Draft version January 27, 2022
Typeset using LaTeX **preprint2** style in AASTeX63

Radio Frequency Interference Classification using Convolutional Neural Networks

A. Suresh,[1] R. J. Hill,[1] E. S. Bair,[1] J. M. Cordes,[1] and S. Chatterjee[1]

[1] *Cornell Center for Astrophysics and Planetary Science and Department of Astronomy, Cornell University, Ithaca, NY 14853, USA*

ABSTRACT

Detection of fast radio bursts (FRBs) in dynamic spectra is often complicated by the presence of radio frequency interference (RFI) and radiometer noise. With the large data volumes output by commensal FRB surveys and their high candidate rates, machine learning algorithms for burst detection and signal classification are fast becoming a necessity. Convolutional neural networks (CNNs) for FRB detection have recently been implemented by Zhang et al. (2018), Connor & van Leeuwen (2018), and Agarwal et al. (2019). However, these studies do not distinguish between different types of RFI and FRBs based on their spectro-temporal morphology in the dynamic spectrum. Here, we experiment with toy models of varying depths to classify input dynamic spectra as either noise, short-lived broadband RFI, short-lived narrowband RFI, long-lived narrowband RFI, or long-lived broadband RFI.

1. INTRODUCTION

Fast radio bursts (FRBs) are bright (typical peak flux density, $S_\nu \sim 10$ mJy $-$ 100 Jy), millisecond duration flashes of coherent radio emission originating beyond the Milky Way. To date, over a hundred FRBs (Petroff et al. 2016)[1] have been detected, of which 11 (Spitler et al. 2016; CHIME/FRB Collaboration et al. 2019a,b; Kumar et al. 2019) have been seen to repeat. Using sub-arcsecond angular resolution achieved with radio interferometers, two repeating (Chatterjee et al. 2017; Marcote et al. 2020) and three apparently non-repeating FRBs (Bannister et al. 2019; Ravi et al. 2019; Prochaska et al. 2019) have been localized to their respective host galaxies. The varied specific star formation rates, metallicities and interstellar medium (ISM) magnetizations of these galaxies suggest that FRB progenitors may reside in diverse local environments.

Searches for FRBs typically involve gathering of data at high temporal and spectral resolution. As radio pulses propagate through the ISM of galaxies and the intergalactic medium (IGM), they suffer a dispersive delay (t_{DM}). As a function of radio frequency (ν), the analytical form of t_{DM} is:

$$t_{DM} = 4.15 \text{ ms } \frac{DM_{\text{pc cm}^{-3}}}{\nu^2_{\text{GHz}}}. \quad (1)$$

Here, the dispersion measure (DM) is the electron column density through which the pulse has propagated. Since the DM of an FRB is unknown prior to detection, corrections for t_{DM} are usually applied by dedispersing data in the frequency-time plane (dynamic spectrum) at a

Corresponding author: Akshay Suresh
as3655@cornell.edu

[1] http://www.frbcat.org

large number of trial DMs. These dedispersed dynamic spectra are then summed over frequency to produce dedispersed time series. The dedispersed time series are convolved with boxcar functions of various widths to identify single pulses hidden in the data (Cordes & McLaughlin 2003). Finally, candidates above a certain detection threshold are labeled for visual inspection by a human.

While this manual detection process has served FRB astronomy well over the past two decades, the recent advent of state-of-the-art dedispersion algorithms and Graphics Processing Unit (GPU)-acccelerated pipelines have enabled real-time dedispersion during commensal FRB surveys. The large data rates ($\gtrsim$ 1 TB/hr) from these surveys coupled with their high candidate rates ($\sim$ 1000 events day^{-1}) necessitate an automated approach for FRB detection. Furthermore, the presence of radio frequency interference (RFI) and radiometer noise can render manual inspection of voluminous data products difficult. To alleviate the sheer number of candidates marked for visual inspection, a number of RFI-mitigation techniques (e.g.: excision of DM = 0 pc cm^{-3} candidates, Cordes & McLaughlin (2003)) and clustering algorithms (k-means clustering, Friends-of-Friends) have been previously implemented. However, these techniques do not classify individual detections as FRBs or RFI.

The use of convolutional neural networks (CNNs) for automated signal classification has, therefore, gained significance for FRB detection. Over the last two years, Zhang et al. (2018), Connor & van Leeuwen (2018), and Agarwal et al. (2019) have implemented deep learning algorithms to distinguish FRBs from RFI in dynamic spectra. While Zhang et al. (2018) operate using non-dedispersed dynamic spectra, Connor & van Leeuwen (2018) and Agarwal et al. (2019) pass dedispersed dynamic spectra as inputs to their networks. All three models output the probabilities of a signal detection being an FRB or an RFI. Considering the diverse morphology that RFI can exhibit in the dynamic spectrum, such models preclude detailed characterization of the RFI environment around a telescope.

In this work, starting from scratch, we attempt to build a CNN that differentiates between dispersed pulses and various kinds (short-lived or long-lived, narrowband or broadband) of RFI based on their structure in the non-dedispersed dynamic spectrum. Section 2 describes the general architecture and basic components of a neural network. In Sec. 3, we detail the construction of fake data sets used to train our network. Section 4 discusses our experimentation with CNNs of varying depths. Finally, in Sec. 5, we review our learnings from this exploration and outline planned future developments.

2. NEURAL NETWORKS

A neural network is comprised of several interconnected layers of neurons, with zero or more "hidden" layers sandwiched between an input layer and an output layer. Based on the concept of a perceptron (Rosenblatt 1958), each neuron in a network attempts to learn a function relationship between its inputs ($\mathbf{x}$) and its outputs ($\mathbf{y}$). Mathematically, the operation of a neuron is:

$$\mathbf{y} = f(\mathbf{w} \cdot \mathbf{x} + b). \tag{2}$$

Here, $\mathbf{w}$, b and f are, respectively, the weights, the bias, and the non-linear activation function used by the neuron. Commonly used activation functions include the rectified linear unit (also called ReLU or ramp, $f(x) = \max(0, x)$), the sigmoid ($f(x) = 1/(1 + e^{-x})$), and the tanh function. For classification problems, the softmax activation function is often employed in the

penultimate layer of neural networks to output probabilities. Given inputs $\mathbf{x} = (\mathrm{x_j})_{\mathrm{j=1,2,..,N}}$, the output of a softmax activation function, $\mathcal{S}(\mathbf{x})$, is:

$$\mathcal{S}(\mathrm{x_j}) = \frac{\mathrm{e^{x_j}}}{\sum_{\mathrm{i=1}}^{\mathrm{N}} \mathrm{e^{x_i}}}, \qquad \mathrm{j} = 1, 2, .., \mathrm{N}. \tag{3}$$

A single pass of $\mathbf{x}$ into a neuron to obtain $\mathbf{y}$ is called forward propagation. Training of a neuron involves the modification of $\mathbf{w}$ and b to ensure that $\mathbf{y}$ matches or approximates the expected output ($\widehat{\mathbf{y}}$). During the process of training, a neural network computes a loss function Q at the end of forward propagation into its output layer. Q measures the departure of the predicted output of a neural network from its expected value for the training data. The choice of Q in a network is decided by its application. While regression problems typically use the mean squared error loss (MSE: $\mathrm{Q}(\mathbf{y}, \widehat{\mathbf{y}}) = ||\mathbf{y} - \widehat{\mathbf{y}}||^2 = \sum_{\mathrm{j=1}}^{\mathrm{N}} (\mathrm{y_j} - \widehat{\mathrm{y}}_\mathrm{j})^2$), the cross-entropy loss (CE, Murphy (2012)) with a softmax activation function is preferred for feature classification.

Once a loss function Q is chosen, the weights, $\mathbf{w}$, of each neuron are updated in accordance with a gradient descent applied to Q. This step is called backpropagation. In other words, the backpropagation algorithm updates a weight w_j to its new value:

$$w_j' = w_j - \eta \frac{\partial \mathrm{Q}}{\partial w_j}, \tag{4}$$

where η is called the learning rate. η is a measure of the sensitivity of changes in w_j to gradients of Q. Ideally, a small value of η is preferable to ensure convergence of the backpropagation algorithm to a value of w_j that is sufficiently close to the global minimum of Q. Typically, accelerated convergence of backpropagation is achieved using optimizers such as the Stochastic Gradient Descent (SGD, Bottou (2010)) and the Adaptive Moments estimation (Adam, Kingma & Ba (2014)).

2.1. *Learning Paradigms*

There exist three main neural network learning paradigms: supervised, unsupervised, and hybrid. In supervised learning, a network is supplied training data that carry "correct" labels for each input. During training, the network weights are modified so that the network predictions agree as closely as possible with the supplied labels. Unsupervised learning (e.g., clustering algorithms) does not require a label associated with each input in the training data set. Instead, the network explores underlying patterns in the data and organizes these patterns into categories. Hybrid learning combines supervised and unsupervised learning techniques. Here, the network weights are determined partly via supervised learning and partly through unsupervised methods.

2.2. *Supervised Training of Neural Networks*

Labeled data for supervised learning are generally divided into three sets: training, validation, and test data. A neural network is initially trained to adjust its weights and minimize its loss function on the training data. The validation data are then used to test the performance of the trained network. If the network performs poorly on the validation data, it is retrained on the training data until it achieves satisfactory performance on the validation data. The weights of the neural network are then held fixed and are applied to a forward propagation of the test data for obtaining results. Note that the validation data and the test data are never used to train the network.

A single run of the training data through the forward propagation and backpropagation steps constitutes a training epoch. Owing to the large number of parameters present in a typical neural network, its training requires several epochs

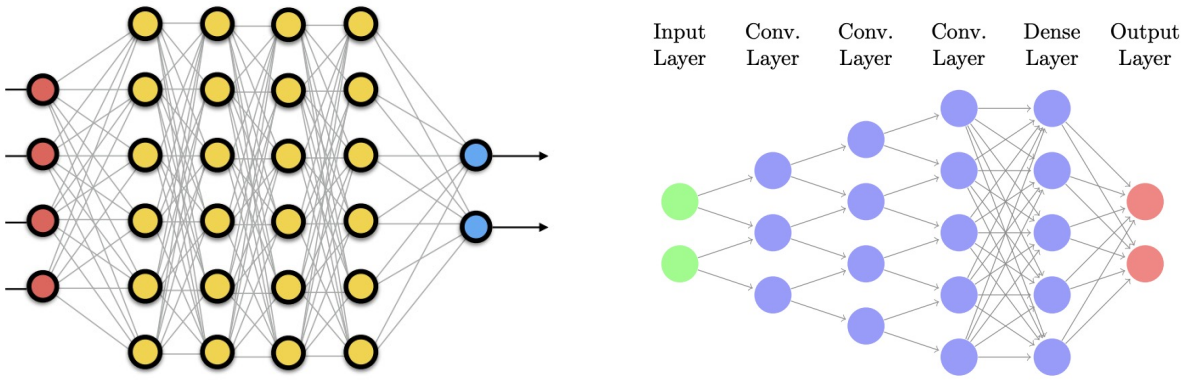

Figure 1. Left panel: A conventional neural network with four dense (yellow) layers hidden between an input layer (red) and an output layer (blue) (Connor & van Leeuwen 2018). Right panel: A convolutional neural network comprising of three convolutional layers and one dense layer sandwiched between an input layer and an output layer (Agarwal et al. 2019). In both panels, the circles represent neurons, while the lines/arrows indicate connections between neurons.

and hardly converges. Instead, the training is halted when a certain accuracy threshold is achieved on the training data.

When training a neural network, one encounters three different scenarios: underfitting, overfitting, and a robust fit. If the performance of a network is poor on both the training data and the validation data, the neural network is said to "underfit" these data. The network then needs to be passed through additional epochs of training. Overfitting arises when a neural net attains a high degree of accuracy on the training data, but shows poor performance on the validation data. In this scenario, the network has memorized the specific patterns of the training data rather than learning to recognize the general trends. Overfitting can be avoided by gathering more training data, supplying a balanced training data set, or reducing the network length. Finally, a network is said to achieve a robust fit if its performance on the training data and the validation data are equally good.

2.3. *Convolutional neural networks*

In a conventional neural network (left panel of Fig. 1), every neuron in one layer is connected to all neurons in the preceding and succeeding layers. As a result, the total number of trainable parameters in the network increases exponentially with the number of layers. This causes the forward and backward propagation algorithms to become computationally intensive. A CNN overrides this difficulty using the practical assumption that nearby elements in an input image are correlated. This diminishes the need for every neuron in a given layer to be connected to all neurons in adjacent layers of the network. Layers with connections between every pair of input and output neurons are termed as dense or fully-connected layers.

Figure 1 compares the architecture of a general neural network (left) with that of a CNN (right). While all hidden layers in a regular neural network are dense layers, a CNN simplifies computations using multiple convolutional layers. A convolutional layer is comprised of several small filters (or kernels) that get convolved with the input. The convolution operation is usually followed by an activation function and a "pooling" layer. In the pooling layer, the outputs from the convolutional layer are downsampled by either taking the maximum (max-pool) or average (mean-pool) of a small rectangle of

pixels. This enables feature detection in a manner invariant to scale and orientation changes[2].

2.3.1. *Output sizes of a convolutional layer*

One major challenge when designing a CNN is to correctly compute the output sizes from each convolutional layer. For the purposes of this discussion, let us assume a 2D input image of size $W \times W$. This image is processed by a convolutional layer comprising of N filters of size $F \times F$. In literature, N is called the depth of the convolutional layer. Let us further assume that we zero-pad the input image with P pixels on each side. We move the filter by stride S along each dimension when performing the convolution. The output of the convolutional layer is now three-dimensional with the third dimension being the depth (N) of the layer. With stride S, filter size F and zero-padding P, the length (or width) of the output is:

$$L = \frac{W - F + 2P}{S} + 1, \quad (5)$$
$$\implies \text{Output shape} = L \times L \times N.$$

Note that if $P = 0$ and $S = 1$, $L \leq W$. Furthermore, if $S > 1$, $L < W$ and the input data are downsampled by the convolution. Typically, convolutional layers have $S = 1$, while pooling layers use $S > 1$.

2.3.2. *Naming Convention*

A neural network is termed an N−layer network[3] if it contains N layers excluding its input layer. A 1−layer network, hence, contains zero hidden layers between its input and output layers.

3. GENERATION OF TRAINING DATA

The use of a balanced training data set is essential for attaining optimal CNN performance on test data. Owing to the rarity of FRBs and the prevalence of RFI in real-world dynamic spectra, large balanced sets of telescope data are difficult to build. Furthermore, manual labeling of voluminous data for supervised learning can be tedious. To alleviate this situation, we simulate fake dynamic spectra for constructing our training data sets.

Empirically, CNNs are found to work best on input frames with zero mean and unit variance. So, we simulate noisy dynamic spectra by drawing samples randomly from a standard normal distribution. The number of time samples and the number of spectral channels contained in a dynamic spectrum frame are dictated by the input sizes of our networks. Traditionally, FRB searches are performed at L-band ($1 - 2$ GHz) with sub-millisecond time sampling. We, therefore, adopt a time resolution, $t_{\rm samp} = 64$ μs for our data. In addition, we define the frequency span of our simulated data to be the spectral coverage ($1.15 - 1.73$ GHz) of the Arecibo L-wide receiver[4].

Both RFI and FRBs (Hessels et al. 2019) are known to exhibit complex time-frequency structure in the dynamic spectrum. For the sake for simplicity, we model these signals as 2D Gaussians with DM = 0 pc cm^{-3} for RFI and non-zero DMs for FRBs (or dispersed pulses). We begin RFI simulation by randomly initializing a seed pixel in the dynamic spectrum. Let ν_0 and t_0 denote, respectively, the seed frequency and the seed time. Centered at (ν_0, t_0), we construct a 2D Gaussian flux density profile of the

[2] https://adventuresinmachinelearning.com/convolutional-neural-networks-tutorial-in-pytorch/
[3] http://cs231n.github.io/neural-networks-1/
[4] https://www.naic.edu/ao/scientist-user-portal/astronomy/receivers

Table 1. Classes of signals in simulated data and their respective parameter values.

Category	Dynamic spectrum contents	W_ν (MHz)	W_t (s) (ms)	S/N (number)	DM (pc cm^{-3})
(1)	(2)	(3)	(4)	(5)	(6)
noise	Noise[a] only	N/A	N/A	N/A	N/A
llbb	Long-lived, broadband RFI + noise	$100-500$	$10-100$	$5-15$	0
llnb	Long-lived, narrowband RFI + noise	$0.1-10$	$10-100$	$5-15$	0
slbb	Short-lived, broadband RFI + noise	$100-500$	$0.1-4$	$5-15$	0
slnb	Short-lived, narrowband RFI + noise	$0.1-10$	$0.1-4$	$5-15$	0
pulse	Dispersed pulse + noise	$100-1000$	$0.1-10$	$5-15$	$10-3000$

[a] Noise samples are drawn from a standard normal distribution.

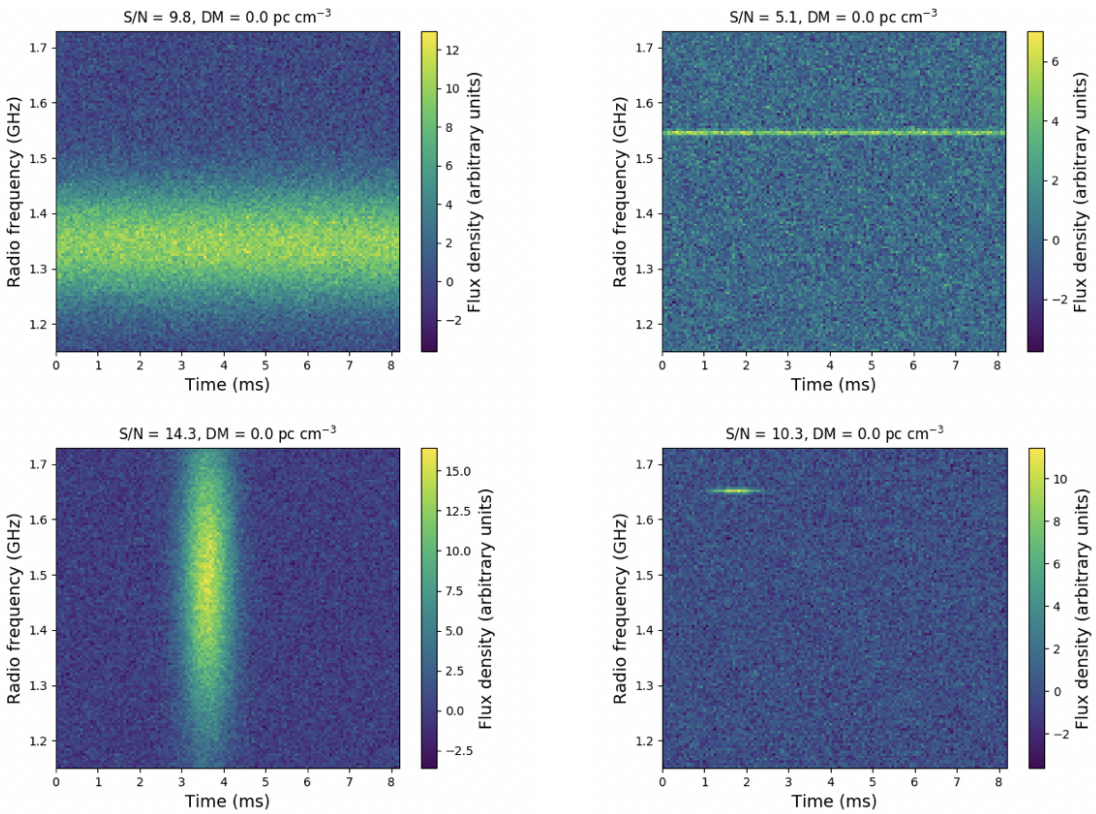

Figure 2. Fake noisy dynamic spectra illustrating simulated long-lived broadband (top left), long-lived narrowband (top right), short-lived broadband (bottom left), and short-lived narrowband (bottom right) RFI. The S/N values of these signals are indicated in their respective panel titles.

following form:

$$A(\nu, t) = (S/N) \times \exp\left(-\frac{(4\ln 2)(\nu-\nu_0)^2}{W_\nu^2}\right) \times \exp\left(-\frac{(4\ln 2)(t-t_0)^2}{W_t^2}\right). \quad (6)$$

Here, S/N sets the signal-to-noise ratio of $A(\nu, t)$ relative to the background noise having unit variance. W_ν and W_t represent the

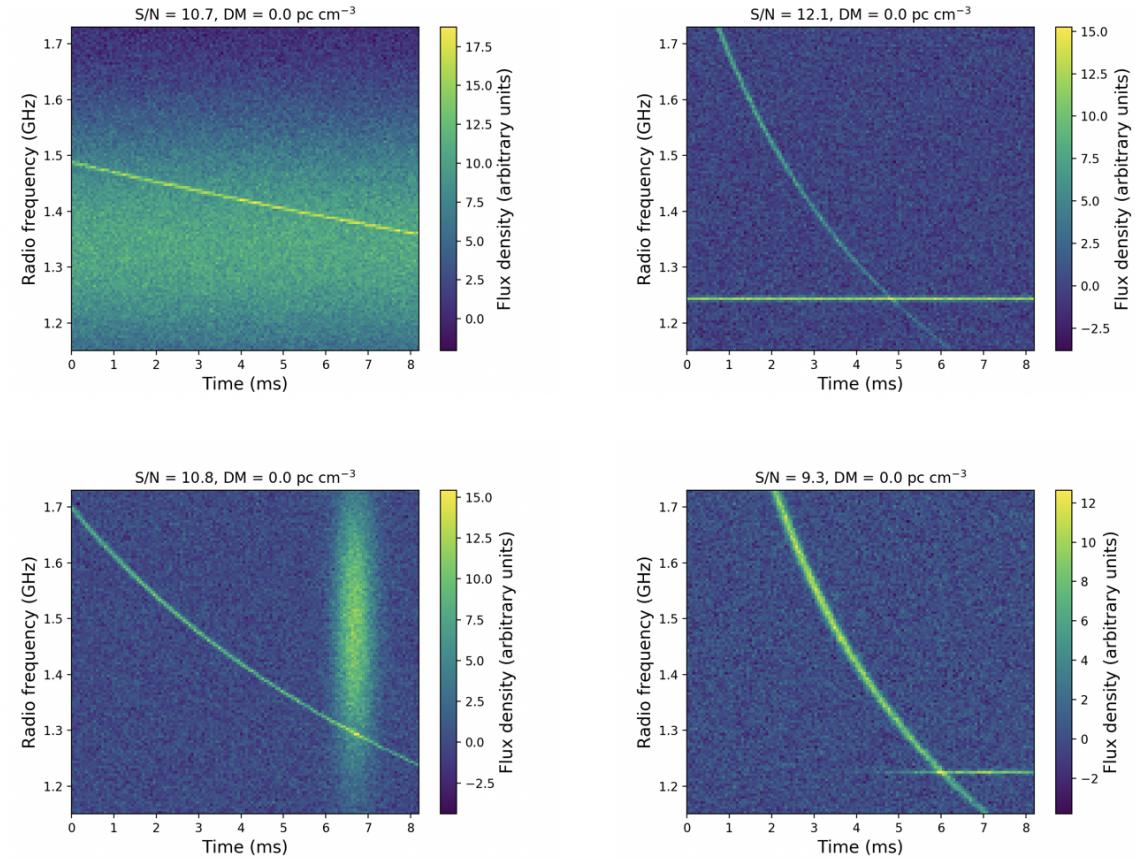

Figure 3. Example noisy dynamic spectra containing dispersed pulses of varying DMs along with long-lived broadband (top left), long-lived narrowband (top right), short-lived broadband (bottom left), and short-lived narrowband (bottom right) RFI. The S/N and DM values specified in each panel title correspond to those of the simulated RFI signal in the respective dynamic spectrum.

full-width-at-half-maximum (FWHM) spans of $\mathrm{A}(\nu, \mathrm{t})$ along the frequency and time axes respectively. When generating $\mathrm{A}(\nu, \mathrm{t})$, values of W_ν and W_t are drawn from separate uniform distributions extending between two specified limits. The allowed range of values of W_ν and W_t depend on the nature (narrowband or broadband, short-lived or long-lived) of RFI or dispersed pulses we intend to model.

Table 1 lists various classes of signals modeled in our simulated data sets along with their allowed parameter ranges. To generate FRBs, we first construct $\mathrm{A}(\nu, \mathrm{t})$ using randomly chosen values of W_ν and W_t that lie within their respective specified ranges for dispersed pulses. A DM value is drawn at random from a uniform distribution extending over $10 - 3000$ pc cm^{-3}. The initialized $\mathrm{A}(\nu, \mathrm{t})$ is then dedispersed at the selected DM in accordance with Eqn. 1.

Figure 2 shows examples of simulated long-lived broadband (llbb) (top left), long-lived narrowband (llnb) (top right), short-lived broadband (slbb) (bottom left), and short-lived narrowband (slnb) (bottom right) RFI. Real-world data often contain dispersed pulses and RFI together in a single dynamic spectrum. Example dynamic spectra containing dispersed pulses along with different RFI types are illustrated in Fig. 3.

4. CUSTOM MODEL IMPLEMENTATIONS

We develop CNNs to perform feature classification on five categories of input data: noise,

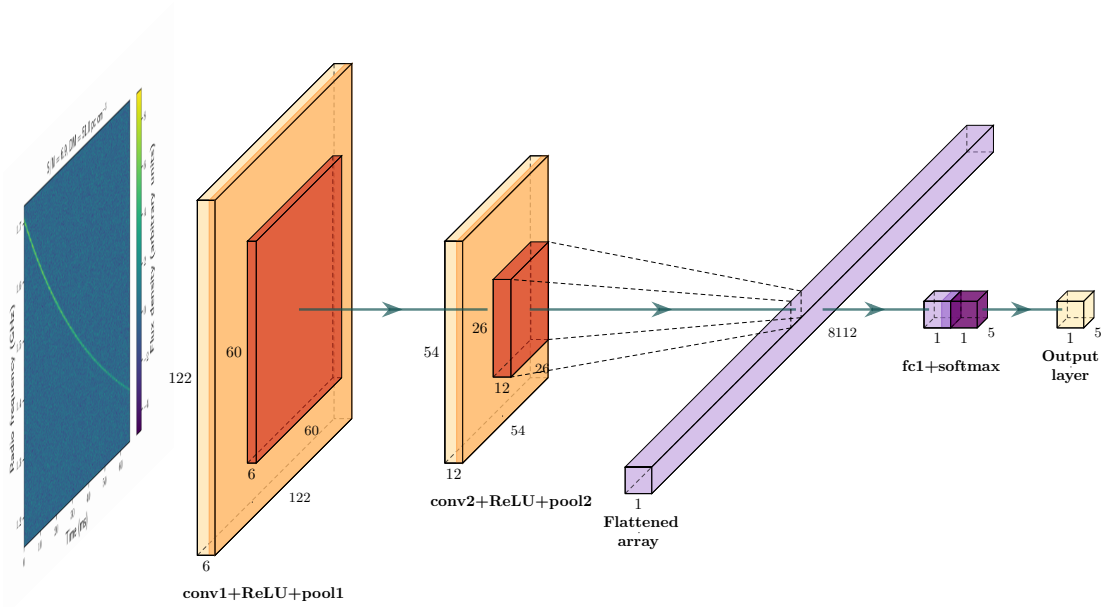

Figure 4. A 6−layer network accepting input dynamic spectra of size 128 × 128. The hidden layers in this network include two convolutional layers (light orange box) with ReLU activation functions (brown edges at right end of convolutional layer), two max-pooling layers (dark orange box), and a fully-connected layer (light purple box) with a softmax activation function (dark purple box). Output sizes are indicated next to each layer. This diagram was generated using a LaTeXtemplate prepared by Iqbal (2018).

llbb, llnb, slbb, and slnb. As there exists no theoretically optimal CNN design for executing a specific task, we experiment with different model architectures to understand the scaling of network performance with various properties (e.g. network depth, convolutional layer filter size, etc.). In Section 4.1, we explore the utility of a "shallow" 6−layer network. Deeper variations of this CNN are considered in Sec. 4.2. Our models adopt the categorical CE function as the loss criterion to perform multi-class classification. In addition, we uniformly use the Adam optimizer with an initial learning rate $\eta = 10^{-5}$ in all our networks. For our investigation in Sec. 4.1 and 4.2, we exclude dispersed pulses from our training and validation data.

4.1. *6−layer Network with Inputs Sized* 128 × 128

Figure 4 shows the structure of a 6−layer network operating on input dynamic spectra of size 128 × 128. This model comprises of two convolutional layers, two max-pooling layers, and a fully-connected layer hidden between its input and output layers. While the convolutional layers use filters of size F = 6 with stride S = 1, the pooling layers are specified with F = 4 and S = 2. The first convolutional layer accepts 1 input channel and outputs 6 channels (N = 6). Successive convolutional layers increase the number of output channels by factors of 2. Once trained, the final outputs from this CNN represent the probabilities of an input dynamic spectrum getting classified as either noise, llbb, llnb, slbb, or slnb.

For training our 6−layer network, we firstly generate a large data set containing 1200 labeled dynamic spectra of each output category. 200 dynamic spectra of each class are then randomly selected and grouped together as validation data, while the remainder constitute the

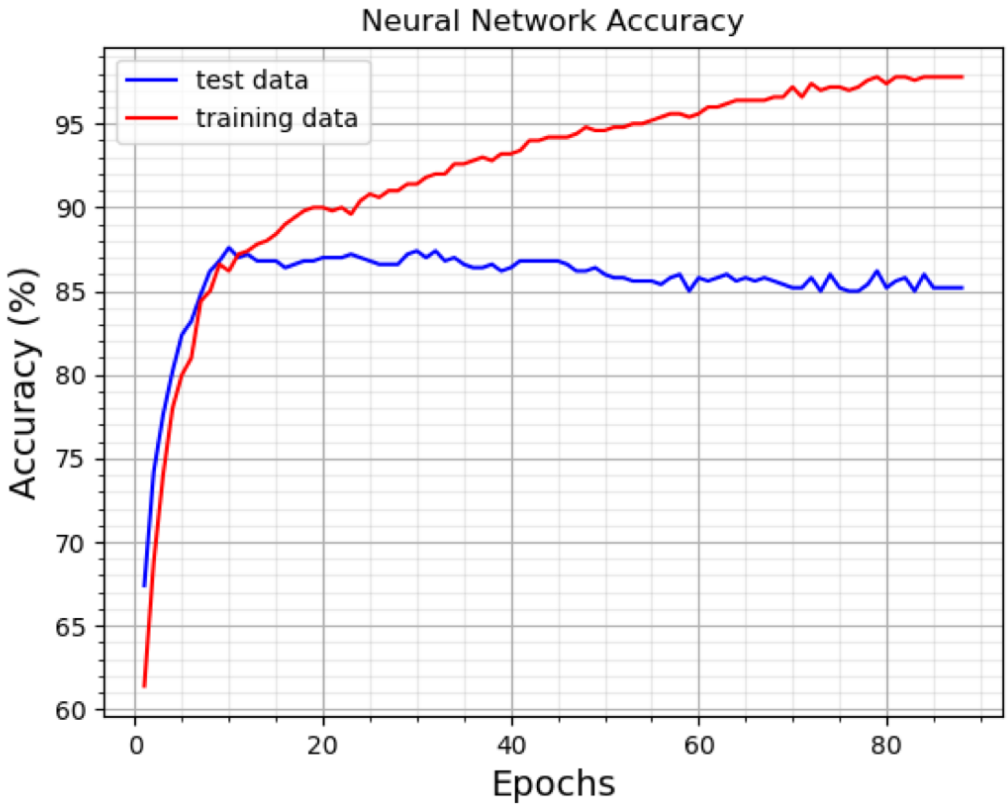

Figure 5. Evolution of the accuracy of our 6−layer network (see Fig. 4) over 90 training epochs. The red and blue lines represent accuracy curves for the training and validation data respectively.

training data set.

Let us define the accuracy of a network as the fraction of input dynamic spectra that it correctly labels. To identify different fitting regimes (see Sec. 2.2), we pass our 6−layer network through 90 epochs of training. The network accuracy is calculated at each epoch for both the training and validation data. Figure 5 illustrates the epoch-dependent evolution of the accuracies achieved by our 6−layer network on the training (red) and validation (blue) data. As expected, the network performance on the training data monotonically increases with additional training. However, the network accuracy on the validation data saturates at $\sim 85 - 87\%$ beyond epoch 10, indicating that our network has begun to overfit the training data. We find that the maximum network accuracy on the validation data hardly increases when we use a larger training set. This suggests that we require deeper CNNs for obtaining network accuracies exceeding 90% on our validation data.

The accuracy matrix shown in Fig. 6 reveals the detailed results of our network training after 57 epochs. While our model does well at recognizing broadband and long-lived features, it struggles to differentiate between noise and slnb in the dynamic spectrum. Considering that the minimum W_t for slnb in Tab. 1 spans only $1 - 2$ time samples, it is not surprising that our network misidentifies short-lived features as noise spikes. Similarly, along the frequency axis, the usage of 128 spectral channels across the $1.15 - 1.73$ GHz bandwidth of the dynamic spectrum implies a channel bandwidth of $\Delta\nu \approx$

Loss = 0.15
Epochs = 57
Total Accuracy = 86%, 94%

Expected vs Predicted Classification Table

■ = test data
■ = training data

Expected \ Predicted	Long-lived broadband	Long-lived narrowband	Short-lived broadband	Short-lived narrowband	Noise
Long-lived broadband	1.0 1.0	0.0 0.0	0.0 0.0	0.0 0.0	0.0 0.0
Long-lived narrowband	0.0 0.0	0.88 0.88	0.0 0.0	0.04 0.01	0.08 0.11
Short-lived broadband	0.01 0.0	0.0 0.0	0.98 1.0	0.01 0.0	0.0 0.0
Short-lived narrowband	0.0 0.0	0.07 0.04	0.0 0.0	0.67 0.85	0.26 0.11
Noise	0.0 0.0	0.02 0.0	0.0 0.0	0.22 0.02	0.76 0.98

Figure 6. Accuracy matrix detailing the performance of our 6−layer network (see Fig. 4) after 57 epochs of training. For every pair of output classes (i, j), the matrix entry at (i, j) represents the fraction of labeled dynamic spectra of category i that have been classified into category j. Red and black values indicate entries for the training and validation data respectively. An ideal network is characterized by an identity accuracy matrix for both the training and validation data.

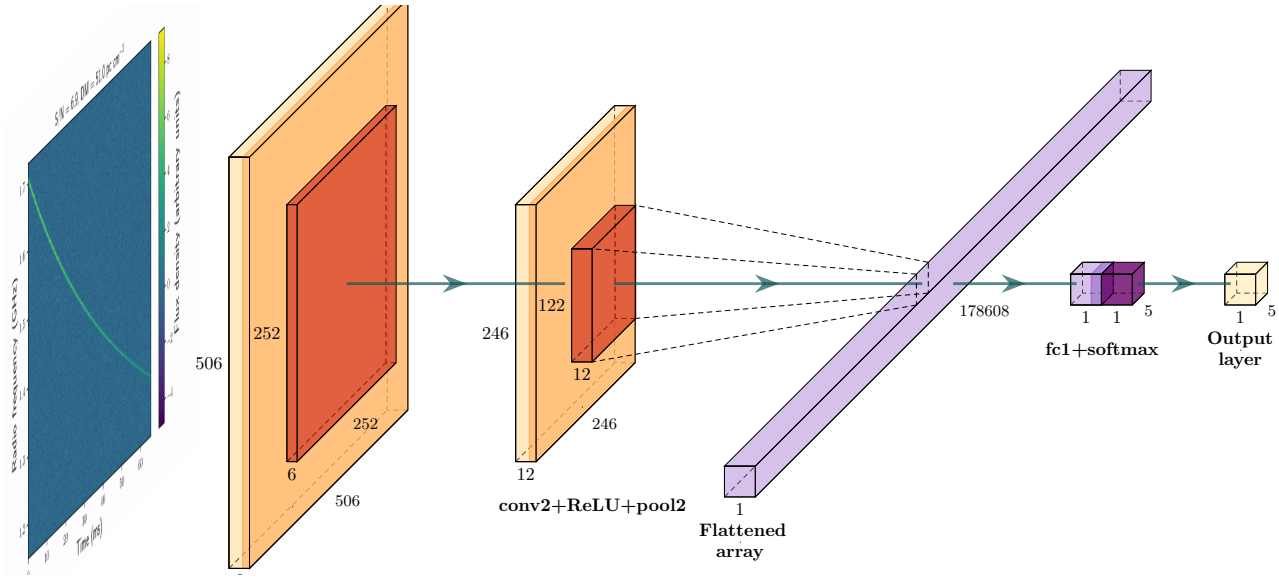

Figure 7. A 6−layer network with input size 512 × 512. Similar to the network shown in Fig. 4, the hidden layers in this model consist of two convolutional layers (light orange boxes) with ReLU activation functions (brown edges at right end of convolutional layers), two max-pooling layers (dark orange boxes), and a fully-connected layer (light purple box) with softmax activation (dark purple box). This diagram was prepared using a LaTeXtemplate developed by Iqbal (2018).

4.53 MHz. Then, $(W_\nu)_{\rm min}/\Delta\nu \ll 1$ for narrow-band structures in our dynamic spectrum. As

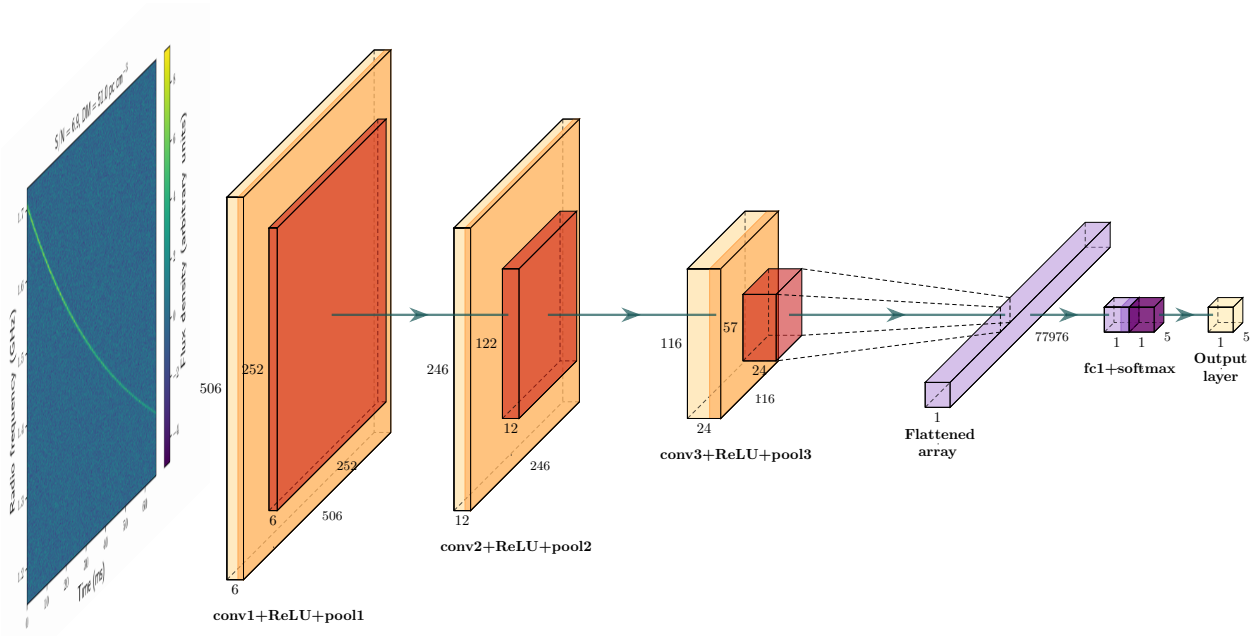

Figure 8. An 8−layer network built by inserting an additional convolutional layer and a max-pooling layer into the 6−layer network in Fig. 7. All layers and activation functions are labeled using the color scheme adopted in Fig. 4 and 6. This figure was generated using a LaTeXtemplate provided by Iqbal (2018).

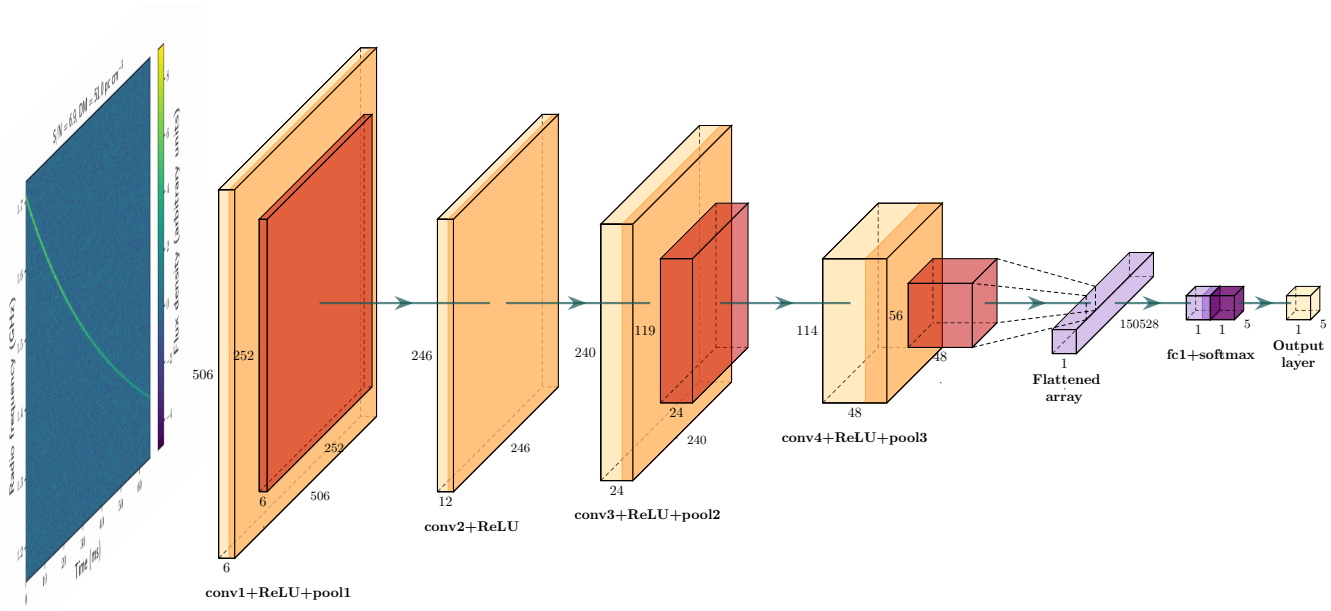

Figure 9. A 9−layer network containing a convolutional layer (conv2) with no accompanying pooling layer. All convolutional layers (light orange boxes), ReLU activation functions (brown edges at right end of convolutional layer), max-pooling layers (dark orange boxes), the fully-connected layer (light purple box), and the softmax activation function (dark purple box) are labeled. Respective output sizes are indicated next to each layer. This diagram was generated using a LaTeXtemplate prepared by Iqbal (2018).

a result, our network performs poorly on slnb when compared to slbb. However, this issue can be mitigated through finer spectral channelization of our dynamic spectrum. Hence, we

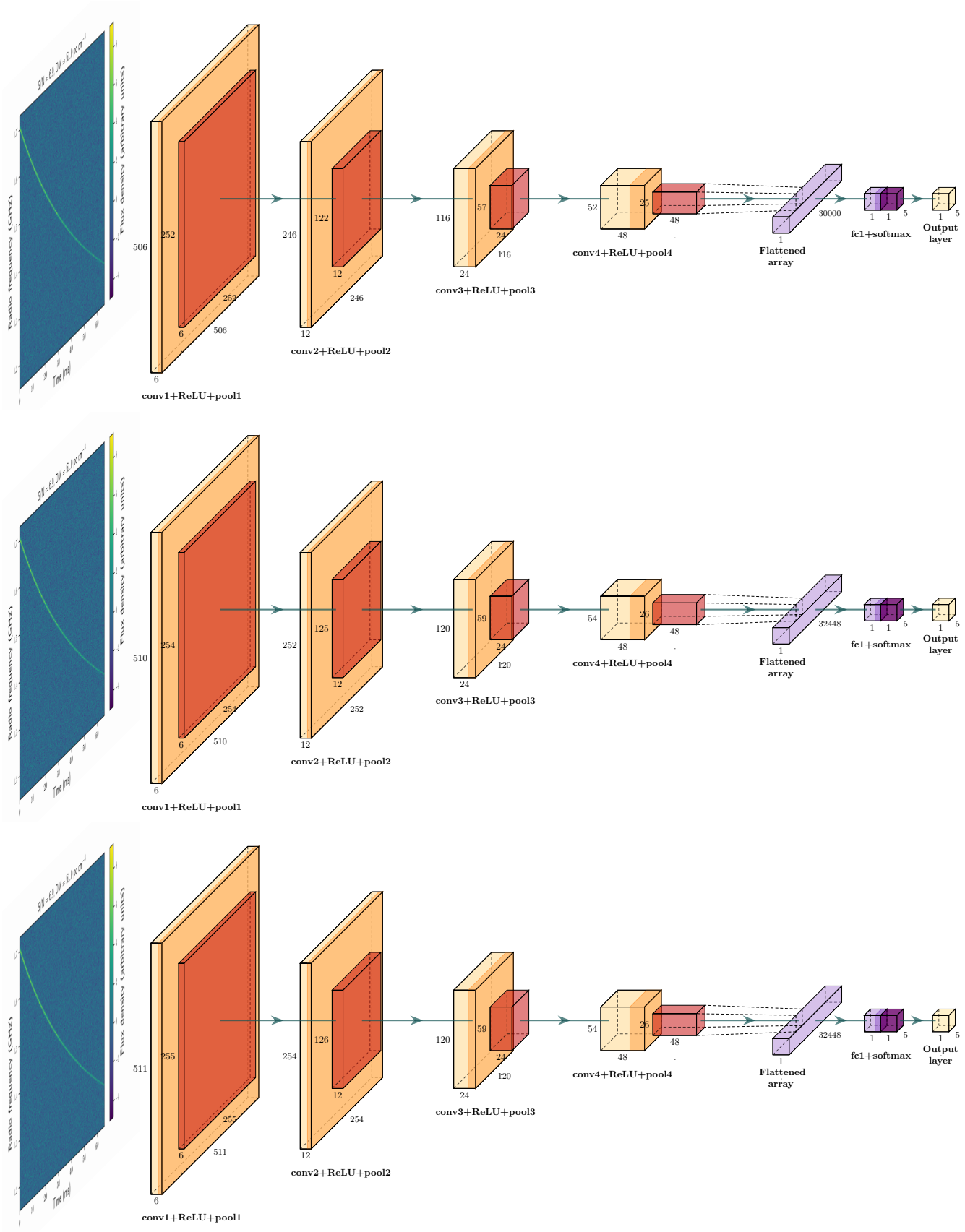

Figure 10. Models 10A (top), 10B (middle) and 10C (right). Convolutional layers (light orange boxes), max-pooling layers (dark orange boxes) and fully-connected layers (light purple box) are labeled with their respective output sizes. ReLU (brown edges at right end of convolutional layer) and softmax (dark purple box) activation functions are also indicated. This diagram was generated using a LaTeXtemplate prepared by Iqbal (2018).

shift to input sizes of 512×512 for our deeper models in Sec. 4.2. Figure 7 shows a version of the CNN in Fig. 4 that works with inputs sized 512×512.

4.2. *Deep models*

Starting out with the 6−layer network in Fig. 7, we build deeper models by inserting additional convolutional and max-pooling layers behind the fully connected layer in our CNN architectures. Figure 8 illustrates an 8−layer CNN containing three convolutional layers along with their accompanying ReLU activation functions and max-pooling layers. The convolutional layers in this network use $F = 7$ and $S = 1$, while the pooling layers have $F = 4$ and $S = 2$. To examine the role of pooling layers in determining network accuracy, we build a 9−layer network. This CNN shown in Fig. 9 differs from our 8−layer network in including a convolutional layer (conv2) that lacks an immediately following pooling layer. Finally, we construct a 10−layer network (top panel of Fig. 10) by incorporating a pooling layer associated with conv2 of our 9−layer CNN. For the sake of convenience, let us henceforth refer to this 10−layer model as "10A".

We note that the first two convolutional layers, conv1 and conv2, in all models discussed so far use filters with size $F = 7$. Since $(W_t)_{min} < 2 \times t_{samp}$ for short-lived RFI, we suspect that the use of such large filters can possibly impede the detectability of short-lived features in our dynamic spectra. We investigate this hypothesis using models "10B" and "10C" whose architectures are displayed, respectively, in the middle and bottom panels of Fig. 10. As evident from this diagram, models 10B and 10C have the same basic framework as model 10A. However, while conv1 and conv2 of model 10A use $F = 7$, the corresponding layers in 10B and 10C use $F = 3$ and $F = 2$ respectively.

Following the training procedure outlined in Sec. 4.1, we train our deep models using a training set comprised of 2000 labeled inputs from each output class. As deeper CNNs take longer to train than their shallow model counterparts, it is not prudent to specify a maximum epoch limit during model training. Rather, we halt network training when the CE loss averaged over all training data at the end of an epoch falls below 0.05.

Figure 11 reports the network accuracy values achieved by various deep models on both the training and the validation data at different epochs. Accuracy matrices for each of these trained CNNs at their respective final training epochs are shown in Appendix A. In terms of performance on the validation data, Figure 11 ranks our models as follows.

$$\begin{aligned} 10B \simeq 10C &> 10A \\ &> 9-\text{layer} \simeq 8-\text{layer} \\ &> 6-\text{layer} \end{aligned} \tag{7}$$

Clearly, our deeper CNNs are able to learn features at more detailed levels of abstractions, thereby enabling sharper pattern recognition in the same input data. The comparable performances of our 9−layer and 8−layer networks highlights the significance of a pooling layer for collating trends from adjacent pixels and reducing the number of redundant computations. The marginally better performances of models 10B and 10C relative to that of 10A verifies our hypothesis that convolution kernels of smaller size are essential for identifying short-lived, narrowband signals in dynamic spectra. Finally, from Fig. 11, we infer that the accuracy gain obtained by increasing the depth of a CNN diminishes as the total number of layers in the network increases. In other words, while the addition of 2 layers to our 6−layer network increased the network accuracy on the validation data by 10%, the corresponding increase on go-

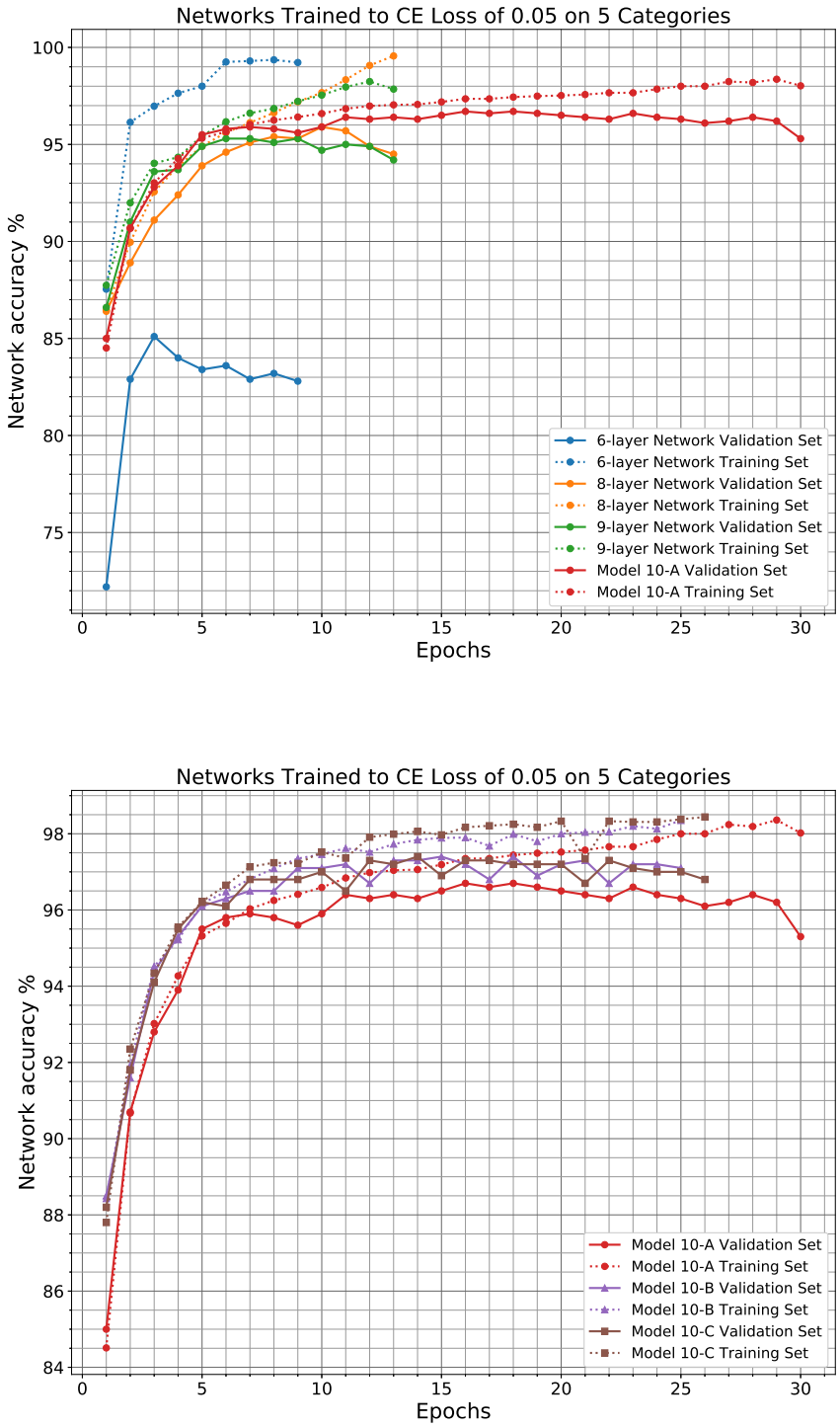

Figure 11. Top: Network accuracy vs. training epoch for our 6−layer (blue), 8−layer (orange), 9−layer (green) networks and model 10A (red). Bottom: Accuracy curves comparing the performances of models 10A (red), 10B (violet) and 10C (brown). All curves in both panels terminate at the epoch where the mean CE loss during training equals 0.05. The dotted and solid lines represent accuracy values at different epochs for the training and validation data respectively.

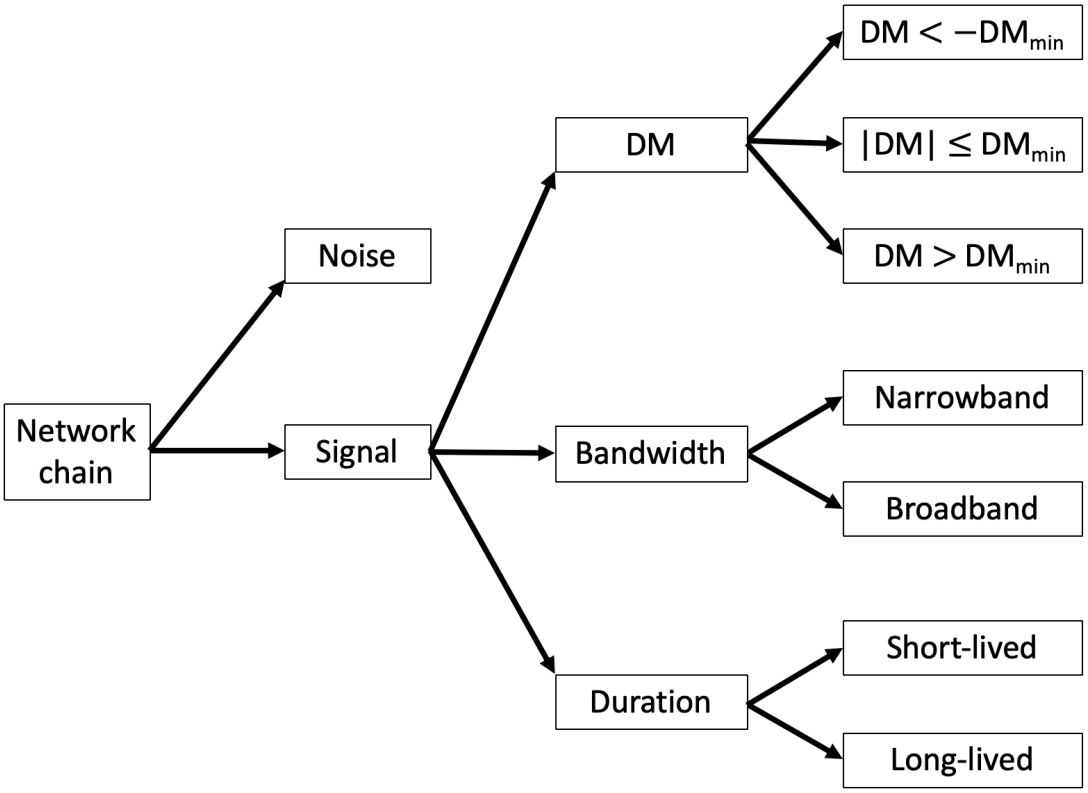

Figure 12. Desired output classification hierarchy from a CNN that can function on both dedispersed and non-dedispersed dynamic spectra. Here, $\mathrm{DM_{min}}$ denotes the minimum DM detectable due to the discreteness of data along the time axis. For a given time sampling $\mathrm{t_{samp}}$, pulses with $\mathrm{t_{DM}} < \mathrm{t_{samp}}$ appear consistent with $\mathrm{DM} = 0$ pc cm^{-3} signals in the dynamic spectrum.

ing from our 8−layer network to our 10−layer CNNs is $\sim 1 - 2\%$.

5. SUMMARY AND FUTURE WORK

We have successfully implemented CNNs of varying depths for performing multi-class classification on different RFI characteristics (short-lived or long-lived, narrowband or broadband) and noise in non-dedispersed, artificially generated dynamic spectra. In accordance with Eldan & Shamir (2015) and Mhaskar et al. (2016), we find that deeper models are able to recognize sharper features in dynamic spectra and thereby, output more accurate classifications. While our 6−layer network produces a network accuracy of $\sim 85\%$ on validation data, our 10−layer models yield correct classifications for $\sim 95 - 97\%$ of their inputs.

Future developments in this work include the incorporation of labeled real-world data and dispersed pulses in model training and testing. We note that the category "dispersed pulses" is too broad and does not permit us to distinguish between dispersed astrophysical signals possessing different spectral and temporal characteristics. For example, bright single pulses from pulsars are often broadband, whereas FRBs (Gajjar et al. 2018; CHIME/FRB Collaboration et al. 2019c) are narrowband in nature. Furthermore, the coherent dedispersion technique employed in FRB and pulsar searches produces dedispersed dynamic spectra where RFI are seen as $\mathrm{DM} < 0$ signals. We hope to accomodate the

broad diversity of spectro-temporal structure seen in dispersed pulses in our model training, and render our CNNs operable on both dedispersed and non-dedispersed dynamic spectra.

We envisage accomplishing this goal by switching from multi-class classification to multi-label classification. In multi-class classification, a model classifies an input into one of N output classes. On the other hand, in multi-label classification, a given input can be assigned multiple labels corresponding to one or more classes. To enable multi-label classification, we plan to experiment with the binary CE and multi-label margin loss criteria in our networks.

Our proposed classification scheme is illustrated in Fig. 12. Plausible network architectures for realizing this hierarchical classification strategy include residual networks (He et al. 2015) and branch CNNs (B-CNNs: Zhu & Bain (2017)).

APPENDIX

A. ACCURACY TABLES FOR DIFFERENT MODEL ARCHITECTURES

Expected vs. Predicted Classification Table for 6−layer Network

Loss = 0.05
Epochs = 9
Accuracy = 83%, 99%

Predicted / Expected	Long-lived broadband	Long-lived narrowband	Short-lived broadband	Short-lived narrowband	noise
Long-lived broadband	1.0 1.0	0.0 0.0	0.0 0.0	0.0 0.0	0.0 0.0
Long-lived narrowband	0.0 0.001	0.91 0.99	0.0 0.0	0.04 0.0	0.05 0.0005
Short-lived broadband	0.02 0.04	0.0 0.0	0.91 0.96	0.06 0.0005	0.01 0.0
Short-lived narrowband	0.0 0.0	0.05 0.0	0.0 0.0	0.44 1.0	0.51 0.0
noise	0.0 0.0	0.0 0.0	0.0 0.0	0.16 0.0	0.84 1.0

■ = validation data
■ = training data

(a)

Expected vs. Predicted Classification Table for 8−layer Network

Loss = 0.05
Epochs = 13
Accuracy = 95%, 99%

Predicted / Expected	Long-lived broadband	Long-lived narrowband	Short-lived broadband	Short-lived narrowband	noise
Long-lived broadband	1.0 1.0	0.0 0.0	0.0 0.0	0.0 0.0	0.0 0.0
Long-lived narrowband	0.0 0.0	0.95 0.99	0.0 0.0	0.01 0.001	0.03 0.006
Short-lived broadband	0.0 0.0	0.0 0.0	0.98 0.99	0.02 0.001	0.0 0.0005
Short-lived narrowband	0.0 0.0	0.02 0.001	0.0 0.0	0.84 0.99	0.13 0.01
noise	0.0 0.0	0.0 0.0	0.0 0.0	0.05 0.001	0.95 0.99

■ = validation data
■ = training data

(b)

Expected vs. Predicted Classification Table for 9−layer Network

Loss = 0.05
Epochs = 13
Accuracy = 94%, 98%

Predicted / Expected	Long-lived broadband	Long-lived narrowband	Short-lived broadband	Short-lived narrowband	noise
Long-lived broadband	1.0 0.99	0.0 0.01	0.0 0.0	0.0 0.0	0.0 0.0
Long-lived narrowband	0.0 0.0	0.96 0.99	0.0 0.0	0.0 0.0	0.04 0.01
Short-lived broadband	0.0 0.001	0.0 0.0	0.97 0.97	0.03 0.02	0.0 0.002
Short-lived narrowband	0.0 0.0	0.03 0.006	0.0 0.0	0.76 0.95	0.2 0.05
noise	0.0 0.0	0.0 0.0	0.0 0.0	0.0 0.0	1.0 1.0

■ = validation data
■ = training data

(c)

Expected vs. Predicted Classification Table for Network 10−A

Loss = 0.05
Epochs = 30
Accuracy = 95%, 98%

Predicted / Expected	Long-lived broadband	Long-lived narrowband	Short-lived broadband	Short-lived narrowband	noise
Long-lived broadband	1.0 1.0	0.0 0.0	0.0 0.0	0.0 0.0	0.0 0.0
Long-lived narrowband	0.0 0.0	0.96 0.98	0.0 0.0	0.004 0.0	0.03 0.002
Short-lived broadband	0.05 0.02	0.0 0.0	0.94 0.97	0.01 0.0005	0.0 0.0
Short-lived narrowband	0.0 0.0	0.0 0.003	0.0 0.002	0.86 0.94	0.13 0.05
noise	0.0 0.0	0.0 0.0005	0.0 0.0	0.013 0.0	0.99 0.99

■ = validation data
■ = training data

(d)

Expected vs. Predicted Classification Table for Network 10−B

Loss = 0.05
Epochs = 25
Accuracy = 97%, 98%

Predicted / Expected	Long-lived broadband	Long-lived narrowband	Short-lived broadband	Short-lived narrowband	noise
Long-lived broadband	1.0 1.0	0.0 0.0	0.0 0.0	0.0 0.0	0.0 0.0
Long-lived narrowband	0.0 0.0	0.97 0.97	0.0 0.0	0.0 0.001	0.03 0.02
Short-lived broadband	0.0 0.0	0.0 0.0	0.99 0.99	0.006 0.0005	0.0 0.0
Short-lived narrowband	0.0 0.0	0.0 0.001	0.0 0.0	0.89 0.94	0.11 0.05
noise	0.0 0.0	0.0 0.0	0.0 0.0	0.004 0.0	0.99 1.0

■ = validation data
■ = training data

(e)

Expected vs. Predicted Classification Table for Network 10−C

Loss = 0.05
Epochs = 26
Accuracy = 97%, 98%

Predicted / Expected	Long-lived broadband	Long-lived narrowband	Short-lived broadband	Short-lived narrowband	noise
Long-lived broadband	1.0 1.0	0.0 0.0	0.0 0.0	0.0 0.0	0.0 0.0
Long-lived narrowband	0.0 0.0	0.96 0.97	0.0 0.0	0.01 0.001	0.03 0.03
Short-lived broadband	0.005 0.0	0.0 0.0	0.98 1.0	0.01 0.0	0.0 0.0
Short-lived narrowband	0.0 0.0	0.0 0.0	0.0 0.0	0.90 0.95	0.1 0.05
noise	0.0 0.0	0.0 0.0	0.0 0.0	0.009 0.0	0.99 1.0

■ = validation data
■ = training data

(f)

Figure 13. Accuracy matrices detailing the performances of our trained: (a) 6−layer network (Fig. 7), (b) 8−layer network (Fig. 8), (c) 9−layer network (Fig. 9), (d) model 10A (top panel of Fig. 10), (e) model 10B (middle panel of Fig. 10), and (f) model 10C (bottom panel of Fig. 10). For every pair of output classes (i, j) in a given matrix, the matrix entry at (i, j) represents the fraction of labeled dynamic spectra of category i that have been classified into category j. Red and black values indicate entries for the training and validation data respectively. An ideal network is characterized by an identity accuracy matrix for both the training and validation data.

Software: Python 3 (http://www.python.org), PyTorch (Paszke et al. 2019), NumPy (van der Walt et al. 2011), Matplotlib (Hunter 2007), SciPy (Virtanen et al. 2019)

REFERENCES

Agarwal, D., Aggarwal, K., Burke-Spolaor, S., Lorimer, D. R., & Garver-Daniels, N. 2019, arXiv e-prints. https://arxiv.org/abs/1902.06343

Bannister, K. W., Deller, A. T., Phillips, C., et al. 2019, Science, 365, 565, doi: 10.1126/science.aaw5903

Bottou, L. 2010, in Proceedings of COMPSTAT'2010, ed. Y. Lechevallier & G. Saporta (Heidelberg: Physica-Verlag HD), 177–186

Chatterjee, S., Law, C. J., Wharton, R. S., et al. 2017, Nature, 541, 58, doi: 10.1038/nature20797

CHIME/FRB Collaboration, Amiri, M., Bandura, K., et al. 2019a, Nature, 566, 235, doi: 10.1038/s41586-018-0864-x

CHIME/FRB Collaboration, Andersen, B. C., Bandura, K., et al. 2019b, ApJL, 885, L24, doi: 10.3847/2041-8213/ab4a80

CHIME/FRB Collaboration, Amiri, M., Bandura, K., et al. 2019c, Nature, 566, 230, doi: 10.1038/s41586-018-0867-7

Connor, L., & van Leeuwen, J. 2018, AJ, 156, 256, doi: 10.3847/1538-3881/aae649

Cordes, J. M., & McLaughlin, M. A. 2003, ApJ, 596, 1142, doi: 10.1086/378231

Eldan, R., & Shamir, O. 2015, arXiv e-prints, arXiv:1512.03965. https://arxiv.org/abs/1512.03965

Gajjar, V., Siemion, A. P. V., Price, D. C., et al. 2018, ApJ, 863, 2, doi: 10.3847/1538-4357/aad005

He, K., Zhang, X., Ren, S., & Sun, J. 2015, arXiv e-prints, arXiv:1512.03385. https://arxiv.org/abs/1512.03385

Hessels, J. W. T., Spitler, L. G., Seymour, A. D., et al. 2019, ApJL, 876, L23, doi: 10.3847/2041-8213/ab13ae

Hunter, J. D. 2007, Computing in Science Engineering, 9, 90, doi: 10.1109/MCSE.2007.55

Iqbal, H. 2018, HarisIqbal88/PlotNeuralNet v1.0.0, doi: 10.5281/zenodo.2526396

Kingma, D. P., & Ba, J. 2014, arXiv e-prints, arXiv:1412.6980. https://arxiv.org/abs/1412.6980

Kumar, P., Shannon, R. M., Osłowski, S., et al. 2019, ApJL, 887, L30, doi: 10.3847/2041-8213/ab5b08

Marcote, B., Nimmo, K., Hessels, J. W. T., et al. 2020, Nature, doi: 10.1038/s41586-019-1866-z

Mhaskar, H., Liao, Q., & Poggio, T. 2016, arXiv e-prints, arXiv:1603.00988. https://arxiv.org/abs/1603.00988

Murphy, K. P. 2012, Machine Learning: A Probabilistic Perspective (The MIT Press)

Paszke, A., Gross, S., Massa, F., et al. 2019, in Advances in Neural Information Processing Systems 32, ed. H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, & R. Garnett (Curran Associates, Inc.), 8024–8035

Petroff, E., Barr, E. D., Jameson, A., et al. 2016, PASA, 33, e045, doi: 10.1017/pasa.2016.35

Prochaska, J. X., Macquart, J.-P., McQuinn, M., et al. 2019, Science, 366, 231, doi: 10.1126/science.aay0073

Ravi, V., Catha, M., D'Addario, L., et al. 2019, Nature, 572, 352, doi: 10.1038/s41586-019-1389-7

Rosenblatt, F. 1958, Psychological Review, 65, 386, doi: 10.1037/h0042519

Spitler, L. G., Scholz, P., Hessels, J. W. T., et al. 2016, Nature, 531, 202, doi: 10.1038/nature17168

van der Walt, S., Colbert, S. C., & Varoquaux, G. 2011, Computing in Science Engineering, 13, 22, doi: 10.1109/MCSE.2011.37

Virtanen, P., Gommers, R., Oliphant, T. E., et al. 2019, arXiv e-prints, arXiv:1907.10121. https://arxiv.org/abs/1907.10121

Zhang, Y. G., Gajjar, V., Foster, G., et al. 2018, ApJ, 866, 149, doi: 10.3847/1538-4357/aadf31

Zhu, X., & Bain, M. 2017, arXiv e-prints, arXiv:1709.09890. https://arxiv.org/abs/1709.09890