However, it leads to another problem: how do we know if our wonderful model trained on simulated data, is any good on real data? Further, if it turns out to not be very good, how do we improve the model? We solve this problem by using transfer learning (Tan et al. 2018) on to real KAT-7 data that was human labelled. We find excellent results that outperform both the U-Net and traditional approaches.

The paper is organized as follows: in Section 2, we describe the simulation and data sets we use; in Section 3, we describe the applied RFI flagging algorithms and explain the details of the different CNN architectures we considered; in Section 6, we give the results and finally conclude in Section 7.

2 DATA SETS

In order to compare the performance of the convolutional neural network-based architecture against the traditional RFI flaggers, we use three data sets in our analysis: two simulated sets and one based on real astronomical observations.

2.1 Single-dish simulations

The first and simplest data set is 13 months of simulated single-dish data using the HIDE & SEEK package (Akeret et al. 2017a) matching the characteristics of the Bleien Observatory 276 frequency channels with 1 MHz bandwidth between 990 and 1260 MHz. We used similar experimental set-up and RFI simulation parameters to those in Akeret et al. (2017b). The training, validation, and test sets consisted of 11 months, 1 month and 1 month, respectively. A single day of simulated HIDE data can be seen in the top panel of Fig. 6 along with the ground truth mask in the second panel.

2.2 MeerKAT array simulations

The second data set simulates the MeerKAT telescope array that consists of 64 antennas, each 13.5 m in diameter. These 64 antennas lead to 2016 independent baselines. The array has a dense core with 48 antennas located within a diameter of 1 km, with the remaining 16 spread out, giving a maximum baseline length of 8 km.

Due to data size and computation limitations, we chose 15 baselines to give the optimal trade-offs between time, frequency, and baseline coverage in our data. The antennas were chosen so the baseline lengths covered scales from 100 m to 8 km and sampled all directions roughly equally. We used a MeerKAT bandwidth of 856 MHz centred at 1284 MHz and channelized at 208.984 kHz into 4096 channels. The MeerKAT feed is linearly polarized and hence measures the horizontal (H) and vertical component (V) of the electromagnetic radiation.

The MeerKAT simulations include a simplified (Airy disc beam for the HH and VV components) full-polarization primary beam model that was fitted to a Zernike polynomial representation from Asad et al. (2019). Neither pointing errors nor gain variations of the primary beam over time were included. Such gain variations could be considered as being factored into the bandpass time variation. A full-polarization bandpass was generated from measurements of a calibrator source with the real MeerKAT array. Time dependence of the bandpass as a whole was introduced through randomly sampling Fourier modes with periods in the range of 10 min to several hours and designed to mimic real data. The amplitude of the bandpass was kept within 10 per cent of the original amplitude.

The astronomical source model (positions, fluxes, and source shapes) used for all MeerKAT RFI simulations comes from a combination of the SUMSS and NVSS catalogues. Spectral indices are not available from these catalogues and were therefore randomly sampled from $\mathcal{N}(-0.9, 0.3^2)$.

The sources of RFI present in the simulations are satellites present in the *L*-band and the nearest five towns to the telescope site. The RFI originating from the five towns is simulated as RFI sources sitting on the horizon. RFI entering the beam from the horizon was treated like any other source where its beam attenuation is determined by its angle from the pointing centre. Multipath effects of RFI were not taken into consideration. Each RFI source was randomly assigned a frequency range in which it was emitting. The distribution of these frequency ranges was taken such as to mimic the average RFI distribution of historical MeerKAT data. Each RFI source was then assigned a random spectral profile consisting of between 2 and 5 Sersic profiled peaks. The RFI considered in these simulations originate from man-made sources and therefore tend to be polarized. Each RFI source was randomly assigned a specific polarization (Q, U, or V) and was treated as being fully polarized.

For a radio interferometer like MeerKAT (with respect to its observing frequency and spatial distribution of antennas), many sources of RFI lie in the near field. In these simulations, near-field effects have not been taken into consideration. The open source framework (Montblanc) used to perform the RIME calculations only allows one the addition of far-field sources.

The time dependence of each RFI source was again introduced through randomly sampling Fourier modes except with periods in the range of seconds. To obtain realistic satellite paths, the Two-Line Element sets (TLEs) for each satellite were used to predict their positions for a given time and date (Vallado & Cefola 2012). The output of the simulations is the complex-valued visibilities for the astronomical component, $\mathcal{A}$, and the RFI component, $\mathcal{R}$, separately. This was done so that contaminated visibilities could be generated through a linear combination of these components and additive noise after the fact to allow for augmenting the training data, as described below.

We simulated 100 samples, which each contain 800 s of observations for all four vertical (V) and horizontal (H) cross-polarizations (HH, VV, HV, and VH). We then averaged the data along the time axis in 8-s bins. The resulting data files have visibilities as a function of the following variables (Time, Baseline, Frequency, Polarization) and of shape (100, 15, 4096, 4). Each sample is 15 GB so the whole data set is 1.5 TB in size. We split the 100 files randomly into training, validation, and test set in the ratio: 40:20:40.

Despite the relatively large size of our simulated data (1.5 TB)[1], we would like more time duration in the data. To address this problem, we

[1] Made up of the 15 baselines, 8 s time sampling, four polarizations, and 4096 frequency channels.

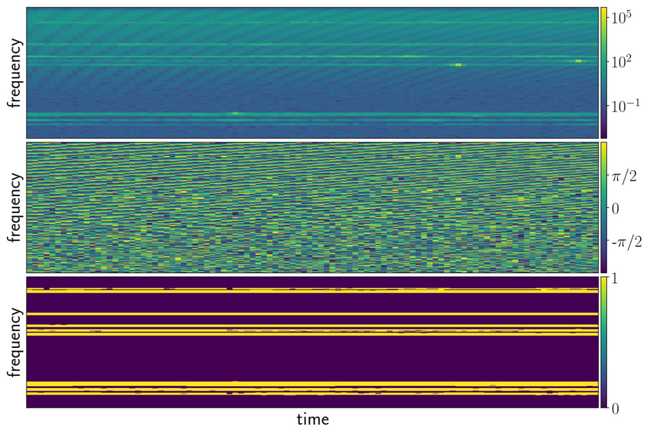

Figure 1. Example of an augmented MeerKAT simulation for a single baseline with $(\alpha, \sigma) = (0.1, 0.3)$. The first row is a map of absolute values of the visibility data for time versus frequency. The second row is the phase versus time and frequency (wrapped between $=\pi$ and $-\pi$) while the lowest panel is the ground truth mask for $\tau_5 = 5 \times 0.168$ Jy.

simulated the astronomical, RFI, and noise signals separately. This allows us to efficiently augment our data by randomly combining astronomical, RFI, and noise data as follows:

$$\mathcal{D} = \mathcal{A} + \alpha\mathcal{R} + \mathcal{N}. \qquad (1)$$

Here, $\mathcal{A}$ is data for the astronomical source information, $\mathcal{R}$ is a template RFI contaminated data, $\mathcal{N}$ is noise generated from a random complex Gaussian with standard deviation randomly chosen from a uniform distribution in the range from 0.168 to 0.336 Jy. 0.168 Jy is the theoretical noise for MeerKAT in a single baseline and time–frequency bin calculated from the measured SEFD of the instrument. This range therefore gives a realistic noise range for MeerKAT. The parameter α controls the amplitude of the RFI contribution and was randomly chosen in the range from 2^{-10} to 1 in logarithmic bins. The resulting cumulative distribution of RFI amplitudes is shown in Fig. 2.

Due to disc space limitations, augmentation is done on the fly in memory during training and validation. The test set is augmented twice (leading to 80×800 s samples) and saved to disc. Because the test set is prepared only once and saved, we use the same test data for all algorithms ensuring that the details of the augmentation do not affect the comparisons. An example of an augmented MeerKAT simulation sample (800 s, 4096 frequencies) is shown in Fig. 1.

2.3 Real KAT-7 observations

Our third data set comes from 12 h of actual observations from the Karoo Array Telescope (KAT-7). KAT-7 was an engineering prototype for the MeerKAT telescope array, but lead to its own scientific contributions (Foley et al. 2016).

The data includes all seven antennas that lead to 42 baselines observed in the summer of 2016. More detailed information about the data set is available in Heald et al. (2016). The RFI in the KAT-7 data are manually flagged by experts and we used these manually obtained RFI masks as ground truth. We note that the hand flagged mask will of course not be perfect but is the best approximation to the truth that we have for real data.

The purpose of using the KAT-7 data is to show that, given a relatively small amount of human-flagged data, our algorithm is able to transfer from simulations to real data with high precision, delivering results at a speed comparable to or better than AOFLAGGER and similar non-deep learning algorithms. This is an important step since otherwise one would have to rely on simulated RFI data being highly realistic. Even in this case, there would be a nagging concern that elements of the full telescope not captured in the simulations might lead to significant degradation of the performance of the RFI flagger when applied to real data.

2.4 Ground truth maps

The goal of RFI flagging and excision is to remove all pixels that are contaminated by RFI beyond a threshold determined by the desired image quality and goals of the science: hence we address this as a classification problem and output a binary mask.[2]

As we use supervised learning algorithms for pixel classification, we need ground truth binary maps, $\mathcal{M}$, to learn from. In contrast, the Science Data Processing (SDP) Online Flagger does not need ground truth data to run out of the box, although we do use the ground truth data to optimize the SDP Flagger.

Since the RFI amplitude varies continuously over the pixels, we create binary ground truth maps by choosing an RFI threshold τ_n. For this purpose, we used $\tau_n = n\sigma$, where σ is the signal to noise ratio and $n = 1$ or 5, is an integer which controls the RFI amplitude, $\mathcal{R}$, needed to classify a pixel *ij* as either clean or contaminated:

$$\mathcal{M}_{ij} = \begin{cases} 1 & \mathcal{R}_{ij} > \tau_n \\ 0 & \mathcal{R}_{ij} \leq \tau_n. \end{cases} \qquad (2)$$

Making τ_n too large results in too few masked pixels which are very easy to find. On the other hand, for very small τ_n, most pixels are masked and again we end up with a highly unbalanced data set. Further, because of the large noise level, there is little signal to learn from.

We consider two values: $\tau_1 = 1\sigma$ and $\tau_5 = 5\sigma$. Since this value changes for each data set, the thresholds τ_n will also vary from data set to data set even for fixed n. We choose 5σ for our main results. Results for 1σ case are shown in the Appendices but are qualitatively similar.

The RFI distribution for both HIDE and MeerKAT simulations are shown in Fig. 2. For the HIDE simulations, $\sigma = 0.670$ Jy while for MeerKAT simulations $\sigma = 0.169$ Jy. The $\tau_5 = 5\sigma$ threshold on average masks 41 per cent of pixels in the HIDE simulations and 16 per cent of the pixels in the MeerKAT simulations. Choosing $\tau_1 = 1\sigma$, in contrast, leads to 46 per cent and 17 per cent of pixels being masked in the HIDE and MeerKAT simulations on average.

3 NEW DEEP RFI ALGORITHM: R-NET

Almost all supervised learning problems are optimizations of some very flexible and non-linear model to return the desired output. The optimization objective is normally a loss function that captures how close the model is to the desired output. Neural networks, deep learning, and specifically CNN show very promising results in different problems among other machine learning algorithms.

In this study, we use a CNN-based model to do a binary classification of every pixel of the time–frequency images given the pixel and those around it. The output is a binary mask image of the same

[2]Finding the optimal threshold is specific to the science under consideration. One type of science may require high purity, another may prefer to have more data but with higher levels of RFI contamination. Determining this would require a full optimization based on the underlying science; see e.g. Bassett (2005) and Parkinson et al. (2007).

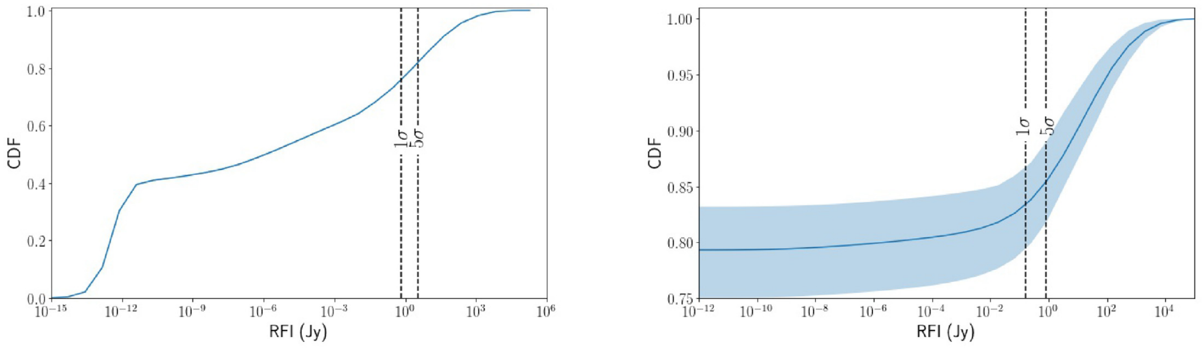

Figure 2. Average Cumulative Distribution Functions (CDF) of the daily averaged RFI amplitude for the HIDE (left) and MeerKAT (right) simulations. The 1σ and 5σ thresholds are shown. We choose 5σ as our primary threshold, leading to 41 per cent and 16 per cent of pixels being masked as RFI pixels in the HIDE and MeerKAT simulations, respectively. The shaded areas show 95 per cent confidence levels for both plots. The HIDE simulations show very small variation in the CDF resulting in very narrow error bars.

size. This is similar to segmentation problem in computer science (Badrinarayanan, Handa & Cipolla 2015; Ronneberger, Fischer & Brox 2015; Badrinarayanan, Kendall & Cipolla 2017; Zhang et al. 2017; Alom et al. 2018).

Our proposed network, which we call R-Net, is inspired by the residual network architecture (ResNet) (see He et al. 2016; Szegedy et al. 2017). Deeper neural networks can be significantly more difficult to train (e.g. due to the vanishing gradient problem, see Hochreiter et al. 2001). The ResNet was introduced to ease the training process of the deeper layers by adding shortcuts between some layers (e.g. up to 152 layers). We use one such shortcut to build our architecture, constructed using sequences of 2D convolutional layers without any reduction in the size of the input. For computational reasons, we did not consider more than six layers and hence did not need more than one skip/shortcut.

R-Net is very simple in terms of its architecture since it consists only in convolutional layers with zero paddings to conserve the size of the image as it moves through the layers. After each convolutional layer (except the last one), we find that inserting batch normalization layer and using RELU activation functions to be optimal. The other hyperparameters of such an architecture are the number of layers, kernel size, number of filters, location of shortcut(s), and activation function. Searching the hyperparameter space is very time consuming and limited by available computing resources: our exploration of the potential hyperparameters was guided by intuition and trial and error on the validation set.

All of the hidden layers contain 12 filters and the kernel size is always 5×5. R-Net does not change the input size through layers so any number of hidden layers works; we try 3, 5, 6, and 7 number of layers. A shortcut (residual network) connection directly transfers information from earlier layers to the deeper layers to avoid the degradation problem (He et al. 2016). We used no shortcut connections for the 3-layer architecture, one for the 5 and 6 layers architectures and two shortcut connections for the 7-layer architecture. Our initial results show the performance of the 5 and 6 layers architectures (hereafter R-Net5 and R-Net6) provide the best performance better in terms of Area Under the Curve (AUC).

The output layer is always a 1×1 convolution layer that aggregates all 12 filters into a one-channel image. The output images are normalized between 0 and 1 and, roughly speaking, each pixel value shows the probability of being RFI contaminated. It is possible to turn the output into a binary mask by applying a threshold and compared to the ground truth whose pixels are either 0 or 1 representing clean and RFI-contaminated pixels, respectively. We try mean square error (MSE) and binary cross entropy as loss functions and find that MSE performs better. This suggests reconsidering RFI problem as a regression problem: an idea we present in a future work.

4 COMPARISON OF ALGORITHMS

To fully evaluate the performance of R-Net, we compare it against two existing RFI algorithms: the deep U-Net algorithm and the standard SDP Flagger.

4.1 U-Net

U-Net is a CNN-based approach (see Ronneberger et al. 2015) first used for RFI flagging by Akeret et al. (2017b) in the context of single-dish experiments. It includes convolutional layers that are usually used in image classification problems since they build a conceptual hierarchy that can extract different kinds of features through down sampling. The U-Net architecture first contracts and then expands the input images back to the original size using deconvolutional (see Zeiler et al. 2010) and upsampling layers. Roughly speaking, the bottleneck shape, like autoencoders, are supposed to force the CNN to learn the most important features and avoid overfitting.

The U-Net architecture has been used for RFI detection in single-dish simulations (Akeret et al. 2017b), where it showed improved performance over a standard non-machine learning algorithm. You can see a schematic illustration of both the U-Net and R-Net architectures in Fig. 3. We use both U-Net and SDP Flagger to compare with R-Net.

4.2 The MeerKAT SDP Online Flagger

The SDP Online Flagger[3] (hereafter called the SDP Flagger) is the default RFI flagger used by the MeerKAT SDP pipeline. It is based on the classic AOFLAGGER,[4] a C+ + RFI flagging algorithm that uses morphological features (Offringa et al. 2010). AOFLAGGER runs on a 2D data array of time and frequency visibilities to flag and mask RFI signals and has been used in a number of interferometric telescope arrays, including LOFAR, WSRT, VLA, GMRT, ATCA,

[3] https://github.com/ska-sa/katsdpsigproc
[4] https://sourceforge.net/p/aoflagger/wiki/Home/

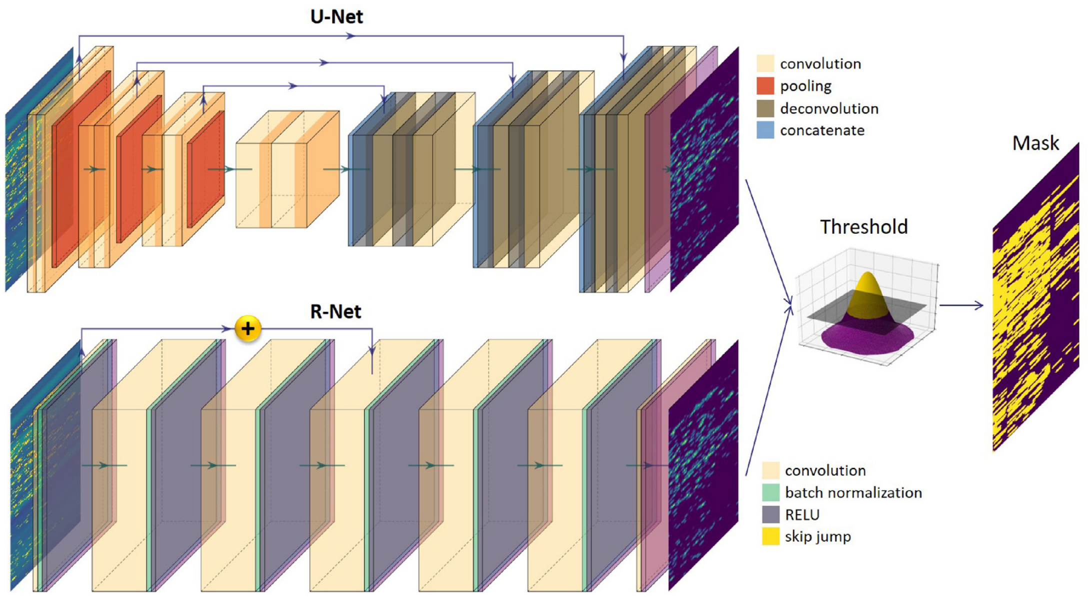

Figure 3. Schematic views of the U-Net architecture (top) and our new R-Net algorithm (bottom). Both algorithms predict RFI probabilities for each pixel in the map. The final binary mask is produced by passing the output maps through a thresholding process that is one of the hyperparameters for the algorithms.

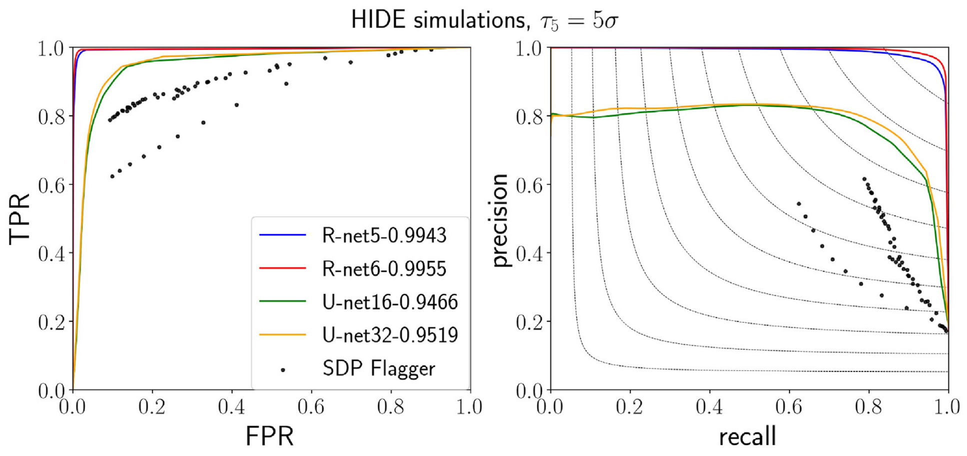

Figure 4. Performance on single-dish simulations – R-Net is almost perfect and significantly outperforms both U-Net and the SDP Flagger when trained on single-dishHIDE simulations when predicting the RFI mask at an RFI threshold of 5σ. The legend shows corresponding AUC values for R-Net and U-Net. Dots are the SDP Flagger results using different hyperparameters. The SDP Flagger performs poorly for all values on the precision–recall plot. Contours denote iso-F1 scores. The plots for 1σ RFI threshold are shown in the Appendix, and are qualitatively similar.

and MWA, and single-dish telescopes such as the Parkes and the Arecibo telescopes.

In order to make the comparison to R-Net as fair as possible, we optimize SDP Flagger performance by varying its various hyperparameters. Due to computational limits, and after experimentation, we choose to vary *outlier-n-sigma*, σ_O, and *background reject*, R_b, as the most significant hyperparameters. In Appendix A, we show in Fig (A1) the performance of SDP Flagger when τ_n, σ_O, and R_b vary.

5 TRAINING

For all simulations, the data sets were split into training, validation, and test sets. We use the validation set for hyperparameter optimization and the results are reported on the test set.

Since R-Net and U-Net architectures are fully convolutional, it is possible to train the models on smaller frequency–time images and use the final model on larger data sets. In fact, this is one advantage of using neural networks: they are expandable to run on GPU. For training window size, we used all frequencies and limited the size of time steps to W_t, which is treated as a hyperparameter: large W_t leads to memory overload while small W_t leads to poor performance. Therefore, we maximize W_t given memory constraints.

We use the cross-entropy loss function for the U-Net architecture and MSE for R-Net. We used the RMSPropOptimizer with the learning rate initiated to 0.2 and decreasing exponentially/linearly for U-Net/R-Net and the weights are initiated using the truncated normal distribution.

One notable point on training R-Net is that for the MSE, loss function models can easily be trapped in a local minimum and predict a constant value for all pixels. This problem is more dramatic when the data labels are unbalanced, as happens in the MeerKAT simulations. We call this the 'dead model' problem. To address this

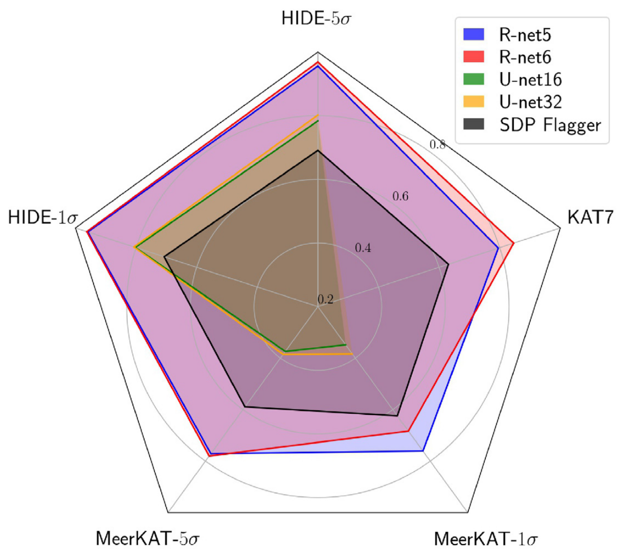

Figure 5. Starplot comparison between F1 scores of all the algorithms for all the data sets and different thresholds (e.g. HIDE-5σ corresponds to the results where the threshold is τ_5). The centre corresponds to a score of zero and the outer contour to a perfect score of one. R-Net outperforms all other algorithm variants as it encloses all other algorithms on all data sets and threshold. Note that U-Net was not included for the transfer learning task on the KAT-7 data set.

issue, we use two recommended techniques in the NGENE package.[5] The first is model resuscitation where the model gets checked frequently during the training process. The constant value returned in dead models, is usually close to the average of the labels. To check this condition, one can use the output's standard deviation at a certain frequency while the model is training. Model resuscitation can be simply done by reinitiation of the weights, which suffices for our purposes.

The second issue is overfitting, where a model easily learns noise and focuses on irrelevant patterns as information. The problem gets worse when the number of trainable parameters is large. There are plenty of different proposals like data augmentation, regularization, dropouts, etc. to address this problem.

The core ML models are flexible and contain trainable parameters, increasing the chance that the model overfits. We exploit this tendency to overfit to initialize the model and train the neural networks given only a few data points and allow the CNN model overfit. This usually lets the neural network use the overfitted weights as good initial values for the whole training set.

R-Net was developed using *Ngene*.[6] Both U-Net and R-Net are trained on NVIDIA Tesla P100 12GB GPUs until the average loss does not improve for 50 epochs.

6 RESULTS

To compare R-Net against the other two reference algorithms (U-Net and SDP Flagger), we consider several metrics. The first is the AUC, corresponding to the Area Under the True Positive Rate (TPR)–False Positive Rate (FPR) curve (Bradley 1997), which is usually a good metric for binary classification problems.

However, since the data sets we consider can be significantly unbalanced (with few RFI-dominated pixels), the usual classification metrics may give misleading results. We therefore also compute the precision–recall curves, F1-score, and Matthews Correlation Coefficient (MCC) which are better metrics for unbalanced data; see e.g. Davis & Goadrich (2006). For more details, see Appendix (B) or Chicco & Jurman (2020). F1-scores are shown in Table 1. The 5 and 6-layer versions of R-Net outperform all other variants of U-Net and SDP Flagger significantly in all metrics. Note that it is often in the precision–recall plots that R-Net significantly outperforms the other algorithms.

For all data sets, we varied the SDP Flagger hyperparameters $\sigma_{\rm O}$ and $R_{\rm b}$. The results are shown as the point cloud in all figures. The effect of $\sigma_{\rm O}$ and $R_{\rm b}$ on fall-out, recall, precision are shown in Fig. A1.

A summary of the performance of all the algorithms on all the data sets and all thresholds is shown in Fig. 5 in terms of the F1-score. While this is only one metric, it does illustrate the power of R-Net. We now discuss performance of the algorithms on each data set separately.

6.1 HIDE simulations

We split the 13 months of data into 11 months for training, 1 month for validation, and 1 month for the test set. We chose $W_{\rm t} = 400$ (the width of the time window for training).

The best SDP Flagger F1-score results for 5σ and 1σ thresholds were 0.69 and 0.71. The best result for the U-Net architecture was achieved by using 32 features in the latent layer and achieved (AUC,F1-score) = (0.95,0.80) for 5σ and (AUC,F1-score) = (0.94,0.81) for the 1σ threshold.

For the R-Net architecture the best results for $\tau_5 \equiv 5\sigma$ threshold is (AUC,F1-score) = (0.99,0.97) and for $\tau_1 = 1\sigma$ is (AUC,F1-score) = (0.99,0.96). Using 6 layers results in slightly better results over 5 layers. R-Net significantly outperforms both SDP Flagger and U-Net and returns almost perfect results. A sample of one day simulated data together with results from the various algorithms is shown in Fig. 6. The Receiver Operating Characteristic (ROC) and precision–recall curves for the HIDE simulations are shown in Fig. 4 and C1.

6.2 MeerKAT simulations

The MeerKAT data set consists of 100 files with 800 s of simulated data. 40 files are augmented in memory and used for training, while 20(40) files are augmented three times for validation (test) sets. We averaged the time axis every 8 s and chose $W_{\rm t} = 50$ for the width of the time window.

The best SDP Flagger F1-scores were 0.69 and 0.71 for the 5σ and 1σ thresholds. The best result of the U-Net architectures was achieved for 32 features in the latent layer achieving (AUC,F1-score) = (0.62,0.39) for 5σ and (AUC,F1-score) = (0.59,0.38) for a 1σ threshold.

For the R-Net architecture using only the amplitude of the visibilities, the best result was (AUC,F1-score) =(0.90,0.77) and (AUC,F1-score) = (0.87,0.76) for the 5σ and 1σ thresholds, respectively, with slightly better results from the 5-layer architecture. R-Net significantly outperforms all SDP Flagger and U-Net configurations with respect to the investigated metrics. TPR–FPR and precision–recall curves for the MeerKAT simulations are shown in Fig. 7 and C2 for the two different thresholds.

We also investigated adding phase information in addition to the amplitude of the visibilities as one of the channels of the input layer

[5]https://github.com/vafaei-ar/Ngene
[6]https://github.com/vafaei-ar/RFI-mitigation

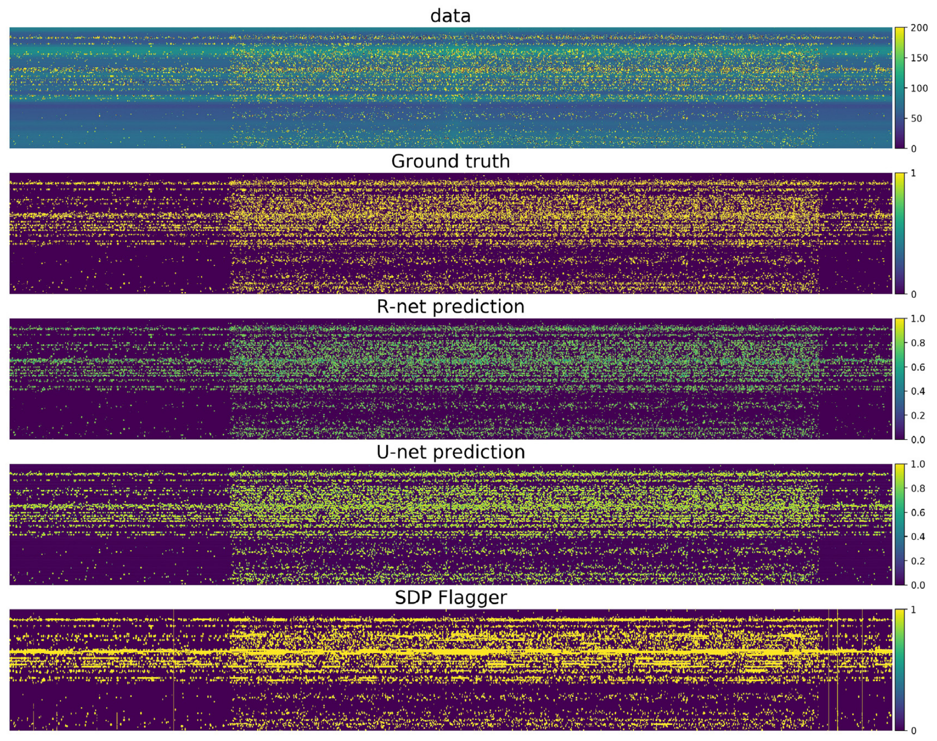

Figure 6. One day of HIDE simulations along with the results of the various flaggers. The first row shows the observed data, clipped between 0 and 200. The second, third, and fourth rows show ground truth, R-Net, and U-Net probability outputs. The final row is the output of SDP Flagger. The axes are time and frequency similar to Fig. 1.

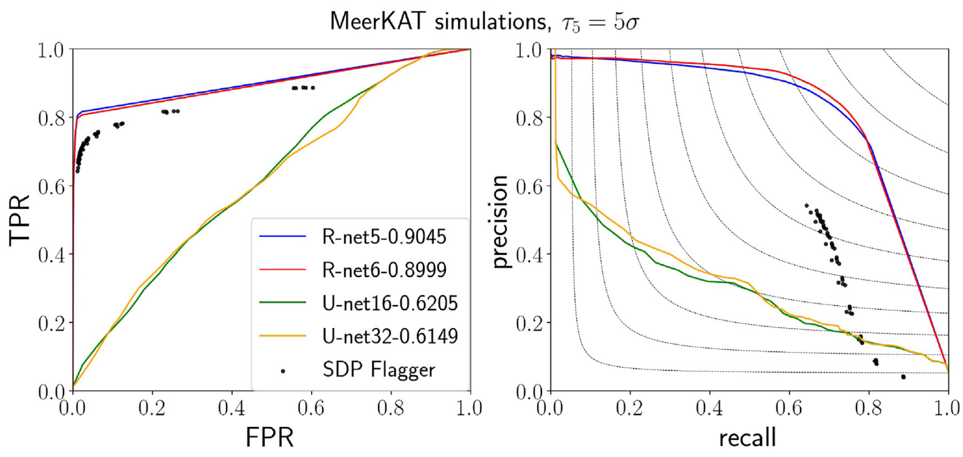

Figure 7. Performance on MeerKAT simulations – R-Net significantly outperforms both U-Net and the SDP Flagger trained on MeerKAT simulations using only the absolute value of the visibility data (RFI threshold of 5σ). The legend shows corresponding AUC values for R-Net and U-Net. Note the significant drop in performance of all algorithms relative to their corresponding results on the much simpler HIDE data. Dots show the SDP Flagger results using different hyperparameters. Although the SDP Flagger does relatively well on the TPR–FPR curve, it performs poorly for all values on the precision–recall plot. Contours denote iso-F1 scores. The equivalent plots for 1σ RFI threshold are shown in the Appendix, and are qualitatively similar. In Fig. C3, we show the equivalent plot with phase information included, demonstrating that uncalibrated phase information adds no additional significant value to R-Net.

of the similar architecture and the results do not significantly improve the performance of the algorithm, as shown in Fig. C3. This is likely due to the fact that the phase is not calibrated.

6.3 Transfer learning for KAT-7

We have so far only considered the performance of the algorithms on simulated data. The concern with this is that the resulting 'best' algorithm might actually perform badly on real telescope data due to the deficiencies in the simulated data.

This could be circumvented if one had very large amounts of accurately labelled real data by simply training an R-Net model from scratch. However, large amounts of human-labelled, real data are not available (especially for large arrays with many baselines). Further, human labels are imperfect, especially in regions where the RFI contamination is fairly week, and therefore will anyway contain errors.

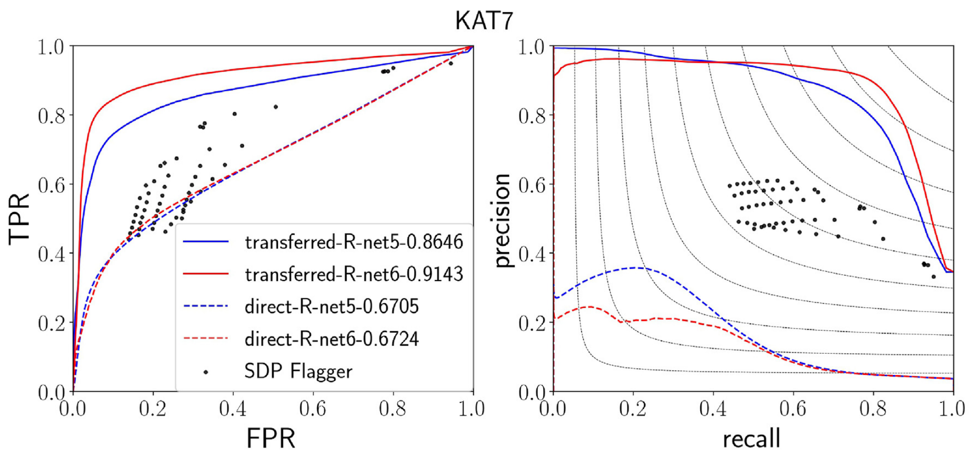

Figure 8. Transfer learning from MeerKAT simulations to real KAT-7 data – R-Net performance on the KAT-7 hand-flagged data set using the best MeerKAT model (dashed lines; 'direct') and the same models after transfer learning with a small amount of real KAT-7 data (solid lines; 'transferred'). Transfer learning massively improves the performance of the model. Results are averaged over four polarizations and are averages of four-fold cross validation. Points indicate the various SDP Flagger results for different hyperparameter combinations of ($\sigma_{\rm O}$, $R_{\rm b}$) values. Note that at a recall of about 0.8, the best transferred R-Net model has approximately double the precision of the best SDP Flagger model. Note also that there were no parameter values for the SDP Flagger that had an FPR <0.1, whereas this can be tuned with the deep learning models. Contours denote iso-F1 scores.

Table 1. F1-scores for the various different algorithms and data sets considered in this paper. The best results for each data set are shown in bold. The R-Net models outperform all the U-Net and SDP Flagger models by a considerable margin.

Algorithm	HIDE-5σ	HIDE-1σ	MeerKAT-5σ	MeerKAT-1σ	KAT7
R-net5	0.96	**0.96**	0.77	**0.76**	0.79
R-net6	**0.97**	**0.96**	**0.78**	0.68	**0.85**
U-net16	0.78	0.80	0.37	0.35	–
U-net32	0.80	0.81	0.39	0.38	–
SDP Flagger	0.69	0.71	0.59	0.62	0.63

To deal with this catch-22 problem, we investigate the use of transfer learning applied to the R-Net model trained on the MeerKAT data. *Transfer learning* (Tan et al. 2018) in this context means we freeze the weights in all of the layers before the shortcut leaving only the weights in the last two layers to be trained. Freezing the lower layers means we retain the key ability of the network to recognize relevant RFI features while leaving only relatively few parameters that need to be learned to adapt to the real data. As a result, we only need a fairly small amount of real data, which also bounds the error introduced by the imperfections in the human labelling.

In this case, we applied transfer learning to the human-labelled KAT-7 data described earlier. For our results, we use average performance on four-fold cross validation. A comparison of SDP Flagger and R-Net with transfer learning is shown in Fig. 8. R-Net with transfer learning on the KAT-7 data actually achieves a slightly higher AUC score than on the MeerKAT simulations (0.91 versus 0.90). Even more impressive, the best MeerKAT-trained model only achieved an AUC of 0.67 when applied directly on the KAT-7 data (dashed lines in Fig. 8), showing that the transfer learning had a massive impact. To illustrate how little data was used in the transfer learning, the best R-Net model trained from scratch on the KAT-7 data alone was only able to achieve an AUC of 0.55. While performance between the 5 and 6-layer R-Net models was almost indistinguishable in earlier tests, the 6-layer R-Net model performed significantly better in the transfer learning task, perhaps showing that the extra flexibility provided by the extra layer could be exploited efficiently.

A summary of the results in terms of F1-score and TPR–FPR curves are shown in Table 1 and Fig. 5. Both R-Net configurations do very well.

As a result, we conclude that transfer learning is a very promising approach to fine-tuning algorithms trained on simulated data. Given that the KAT-7 data are very different in frequency and angular resolution to MeerKAT, the success of transfer learning here also shows that it can also be applied to fine-tuning algorithms trained on data from a different telescope.

6.4 Performance comparison

An algorithm that is accurate but extremely slow to evaluate on new data is of limited use in future real world applications such as the SKA, which will be dominated by huge incoming data rates. We therefore also evaluated the computational cost of running each trained algorithm on test data; with the results shown in Table 2. The comparison is performed between different algorithms in terms of the number of trained variables in the case of R-Net and U-Net, execution time, and number of FLOPs measured on a 3.3GB test data set. One can see the number of trainable variables for R-Net is less than U-Net by a factor ≈10. As a result, there is a higher chance of overfitting in U-Net models. The number of FLOPs required by R-Net is larger than SDP Flagger and U-Net but despite this fact, the R-Net execution time is actually slightly smaller than the other methods because the R-Net architecture contains convolutional layers that are all the same size and hence operations are more efficiently handled in parallel.

Table 2. Number of trainable variables, FLOPs, and execution time for the different algorithms. The CNN-based algorithms used a Tesla P100, 12GB GPU and SDP Flagger used 6 CPU cores.

Model	Variables	exe. time(s)	FLOPs
SDP Flagger	–	31	6.8T
R-Net-5	11 473	18	8.3T
R-Net-6	15 181	24	11.9T
Unet-3-16	116 770	49	4.0G
Unet-3-32	465 986	53	4.0G

7 CONCLUSION

We describe a new ResNet-style convolutional neural network algorithm for RFI flagging. We test this algorithm on both single-dish and realistic interferometric telescope array RFI simulations showing that it significantly outperforms the current state-of-the-art algorithms (both U-Net and the modified version of AOFLAGGER currently used in the MeerKAT data reduction pipeline).

The default R-Net algorithm was trained on the magnitudes of the complex visibilities. We did also explore the use of the phase to aid in RFI flagging but found that uncalibrated phase information did not lead to any additional improvements over the magnitudes alone, whether they were included by splitting the visibilities into real and imaginary components or into magnitude and phase.

Finally, we show that models trained on simulated MeerKAT data can be very efficiently fine-tuned via transfer learning to provide state-of-the-art results on real data flagged by humans from a very different telescope; in this case, the KAT-7 array. Transfer learning involves retraining the last few layers of the existing model on the real data. This process boosted average performance from an AUC of 67 per cent to 91 per cent. This success suggests that transfer learning will be a powerful and exciting tool for RFI flagging as we move into the era of the SKA.

ACKNOWLEDGEMENTS

The authors thank Michelle Lochner and Ethan Roberts for insightful discussions and comments and the anonymous reviewer for useful comments. AVS thanks Golshan Ejlali and Anke Van Dyk for useful discussions. We thank the members of the SARAO Data Science and SDP teams and the RFI Working Group for discussions. CF thanks Ben Hugo, Simon Perkins, and other members of the SARAO RARG team for useful discussions around radio interferometry simulations. The numerical computations were carried out on SARAO facilities, the Baobab cluster at University of Geneva and the CHPC. YF was supported by the Robert Bosch Stiftung through the AIMS-ARETE chair position. This research has been conducted using resources provided by the United Kingdom Science and Technology Facilities Council (UK STFC) through the Newton Fund and the South African Radio Astronomy Observatory (SARAO). The financial assistance of the SARAO is hereby acknowledged.

DATA AVAILABILITY

The data used in this paper are available upon request. You can send an email to alireza.vafaeisadr@unige.ch to get access.

REFERENCES

Akeret J., Seehars S., Chang C., Monstein C., Amara A., Refregier A., 2017a, Astron. Comput., 18, 8
Akeret J., Chang C., Lucchi A., Refregier A., 2017b, Astron. Comput., 18, 35
Alom M. Z., Hasan M., Yakopcic C., Taha T. M., Asari V. K., 2018, preprint (arXiv:1802.06955)
Asad K., Girard J., de Villers M., Lehmensiek R., Ansah-Narh T., Iheanetu K., Smirnov O., Santos M., et al., 2019, preprint (arXiv:1904.07155)
Baan W., Fridman P., Millenaar R., 2004, AJ, 128, 933
Badrinarayanan V., Kendall A., Cipolla R., 2017, IEEE transactions on pattern analysis and machine intelligence, 39, 2481
Badrinarayanan V., Kendall A., Cipolla R., 2017, IEEE Trans. Pattern Anal. Mach. Intell., 39, 2481
Bassett B. A., 2005, Phys. Rev. D, 71, 083517
Bradley A. P., 1997, Pattern Recognition, 30, 1145
Burd P. R., Mannheim K., März T., Ringholz J., Kappes A., Kadler M., 2018, Astron. Nachr., 339, 358
Chicco D., Jurman G., 2020, BMC Genomics, 21, 6
Connor L., van Leeuwen J., 2018, AJ, 156, 256
Czech D., Mishra A., Inggs M., 2018, Astron. Comput., 25, 52
Davis J., Goadrich M., 2006, Proceedings of the 23rd international conference on Machine learning. Association for Computing Machinery, New York, USA, p. 233
Ellingson S. W., 2005, The Square Kilometre Array: An Engineering Perspective. Springer-Verlag, Berlin, p. 261
Foley A. et al., 2016, MNRAS, 460, 1664
Fridman P., Baan W., 2001, A&A, 378, 327
Harp G., Richards J., Tarter S., Mackintosh G., Scargle J., Henze C., Nelson B., Cox G. et al., 2019, preprint (arXiv:1902.02426)
He K., Zhang X., Ren S., Sun J., 2016, in Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, IEEE, New Jersey, United States, p. 770
Heald G. et al., 2016, MNRAS, 462, 1238
Hochreiter S., Bengio Y., Frasconi P., Schmidhuber J., 2001, Gradient Flow in Recurrent Nets: The Difficulty of Learning Long-Term Dependencies. A field guide to dynamical recurrent neural networks. IEEE Press, USA
Kerrigan J. et al., 2019, MNRAS, 488, 2605
Matthews B. W., 1975, Biochim. Biophys. Acta, 405, 442
Mosiane O., Oozeer N., Bassett B. A., 2016, in 2016 IEEE Radio and Antenna Days of the Indian Ocean (RADIO) conference, Réunion Island. International Union of Radio Science, Rome, Italy, p. 1
Nieto D., Brill A., Feng Q., Humensky T., Kim B., Miener T., Mukherjee R., Sevilla J., 2019, preprint (arXiv:1912.09877)
Offringa A., De Bruyn A., Biehl M., Zaroubi S., Bernardi G., Pandey V., 2010, MNRAS, 405, 155
Parkinson D., Blake C., Kunz M., Bassett B. A., Nichol R. C., Glazebrook K., 2007, MNRAS, 377, 185
Raza J., Boonstra A.-J., Van der Veen A.-J., 2002, IEEE Signal Process. Lett., 9, 64
Ronneberger O., Fischer P., Brox T., 2015, Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016 (19th International Conference, Athens, Greece, 2016, Proceedings). Springe, New York, United States, p. 234
Sclocco A., Vohl D., van Nieuwpoort R. V., 2019, RFI Workshop 2019-Coexisting with Radio Frequency Interference (RFI) . IEEE, New York, USA, p. 1
Szegedy C., Ioffe S., Vanhoucke V., Alemi A. A., 2017, in Markovitch S., Singh S., eds., Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence. AAAI, California, United States, p. 12
Tan C., Sun F., Kong T., Zhang W., Yang C., Liu C., 2018, in Kurková V., Manolopoulos Y., Hammer B., Iliadis L., Maglogiannis I., eds., International conference on artificial neural networks 2018 proceeding. Springer, New York, United States, p. 270
Vafaei Sadr A., Vos E. E., Bassett B. A., Hosenie Z., Oozeer N., Lochner M., 2019, MNRAS, 484, 2793

Vallado D. A., Cefola P. J., 2012, 63rd International Astronautical Congress 2012 (IAC 2012) proceeding. International Astronautical Federation (IAF), Paris, France
Vos E., Francois Luus P., Finlay C., Bassett B., 2019, in IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP). IEEE , New York, USA
Wolfaardt C. J., 2016, PhD thesis. Univ. Stellenbosch
Yang Z., Yu C., Xiao J., Zhang B., 2020, MNRAS, 492, 1421
Yatawatta S., 2020, preprint (arXiv:2006.00062)
Zeiler M. D., Krishnan D., Taylor G. W., Fergus R., 2010, in Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. IEEE, New York, USA, p. 2528
Zhang Q., Cui Z., Niu X., Geng S., Qiao Y., 2017, in Liu D., Xie S., Li Y., Zhao D., El-Alfy E., eds, Springer, New York, U SA, p. 364
Zhao J., Zou X., Weng F., 2013, IEEE Trans. Geosci. Remote Sens., 51, 4830

APPENDIX A: SDP FLAGGER OPTIMIZATION RESULTS

To provide fair comparison to R-Net and U-Net, we explore how the SDP Flagger results (FPR, recall, precision) are affected by varying the hyperparameters. We run the SDP Flagger on HIDE and MeerKAT simulations varying three the hyperparameters (τ_n, $\sigma_{\rm O}$, $R_{\rm b}$); namely *RFI threshold*, *outlier-n-sigma*, and *background reject*, respectively. The first controls how the user chooses to define the ground truth mask, the second and third are SDP Flagger hyperparameters. Each single dot in the figures shows performance on one simulated file (one day in HIDE simulations and 800 s for MeerKAT simulations). We show all results as dots in the figures and choose colour to show the data set. Fig. A1 combines all results in one figure. The results on the MeerKAT simulations are more sensitive to hyperparameter choice.

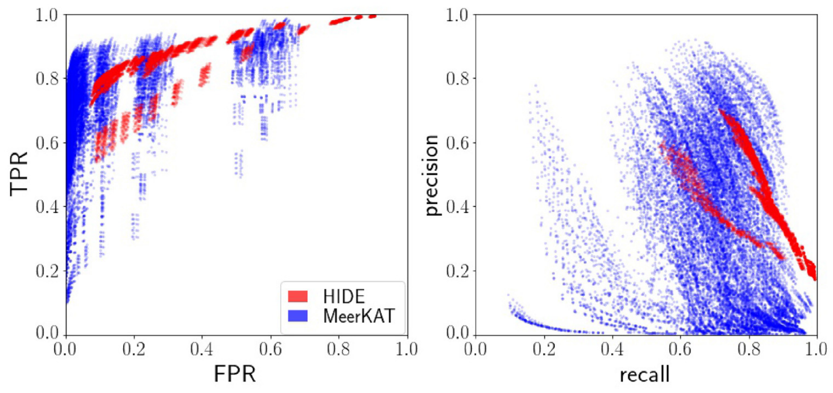

Figure A1. The SDP Flagger performance in terms of TPR, FPR, recall, and precision on both the HIDE (red dots; evaluated on 1 day of data) and MeerKAT simulations (blue dots; evaluated on 800 s of data) as the hyperparameters, $R_{\rm b}$ (background reject), $\sigma_{\rm O}$ (outlier-n-sigma), and τ_n (RFI threshold) are varied widely. There are no combinations that simultaneously lead to both excellent precision, recall, and FPR.

APPENDIX B: METRICS

This appendix gives definitions about the used metrics for who needs to see them in more details. TPR also called sensitivity or recall is

$$\text{Recall} = \frac{TP}{TP + FP}, \quad \text{(B1)}$$

where TP (true positive) is the number of pixels that are truly predicted as positive and FP (false positive) is the number of pixels that are mistakenly predicted as positive pixels. The FPR, also known as fallout, is

$$\text{FPR} = \frac{FP}{FP + TN}, \quad \text{(B2)}$$

where TN (true negative) is the number of pixels that are truly predicted as negative pixels. Then one is able to define the ROC curve as TPR plotted against FPR as one changes the classification threshold. The AUC metric is the integral of the ROC curve.

The MCC is a number between −1 (worst case) and 1 (perfect case) and is defined by Matthews (1975):

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{(TP + FP)(TN + FN)(TP + FN)(TN + FP)}, \quad \text{(B3)}$$

where TN and FN are the True Negative and False Negative ratios. It is particularly suited to unbalanced data.

The precision is defined by

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{(B4)}$$

which then allows one to calculate the F1-score using:

$$F1\text{-score} = \frac{2}{\text{Precision}^{-1} + \text{Recall}^{-1}}, \quad \text{(B5)}$$

which mixes the precision and recall metrics.

APPENDIX C: ADDITIONAL RESULTS

In this appendix, we show the results for the HIDE and MeerKAT data for a threshold of $\tau_1 \equiv 1\sigma$ in Fig. C1 and C2. In addition, in Fig. C3, we show the effect of adding uncalibrated phase information as a new channel to the visibility magnitude maps on the performance of R-Net and U-Net. Comparison with Fig. 7 shows that very little is gained in terms of AUC for R-Net.

Finally, in the starplots C4 and C5, we compare the performance of all the algorithms on all the data sets for both combinations of threshold, τ_1 and τ_5.

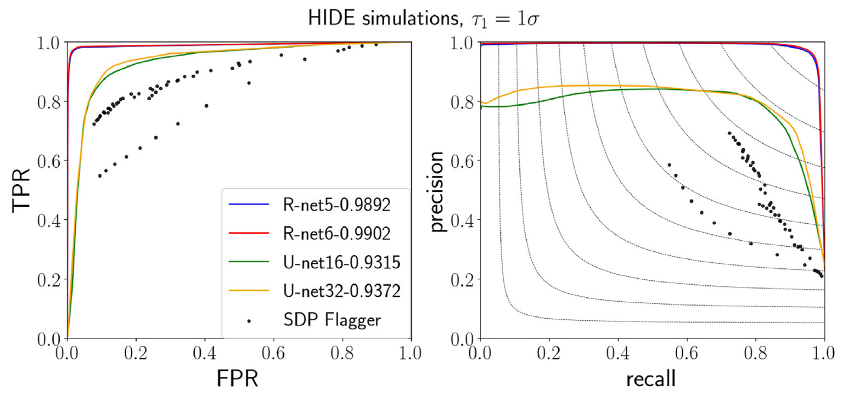

Figure C1. Comparison between R-Net and U-Net performance trained on HIDE simulations to predict for the threshold $\tau_1 = 1\sigma$. Dots show the SDP Flagger results found varying the different hyperparameters. R-Net outperforms the other algorithms also for this threshold.

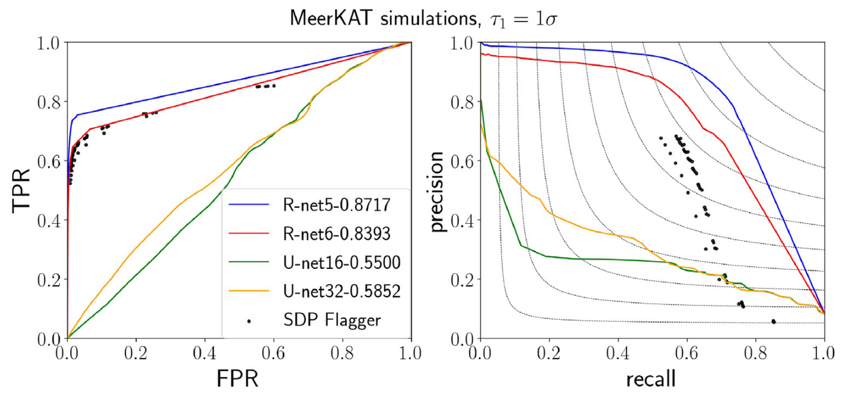

Figure C2. Comparison between R-Net and U-Net performance trained on MeerKAT simulations using only absolute value of the data to predict the threshold $\tau_1 = 1\sigma$. Dots show the SDP Flagger results using different hyperparameters values. R-Net outperforms the other algorithms also for this threshold.

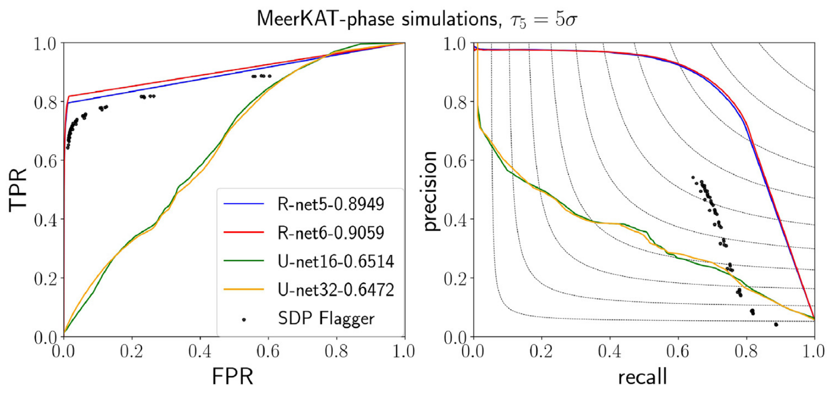

Figure C3. Adding phase information yields no improvement – R-Net and U-Net performance trained on MeerKAT simulations with amplitude and phase of the data to predict threshold = 5σ. Dots are SDP Flagger results varying σ_O and R_b. This should be compared with Fig. 7 which shows almost identical performance for R-Net, though modest improvement for U-Net.

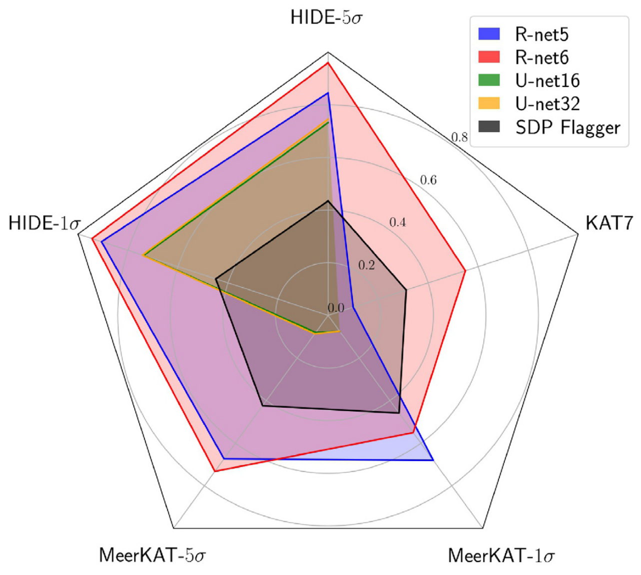

Figure C4. Starplot comparison between the MCC scores of all the algorithms for all the data sets and different thresholds (e.g. HIDE-5σ corresponds to the HIDE results where the threshold is τ_5).

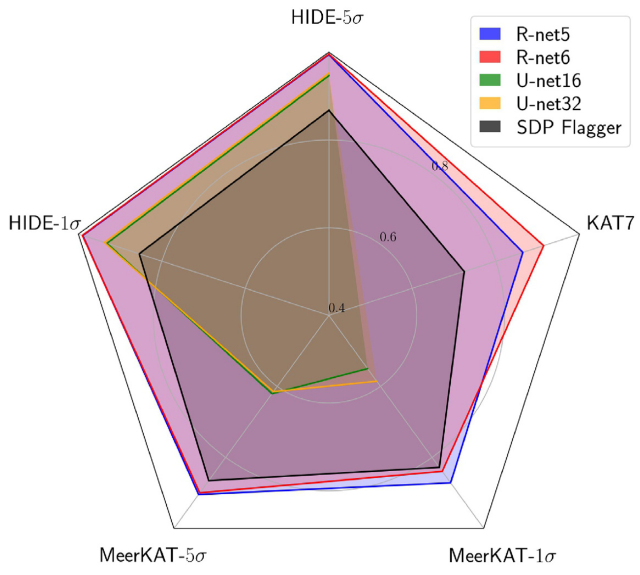

Figure C5. Starplot comparison between AUC scores of all the algorithms for all the data sets and different thresholds (e.g. HIDE-5σ corresponds to the results where the threshold is τ_5). R-Net outperforms all other algorithm variants. The SDP Flagger AUC is estimated by interpolating points in the TPR–FPR plane.