The Astronomical Journal, 156:256 (13pp), 2018 December https://doi.org/10.3847/1538-3881/aae649

Applying Deep Learning to Fast Radio Burst Classification

Liam Connor[1,2] and Joeri van Leeuwen[1,2]
[1] ASTRON, Netherlands Institute for Radio Astronomy, Postbus 2, 7990 AA Dwingeloo, The Netherlands; liam.dean.connor@gmail.com
[2] Anton Pannekoek Institute for Astronomy, University of Amsterdam, Science Park 904, 1098 XH Amsterdam, The Netherlands
Received 2018 March 8; revised 2018 October 2; accepted 2018 October 2; published 2018 November 9

Abstract

Upcoming fast radio burst (FRB) surveys will search $\sim 10^3$ beams on the sky with a very high duty cycle, generating large numbers of single-pulse candidates. The abundance of false positives presents an intractable problem if candidates are to be inspected by eye, making it a good application for artificial intelligence (AI). We apply deep learning to single-pulse classification and develop a hierarchical framework for ranking events by their probability of being astrophysical transients. We construct a treelike deep neural network that takes multiple or individual data products as input (e.g., dynamic spectra and multibeam information) and trains on them simultaneously. We have built training and test sets using false-positive triggers from real telescopes, simulated FRBs, and pulsar single pulses. Training the network was independently done for both the CHIME Pathfinder and Apertif. High accuracy and recall can be achieved with a labeled training set of a few thousand events. Even with high triggering rates, classification can be done very quickly on graphical processing units, which is essential for selective voltage dumps or real-time VOEvents. We investigate whether dedispersion back ends could be replaced by a real-time DNN classifier. It is shown that a single forward propagation through a moderate convolutional network could be faster than brute-force dedispersion, but the low signal-to-noise per pixel makes such a classifier suboptimal for this problem. Real-time automated classification will prove useful for bright, unexpected signals, both now and when searchable parameter spaces outgrow our ability to manually inspect data, such as for the SKA and ngVLA.

Key words: methods: data analysis – pulsars: general – techniques: image processing

1. Introduction

Fast radio bursts (FRBs) are bright, millisecond-duration, extragalactic radio transients characterized by dispersion measures (DMs) that are significantly larger than the expected Milky Way contribution. They have been detected at flux densities between tens of μJy and tens of Jy (Lorimer et al. 2007; Thornton et al. 2013; Petroff et al. 2015a; Ravi et al. 2016). The majority of early detections were made with the Parkes telescope multibeam receiver, but in recent years, detections have been made at Arecibo (Spitler et al. 2014), Green Bank Telescope (GBT; Masui et al. 2015), the Upgraded Molonglo Synthesis Telescope (UTMOST; Caleb et al. 2017), and the Australian Square Kilometre Array Pathfinder (ASKAP; Bannister et al. 2017). The only source known to repeat is FRB 121102 (Scholz et al. 2016; Spitler et al. 2016), allowing for the first host galaxy localization using very long baseline interferometry (VLBI; Marcote et al. 2017; Tendulkar et al. 2017). Recently, the repeating bursts from this source were found to be almost 100% linearly polarized with a Faraday rotation measure (RM) of 10^5 rad m^{-2} (Michilli et al. 2018a).

There are likely thousands of detectable events each day across the full sky, but only ~ 50 have been observed to date. This is due to the moderate field of view (FoV) and relatively low duty cycle of current FRB surveys. Still, such surveys have produced thousands of false-positive triggers for each true FRB, the diagnostic plots of which have traditionally been inspected by eye (Masui et al. 2015; Amiri et al. 2017; Caleb et al. 2017; Foster et al. 2018). For upcoming fast transient surveys, the false-positive problem will be intractable if single-pulse candidates are to be human-inspected, even with rigorous removal of radio frequency interference (RFI). The Canadian Hydrogen Intensity Mapping Experiment (CHIME) will search 1024 beams at all times between 400 and 800 MHz and up to very high DMs (Kaspi & CHIME/FRB Collaboration 2017; Ng et al. 2017). The Aperture Tile in Focus (Apertif) experiment on the Westerbork telescope will continuously search thousands of synthesized beams at 1.4 GHz (van Leeuwen 2014). ASKAP (Bannister et al. 2017) and UTMOST (Caleb et al. 2017) are also expected to have high detection rates, searching many beams with high duty cycles. As a result, we will go from roughly five new FRB detections per year (2012–2017) to, potentially, thousands (>2019). This will also correspond with an orders-of-magnitude increase in the number of false-positive candidates, meaning that the generation of such events must be mitigated, and the process of sifting through them must be automated.

In pulsar searching, the problem is arguably worse due to the larger number of parameters involved, like rotation period and its derivatives. Over the last decades, the ranking of pulsar candidates has involved an initial step of selection through simple heuristics, the main one being the peak signal-to-noise ratio (S/N) of the frequency-collapsed pulse profile. Thereafter, astronomers go through the ordered list of candidate plots looking for further pulsar signs, such as broadband, properly dispersed signal; a sharply peaked (not sinusoidal) folded profile; and steady emission throughout the observation. An experienced pulsar astronomer can average one to two plots per second, and human brains are very capable of singling out the most promising candidates. But modern multibeam pulsar surveys and the increasing bandwidths and new frequencies outside of the radio-quiet protected spectrum are making this approach unfeasible. A telescope like LOFAR employs many hundreds of beams (van Leeuwen & Stappers 2010) and produces vast numbers of candidates. The LOFAR pilot surveys the LPPS and LOFAR Tied-Array All-Sky Survey

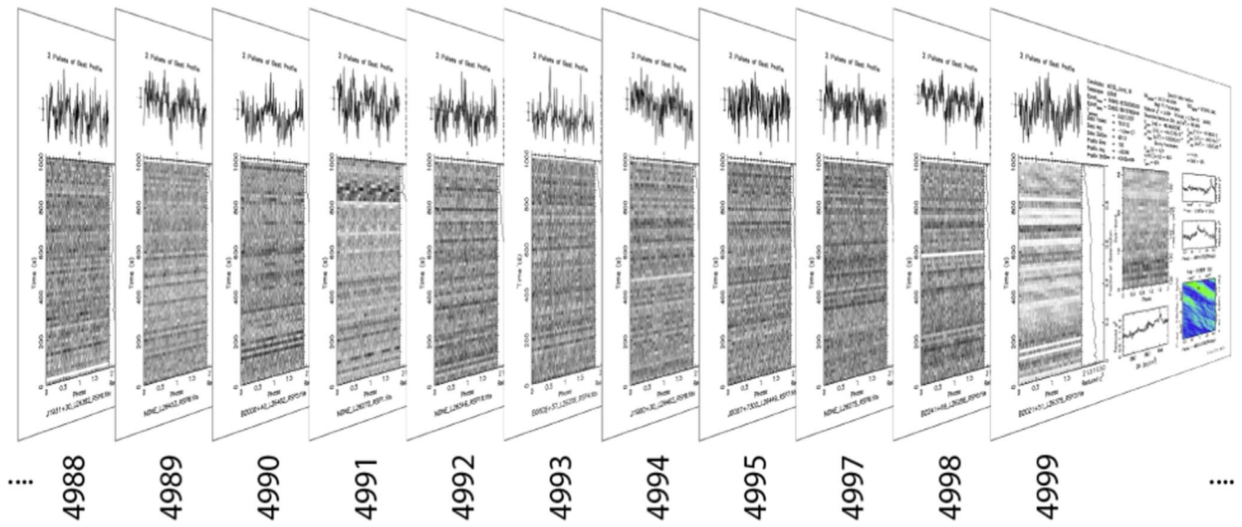


Figure 1. Small subset of the real-life diagnostic data used in the LOTAAS survey with LOFAR (Coenen et al. 2014). Out of ~20,000 candidates, pulsar J0613+3731 was found by eye in plot #4993.

(LOTAAS; Coenen et al. 2014) produced ~20,000 candidates that were ranked and perused by humans. This took about four person-days. It found the first two pulsars with LOFAR. Shown in Figure 1 is a subsection of the ranked list that included pulsar J0613+3731.

This approach is reaching the limits of what is efficient. For a long-integration, multibeam LOFAR search for young pulsars in supernova remnants, the lead author of S. Straal & J. van Leeuwen (2018, in preparation) checked, by eye, the staggering number of 140,000 periodic candidates plus about 15,000 single-pulse candidates. This amounted to three full-time person-weeks of time.

While efforts like the Pulsar Search Collaboratory (Rosen et al. 2013) have been successful in engaging hundreds of citizen scientists in ranking and analyzing candidates from GBT pulsar survey data, the overall person-power requirement remains unchanged and daunting.

The necessity of replacing manual inspection has led to a variety of approaches. Zhu et al. (2014) developed a sophisticated framework for pulsar candidate ranking using multiple machine-learning (ML) techniques to emulate a human expert inspecting diagnostic plots for tens of thousands of pulsar candidates from PALFA. They used convolutional neural networks (CNNs) in tandem with a support vector machine (SVM) on the pulsar candidates' two-dimensional (2D) arrays and artifical neural networks (ANNs) with SVMs on one-dimensional (1D) data products, like the pulse profile. Guo et al. (2017) utilized a convolution generative adversarial network (DCGAN) to improve the ability of deep CNN classifiers.

The LOTAAS uses 222 digitally formed tied-array beams per pointing, and its search pipeline reports the ~100 best periodic candidates per beam. As of early 2018, 1500 pointings had been observed, and over 30 million periodic candidate signals had been found. To manage these candidates, Lyon et al. (2016) built a decision tree–based ML classifier using a set of features from these periodic candidates.

The application of artificial intelligence (AI) to single-pulse classification is less well developed, in part due to the nascency of FRB and rotating radio transient (RRAT) science. Though there is significant overlap with candidate ranking in pulsar periodicity searches, the problem of single-pulse classification has several distinctions, particularly for upcoming multibeam real-time FRB surveys. These include the need for real-time classification for VOEvents and voltage dumps, as well as the usefulness of multibeam information. Devine et al. (2016) developed a method for identifying clustered groups of dispersed pulses, primarily in order to discover pulsars that might be missed by periodicity searches. There, 16 group features (e.g., start–end DM, maximum S/N) are used in six traditional ML algorithms to find the best combination of hyperparameters and classifier. Random forest (Breiman 2001) models perform best. The V-FASTR survey for FRBs on the Very Long Baseline Array (Wagstaff et al. 2016) also employs a random forest classifier. Beyond general pulse features similar to those used by Devine et al. (2016), the V-FASTR system is trained on informative local heuristics, such as the observing frequency (in relation to RFI), network dropouts (from processing artifacts), and the antenna coherence. In Arecibo's commensal FRB search, ALFABURST, Foster et al. (2018) built a training set of 15,000 events and extracted 409 features from each. Then, a random forest was again applied to group each trigger into one of nine classes. For the LOTAAS survey on LOFAR, Michilli et al. (2018b) adapted the Gaussian–Hellinger Very Fast Decision Tree used for periodicity classification (Lyon et al. 2016) and implemented a single-pulse search pipeline. It was trained on 3.5×10^4 labeled RFI instances and 1.8×10^4 single pulses from 47 known pulsars as recorded in the LOTAAS data and has discovered seven pulsars based on features like pulse width, DM, and S/N versus DM (Michilli et al. 2018b).

A next step, and challenge, in wide-field FRB searching will be the ALERT[3] survey on Apertif. A hierarchical series of beam forming starts with 39 compound beams formed on each of the phased-array feeds in the 12 dishes equipped with these. Every compound beam is next coherently beam-formed in 12 offset grating response beams; a refinement step of on-the-fly beam forming for removing chromatic sidelobe effects within this wide-bandwidth system finally increases the beam count by a factor of 6, for a total of ~2800 synthesized beams (Maan &

[3] www.alert.eu

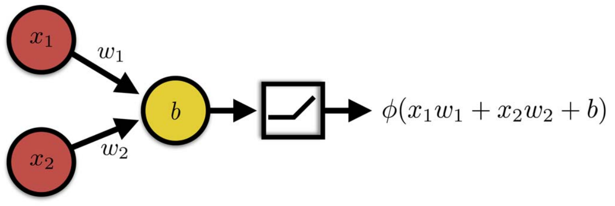
van Leeuwen 2017). These are searched in a real-time single-pulse pipeline powered by a large graphical processing unit (GPU) cluster (the Apertif Radio Transient System (ARTS); J. van Leeuwen et al. 2018, in preparation). The ALERT survey will run 24/7 for approximately 3 calendar yr. At 2× the number of beams, 5× the bandwidth, and more than 10× the on-sky time of LOTAAS, the number of single-pulse candidates produced in ALERT will exceed what can be inspected by humans.

In this paper, we apply deep learning to the problem of single-pulse classification for the first time. We develop a flexible toolkit that allows for the construction of hierarchical deep neural networks (DNNs) with multiple data products as inputs. Our approach will be useful for both multibeam surveys and single-pixel telescopes. Classification can be done very quickly on GPUs by using the highly optimized software library `TensorFlow`, which will be necessary if post-dedispersion real-time decisions are to be made. The paper is organized as follows. In Section 2, we introduce the key concepts of deep learning and discuss its advantages over more traditional ML algorithms. In Section 3, we describe our model's treelike architecture and show how arbitrary data inputs and feature extraction branches can be added to the network. We also offer tools to remedy the black-box problem. Section 4 discusses how we assemble a labeled training set, despite there being only two dozen FRBs to date. We then present the classifier's results in Section 5, showing that very high recall and accuracy can be achieved with a sufficiently comprehensive training set. Section 6 asks if it would be possible to replace real-time dedispersion back ends with a neural network classifier. We show that, somewhat surprisingly, forward propagation through a simple CNN could be faster than brute-force dedispersion. Simpler statistical approaches like current dedispersion algorithms are nearly optimal in signal recovery, but real-time AI-based classification may have higher precision under the right circumstances.

2. Deep Learning

Within the concentric circles of AI, ML has made the most progress in recent decades. The term ML refers to a class of tools that aims to let computers learn without being explicitly programmed. Representation learning is a further subset of ML whose goal is not only to model the mapping from input features to output but also to actually discover the feature (or representation) itself (Goodfellow et al. 2016). Representation learning circumvents the limitations of "feature engineering," in which data-specific features must be chosen by hand. This often requires domain expertise and can be time consuming. With real-world data, extracting the salient features from input data tends to be difficult. The last subset in these concentric circles, deep learning, helps with the representation problem by building complexity out of multiple, smaller representations (Lecun et al. 2015; Goodfellow et al. 2016).

A DNN is typically just a neural network that has multiple hidden layers. The DNNs make use of the "multilayer perceptron," a combination of artificial neurons and connections between them. Each subsequent hidden layer represents higher levels of abstraction of the input. An example of such a network is shown in Figure 2. The perceptron has weights $\boldsymbol{w} = (w_1, .., w_n)$ corresponding to each of the n connections between the input data, $\boldsymbol{x}$, and a neuron, as well as a single offset value, b.



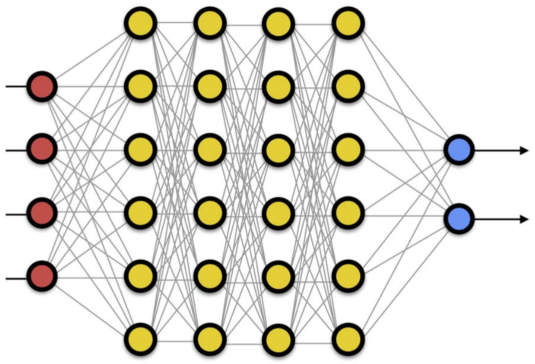


Figure 2. Schematic diagram of a perceptron (top) and a collection of perceptrons combined to form a neural network (bottom). The yellow nodes are artificial neurons, the red are input data, and the blue are output classes. The perceptron's output is a nonlinear function of the input vector, $\boldsymbol{x}$, projected onto the weight vector, $\boldsymbol{w}$, with some offset, b. The network shown is a DNN because it has multiple hidden layers.

The perceptron computes a linear combination of the input with the weights, such that

$$z = \boldsymbol{x} \cdot \boldsymbol{w} + b. \quad (1)$$

A nonlinear activation function is then applied to z, such that the output of the perceptron is some function $\phi(z)$. Such activators must be nonlinear; otherwise, no matter how many hidden layers are in a network, the output would simply be a linear function of the input. It is also a problem that a linear activator's gradient is independent of the input, making training via gradient descent impossible. Common examples include the logistic function,

$$\phi(z) = \frac{1}{1 + e^{-z}}; \quad (2)$$

a hyperbolic tangent,

$$\phi(z) = \tanh(z); \quad (3)$$

and a rectified linear unit (ReLu),

$$\phi(z) = \begin{cases} z & \text{for } z \geqslant 0 \\ 0 & \text{for } z \leqslant 0. \end{cases} \quad (4)$$

In this work, we mostly use ReLu functions, which have been empirically found to be highly effective (Maas et al. 2013).

It is easy to see how complex functions could, in principle, be modeled by finding the right weight and offset parameters for each connection and neuron, given enough labeled input data. What is less obvious is why this can be done with relatively few parameters. The size of the input space of a 100×100 gray-scale image is 256^{10000}, yet in many cases, the

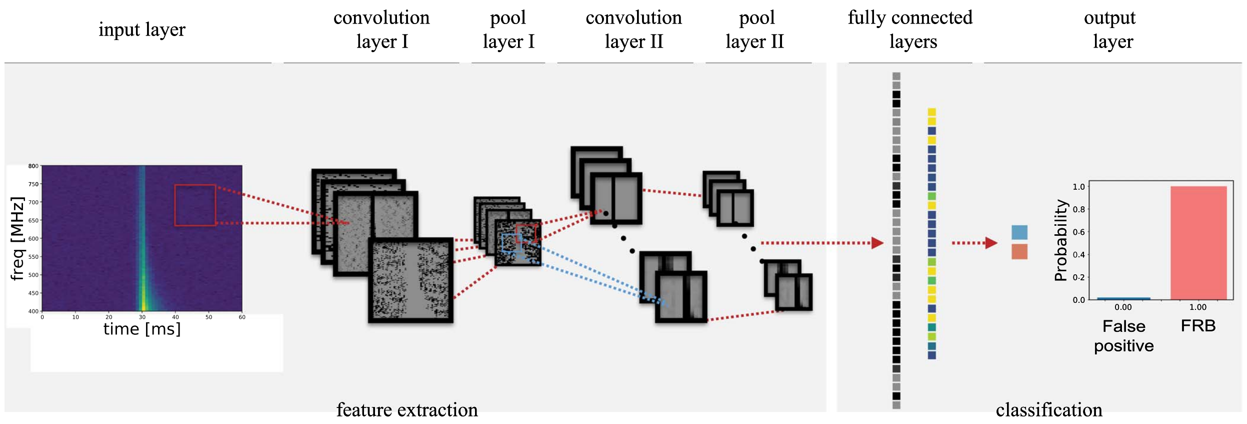


Figure 3. Example of our CNN architecture for the frequency–time array of a dedispersed FRB. The input image is a simulated strong, scattered burst, artificially bright for the sake of this example. The input data are convolved with 32 different convolutional kernels, the results of which are fed through a nonlinear ReLu function, then reduced in a max-pooling layer. Sixty-four more kernels are then applied to the binned arrays, and pooling is done again by taking the maximum value in each 2 × 2 bin. Each successive layer in the network's feature extraction component is meant to discover features at higher levels of abstraction. We have included in this figure the real activations of a trained model for this input image. These give an idea of what the neural network's classification is based on and can help with the "black-box" problem. After the fully connected layers, a probability is assigned to the trigger's likelihood of being an FRB.

mapping from input image to output class can be well approximated by ∼millions of parameters (Lin et al. 2017).

Irrespective of *why* deep learning has been so successful, its advancement of AI in recent years is undeniable. Deep learning has led to quantum leaps in self-driving car AI, superhuman image recognition, natural language processing, and machine translation (Goldberg 2015; Lecun et al. 2015; Goodfellow et al. 2016).

One type of network, the CNN, has proven particularly powerful. Such architectures use convolution along with pooling to extract high-level features from input data. In the case of image recognition, an input image is typically convolved with multiple different kernels, which are meant to find structure in the data and identify relevant attributes. A nonlinear activation function is then applied to the multiple convolved images, or "feature maps." Each convolution step is followed by a "pooling" layer. The pooling can be as simple as taking the maximum pixel value in a small region (max pooling) and is meant to act as a summary statistic, allowing for some translation invariance and robustness against noise (Goodfellow et al. 2016). An example of our CNN is shown in Figure 3 with the real activations of an input dynamic spectrum generated by a trained model.

3. Multi-input CNN

The classification of dedispersed single-pulse candidates is slightly different from other problems to which CNNs have been applied. For example, training a model for image recognition of, say, different breeds of dogs requires building a network that can learn a very large and complex image space based on photos with effectively infinite S/N per pixel. The FRBs occupy a much smaller volume of image space, but the S/N per pixel is ∼1. This turns out to be considerably less difficult than some other applications. Therefore, we can achieve high recall and precision with modest-sized training sets (tens of thousands of triggers) and relatively few layers.

3.1. Frequency–time Data

The most informative input data array is the frequency–time intensity data, or dynamic spectrum. This input lends itself well to a 2D CNN, where image topology is preserved. In other ML algorithms, such as SVMs, input data are flattened into a 1D vector, and the image's 2D structure is partially lost. Dedispersion algorithms search a frequency-collapsed time series, triggering on outliers in one dimension. Therefore, at a single DM, valuable spectral information is thrown out where most false positives from thermal noise will not look like a broadband pulse. By applying a deep CNN to the dynamic spectrum image, the model can discriminate based on frequency structure.

Employing a deep CNN has a similar advantage over the application of image analysis techniques such as the Hough transform (cf. Aulbert 2007). As these transforms are engineered to maximize the total response to a predefined feature shape (in the case of single-pulse detection, a line in the dedispersed dynamic spectrum), the results do not contain (spectral) information encompassed within that shape.

Our dynamic spectrum model is a CNN with two convolutional layers, two max-pooling layers, and two fully connected layers. We preprocess input data by demanding that each trigger have unit variance and zero median. We find that input frequency–time arrays of shape 32 × 64 allow for sufficient signal per pixel, but there is flexibility in the resolution of the input image.

A scaled-down example of this architecture is shown in Figure 3. The figure not only allows for visualization of the network's architecture but is also a way of peeking inside the model and looking at each hidden layer's activations. By saving the trained network's weights and convolutional kernels, a given input array can be forward-propagated through the model to produce activations at each layer. The activations give one an idea of what the neural network "sees" in a given hidden layer, which alleviates the black-box problem of DNNs and is helpful as a debugging tool. The CNN clearly tries to separate the input data's background noise from the features intrinsic to the FRB pulse, such as a scattering tail. Even so, the black-box problem is a significant issue for neural networks, and it has garnered growing interest from the deep-learning community. In Section 7, we discuss the opacity of DNNs beyond the visualization of activations.

3.2. Pulse Profile

We apply a 1D CNN to the pulse profile dedispersed to the DM that maximizes S/N. The DNN's first convolutional layer applies 32 length-5 kernels with strides of 2. After a 1D max pooling, another convolutional layer is applied with 64 length-2 kernels. The output is flattened and applied to a fully connected layer with 1024 neurons.

3.3. DM–time Data

The DM-transformed data is a DM–time array whose rows are the frequency-collapsed time stream at a given DM. Broadband, dispersed pulses will show up as a small island of preferred DM–time pairs exhibiting a bow-tie pattern due to a degeneracy between optimal DM and pulse arrival time. For this input, we also use a simple 2D CNN with DM–time arrays of dimension 100×64. An example is shown in the third row from the top of Figure 4.

3.4. Multibeam Detections

Most upcoming competitive FRB surveys will search multiple beams simultaneously. Objects beyond an antenna's far-field limit are not expected to be seen in more than a couple of adjacent beams, whereas terrestrial RFI can be detected in many nonneighboring beams. Other groups have taken this into account by rejecting triggers that showed up in unexpected beam permutations.

We allow our model to learn such permutations without explicitly telling it a given telescope's on-sky beam configuration. This was done using a simple feed-forward neural network whose input data is a 1D length-$N_{\rm beam}$ vector of detected S/N per beam. If no event was found above the cutoff significance, an S/N of zero is assigned to that beam. After training, the model learns which beams ought not to trigger simultaneously and which combinations are acceptable for a real astronomical detection.

3.5. DNN Tree

We developed a multi-input neural network to which arbitrary additional nets can be appended. We will typically refer to this as the "hybrid" model, in that it takes inputs from multiple base-level neural networks. The idea is to extract features from each input data product independently, since a given burst's salient characteristics will depend on the space in which it is being viewed. The multiple networks can then be concatenated at the classification layers (in our case, fully connected layers after convolution), creating a hierarchical treelike neural network, shown in Figure 4. This setup preserves the ability to use a single input, depending on the availability of data products or the relative efficacy of the various models.

The first three data products we use in Figure 4 are not independent. Indeed, their information content is highly redundant: the 1D pulse profile is simply the dedispersed frequency–time array collapsed along the frequency axis, and the DM–time array is the frequency–time data after the DM transform. And yet, empirically, better results are achieved by including combinations of the three than any individual one. This is because the feature extraction step is imperfect, so projecting the data in different ways allows the networks to detect different modes. The same is true for human classifiers. When sifting through pulse candidates, one often looks at multiple statistics and figures with overlapping information.

4. Training Data

The two conditions that have allowed deep learning to thrive in this decade have been the availability of large, labeled data sets and computers that can train multilayer models in a reasonable amount of time. Despite the high all-sky event rate of FRBs, only a couple of dozen events have been discovered to date (Petroff et al. 2016[4]). This presents a problem that does not exist for pulsar candidate classification. The small catalog of real events is probably not yet a representative sample of the underlying burst population, nor is it big enough to build a meaningful training set for ML, deep or otherwise.

This means that either bursts can be simulated, single pulses from Galactic pulsars can be used as an approximation, or a combination of both. In this work, we choose the latter, but with a training set dominated by simulated events injected into real data rather than pulsar pulses. We simulate most of our "true positives" but use only false positives that have been generated in real surveys and labeled by eye. We construct two data sets, one based on CHIME Pathfinder data and one based on Apertif data.

We do not use single pulses from Galactic pulsars as a primary training set for the following reasons. For one, it helps with the class-imbalance problem, in which a training or test set has unequal fractions of each classification category. More importantly, even though a large number of pulses can be collected from individual pulsars, the variation within FRBs appears to be larger than the pulse-to-pulse variation from a single pulsar, in terms of pulse characteristics (width, scattering, frequency structure, etc.). Therefore, if a survey can only detect single pulses from a handful of pulsars, the model will overfit for the properties of those sources.

Even if the survey is sensitive enough to see single pulses from large numbers of sources, differences between FRBs could also be larger than the variation between Galactic pulsars, given the extreme conditions they appear to live in (Masui et al. 2015; Michilli et al. 2018a). For example, FRB 121102 shows frequency structure on at least two different scales and has bursts ranging in duration from 30 μs to several ms (Michilli et al. 2018a). Finally, while dedispersed FRBs are qualitatively similar to single pulses from nearby pulsars ($\sim$millisecond-duration, broadband, etc.), there may be systematic differences that are not obvious or visible but would bias the learner. In a simulated set, there is more control of and insight into the parameters producing the set that we train against. Therefore, for the sake of prudence, we have decided to inject simulated FRBs into real data with a wide range of parameters so that whatever the true distribution of FRB properties is, they will fall under the umbrella of our training set. We test this assumption in Section 5 by training a model only on simulated bursts and then classifying a collection of single pulses from Galactic pulsars.

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

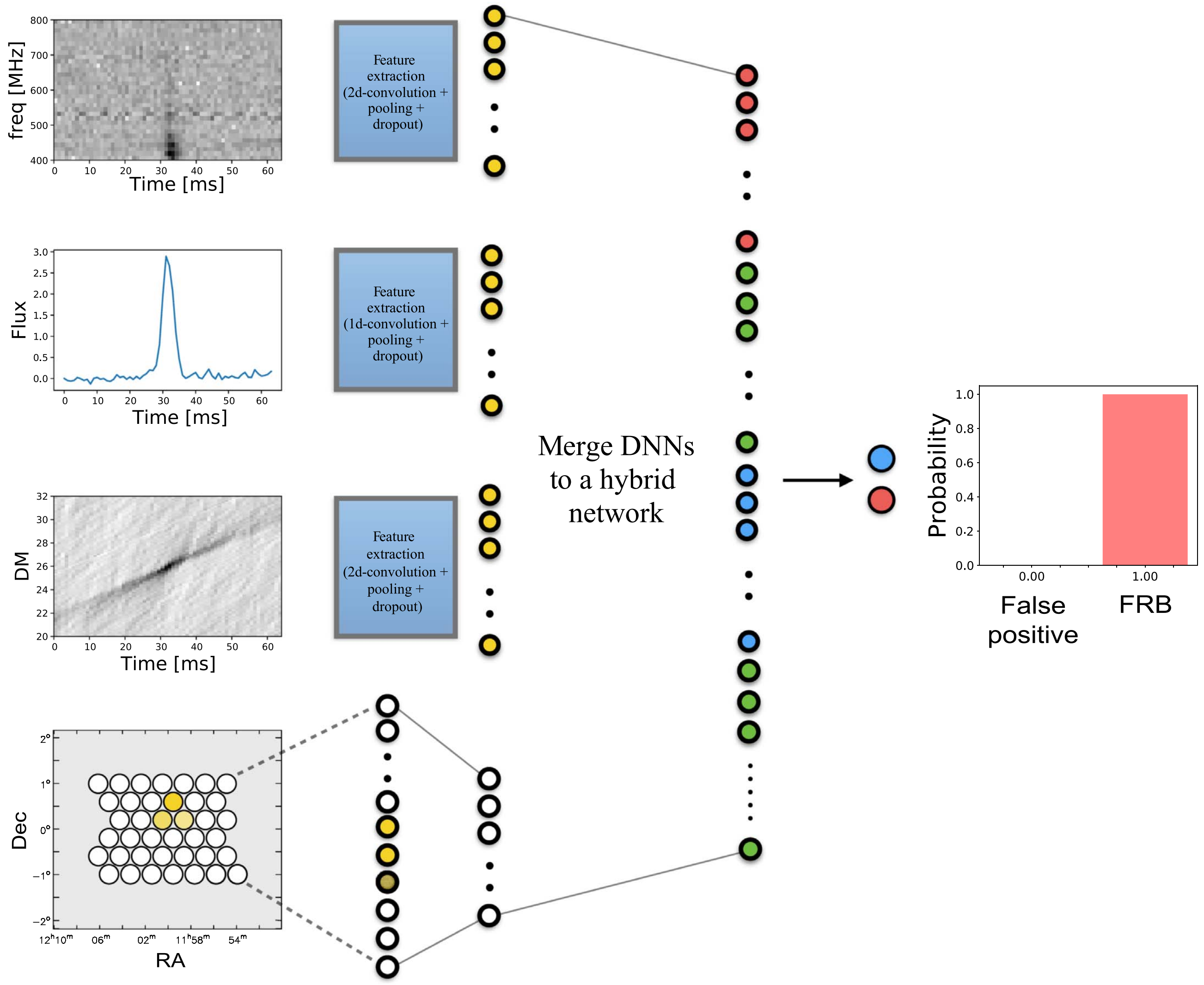

freq [MHz]
Time [ms]
Flux
DM
Dec
RA
Feature extraction (2d-convolution + pooling + dropout)
Feature extraction (1d-convolution + pooling + dropout)
Feature extraction (2d-convolution + pooling + dropout)
Merge DNNs to a hybrid network
Probability
False positive
FRB

Figure 4. Hierarchical hybrid neural network built from concatenating multiple nets after their feature extraction layers, creating a large fully connected layer resulting in a single binary classification for all inputs. Here we use as inputs the dedispersed frequency–time intensity array, frequency-collapsed pulse profile, DM–time array, and multibeam detection S/N, but our framework allows for any combination of these, as well as additional input data products and networks. Since the full, merged DNN is trained together, the model will learn the relative importance of each input. For example, the dedispersed intensity array will tend to have more predictive power than the multibeam statistics; thus, the former will have a greater influence on the output probability.

While we choose to simulate our true positives, false-positive triggers should not be simulated. Events generated by RFI, thermal noise, or dropped packets occupy a much larger volume of image space than single pulses from FRBs, RRATs, or pulsars. Simulating RFI triggers would be difficult, since there is no good model that describes such events. On top of that, each instrument will produce a different set of false positives due to their disparate RFI environments and signal-processing back ends. Conversely, FRBs can be modeled with far fewer parameters. Though they can suffer to varying degrees from temporal scattering, frequency scintillation, and DM smearing, these effects can, in principle, be accounted for. By including in one's training set a large collection of events that plausibly samples the full phase space of FRBs, a sufficiently sized neural network can learn to identify a wide range of pulses. Casting such a wide net should catch true single pulses.

We have built a training set from the 1268 hr of data in the CHIME Pathfinder incoherent-beam FRB search plus simulated events injected into those data (Amiri et al. 2017). We use 4650 events that triggered the dedispersion pipeline with an S/N above 10 but were found to be false positives after inspection by eye. We have an equal number of true positives, of which ∼4550 are simulated and ∼100 are either giant pulses from the Crab or B0329+54 single pulses. The simulated FRBs drawn from the distributions are described in Section 4.1. A larger set of astronomical true positives were separated and used later in the verification that the model was able to classify correctly on nonsimulated single pulses. For our Apertif model, the training set consists of 21,246 candidates, half of which are known false positives. Of the remaining triggers, roughly 9800 are simulated FRBs added to real data, along with over 800 single pulses from Galactic pulsars.

4.1. Simulation

We simulate FRBs in one of two ways. The preferred approach is to randomly inject events in the data using the real-time tree dedispersion pipeline `burst_search`.[5] Another way is to add simulated FRBs to real background data that have already been dedispersed to a random DM. The injected, simulated bursts differ from Galactic single pulses in that they have wider ranges of widths and fluences and more varied frequency structure. The single pulses, however, sometimes exhibit temporal structure that we do not try to imitate in our simulations.

We calculate the pulse profile at each frequency by convolving a Gaussian with a scattering profile,

$$s(t) = \frac{1}{\tau_\nu} e^{-t/\tau_\nu}, \tag{5}$$

where τ_ν is the scattering timescale at a frequency ν and is given by

$$\tau_\nu = \tau_0 \left(\frac{\nu}{\nu_{\rm ref}} \right)^{-4} \tag{6}$$

for a reference frequency $\nu_{\rm ref}$. The Gaussian is taken to have width t_I,

$$t_I^2 = t_i^2 + t_{\rm samp}^2 + t_{\rm DM}^2, \tag{7}$$

where t_i is the intrinsic pulse width, $t_{\rm samp}$ is the sampling time, and $t_{\rm DM}$ is the DM-smearing timescale.

Burst fluence is drawn from a Euclidean distribution, resulting in an S/N distribution of pulses that is also approximately Euclidean. This is consistent with current data (Oppermann et al. 2016; Amiri et al. 2017), but we find that changing the input brightness distribution does not significantly affect the model's performance. For the CHIME Pathfinder set, widths are assumed to follow a lognormal distribution with a mean of 1.6 ms, which is that survey's sampling time, resulting in widths between 1.6 and 50 ms. This is partly motivated by the empirical fact that FRB widths appear to tightly hug the DM/$t_{\rm DM}$ curve, such that many bursts are unresolved in time and have effective widths close to the DM-smearing timescale (Ravi 2017). Scattering measure is loguniform, in a way that roughly one in three bursts is noticeably temporally scattered. Frequency scintillation is included via intensity modulation across the band using the positive half of a sinusoid with random phase and random decorrelation bandwidth. The scintillation bandwidth distribution is such that only about one-third of simulated bursts show discernible frequency variation, consistent with the current population of detected FRBs. We take a uniform distribution of the spectral index, γ, between −4 and +4, where $F_\nu \propto \nu^\gamma$.

After the signals are injected, events are kept if their S/N falls between 8 and 80. Ultrabright events are discarded because they do not add much predictive power to the trained model; if a 500σ event is found in the data, a model that has learned 80σ events will still find it. All data are preprocessed to have unit variance and zero median. Uniformity in the treatment of both the simulated FRBs and the detected false positives is important, because otherwise the binary classifier will learn based on trivial differences like noise rms or power offset. In Figure 5, we show examples of the FRBs generated.

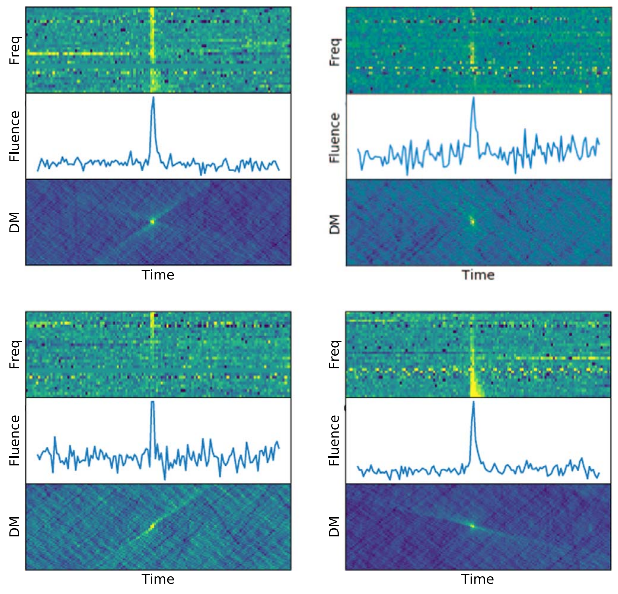


Figure 5. Four examples of simulated FRBs injected into real data from the CHIME Pathfinder survey. By combining thousands of these true positives with known false positives, we have built a large labeled training set. The three panels in each subfigure are the frequency–time intensity array of the dedispersed pulse, the frequency-averaged pulse profile, and the DM–time intensity array (top to bottom). Combinations of these data products can be used as inputs to the multi-input neural net described in Section 3.

The exact parameters of the distributions chosen are, of course, tunable and will be subject to change depending on the survey for which the model is being built. The simulation tools are available on github[6] (Connor 2018).

4.2. RFI Excision versus Classification

The RFI will be an appreciable problem for all upcoming FRB surveys. Though we may want to mitigate RFI as much as possible, there exists a trade-off between pre-dedispersion RFI cleaning and false-positive rejection post-triggering. If one wants to minimize the number of RFI events triggered by a dedispersion algorithm, then data preprocessing needs to be done thoroughly. However, this runs the risk of overcleaning the data and removing astronomical events. For example, if one of the steps in the RFI excision pipeline is a standard σ cut in which samples above, say, 3σ in their local neighborhood are removed, then events like the Lorimer burst (Lorimer et al. 2007) or FRB 150807 (Ravi et al. 2016) could be missed, especially if the events fluctuate in frequency and time. Another approach would be to preserve as many triggers as possible by doing modest or no RFI cleaning, allow large numbers of false positives to trigger the dedispersion pipeline, and only make a final decision after the triggers have been classified by the ML algorithm. This would make sense if one had high confidence in one's classifier; otherwise, real signals might drown in the flood of false positives. The solution is probably somewhere in between the two extremes: data ought to be cleaned enough that the RFI within a time-frequency block containing an FRB does not decrease the event's S/N, and a balance must be struck between the number of triggers generated and the risk of missed events, i.e., false negatives,

[5] https://github.com/kiyo-masui/burst_search

[6] https://github.com/liamconnor/single_pulse_ml

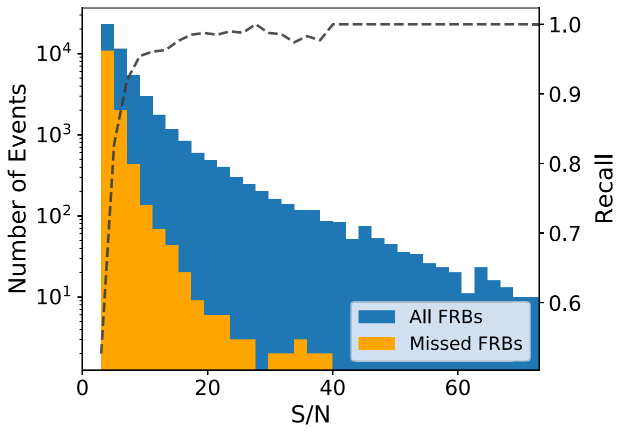


Figure 6. Statistics of missed FRBs as a function of S/N from the CHIME Pathfinder. The histogram shows the distribution of 50,000 simulated FRBs in the test set (blue), as well as the events from that test set that were mislabeled as RFI by our frequency–time 2D CNN (orange). The false-negative rate goes to 0.5 for low S/N, as expected, since a binary classifier with no predictive power will classify correctly half of the time. The fraction of recovered events, or recall, gets close to 1 for high S/N.

which will require experimentation and be survey-dependent. For the CHIME Pathfinder data used in this paper, some RFI cleaning had to be done, otherwise a trigger above 10σ was produced every couple of seconds. For Apertif, where the RFI environment between 1250 and 1550 MHz is quite clean, only a few persistent-RFI frequency channels were masked, and no further cleaning was done.

5. Results

In order to assess our model's performance, we use standard metrics based on the confusion matrix. Here "accuracy" corresponds to the fraction of classified events that were labeled correctly, "precision" is the ratio of true positives to the number of events classified as positives, "recall" is the fraction of true events that were labeled as such, and "F1" is the harmonic mean of recall and precision. Using TP, TN, FP, and FN as the number of true positives, true negatives, false positives, and false negatives, respectively, the metrics are given by,

$$\text{acc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \quad (8)$$

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad (9)$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad (10)$$

$$\text{F1} = \frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}}. \quad (11)$$

We care about the recall rate because this determines the probability of missing an FRB. However, the precision is also important in case of real-time triggering. If voltage data are to be written after the dedispersion pipeline is triggered, as with ASKAP or UTMOST, then one must be sure that most of those events really are FRBs. The same is true for email notifications and VOEvents (Petroff et al. 2017), such as when Apertif will

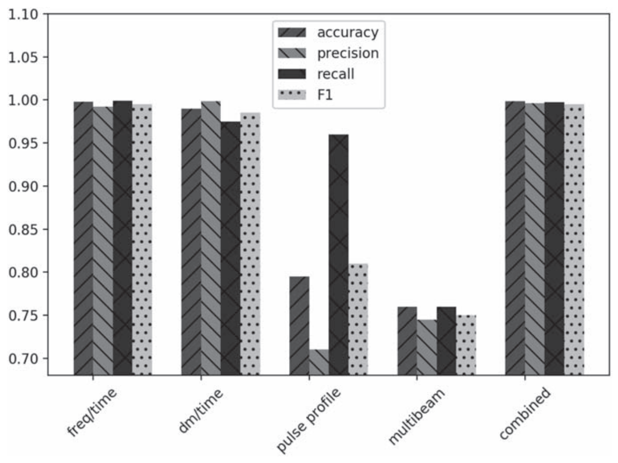


Figure 7. Comparison of performance metrics between the four input data products using a subset of Apertif FRB candidates. As expected, the time/frequency array (dedispersed dynamic spectrum) has the most predictive power, and in this case is not significantly outperformed by the combined information. However, in special cases like multibeam peryton detections or low-DM events, the extra information is critical, even if they do not affect the overall statistics.

trigger LOFAR's transient buffer boards for low-frequency localization.

In Figure 6, we plot recall as a function of S/N. As expected, at very low S/N, the model loses its predictive power: recall drops to 50% because the algorithm is making a random guess at binary classification. However, above $\sim 8\sigma$, the fraction of missed FRBs is quite flat and low, with recall and accuracy above 99%.

5.1. Relative Input Performance

The inputs of our model can be trained together, as seen in the hybrid model in Figure 4, or with permutations thereof. This includes using each input independently as its own neural network. Since there may be instances where only one or a subset of the total input data arrays is available, it is valuable to compare them against each other in an ablative study.

In Figure 7, we show results from the four input data products described in Section 3 using a smaller subset of events from Apertif for which DM–time inputs were available. This consisted of 1400 triggers. Unsurprisingly, the 2D dedispersed dynamic spectrum consistently provides the most predictive power, followed by the DM–time array. The 1D pulse profile has a relatively low precision (i.e., high false-positive rate) because classification was already done in that space via the dedispersion detection algorithm. This is consistent with the experience of human experts inspecting diagnostic plots. Averaged over all events, the combined information does not significantly outperform the 2D dynamic spectrum. It does, however, prove critical for special cases of false positives, such as multibeam peryton detections (Petroff et al. 2015b) and low-DM events that the frequency–time model can easily classify as an FRB. Conversely, true events that are too weak to be persuasive in the dynamic spectrum may have a signal distribution in DM–time space that looks like a real FRB. That is also true of multibeam information for, e.g., a pulse that triggers in three adjacent beams at $\lesssim 8\sigma$, which may convince the hybrid model that the event is real.

Table 1
Comparison of the Performance of the Single-pulse ML Model Currently Described in the Literature

Publication	Method	Survey	Accuracy	Precision	Recall
Devine et al. (2016)	Random forest (SMOTE)	GBT	…	54%	83%
Wagstaff et al. (2016)	Random forest	V-FASTR	96%	97%	96%
Foster et al. (2018)	Random forest	ALFABURST	>99%	92%	96%
Michilli et al. (2018b)	Decision tree	LOTAAS	99%	98%	98%
This work	Random forest (freq-time)	Apertif	97%	97%	97%
This work	SVM (freq-time)	Apertif	96%	94%	98%
This work	DNNs	Apertif	>99%	>99%	>99%
This work	DNNs	CHIME Pathfinder	>99%	>99%	>99%

Note. Metrics from Devine et al. (2016) are taken from their Table 3 for their best-performing classifier. For V-FASTR, values were derived using Equations (8)–(11) from data presented in Table 1 of Wagstaff et al. (2016). Values for Foster et al. (2018) derived from their Figure 3 confusion matrix. Performance for Michilli et al. (2018b) is based on Table 2 of that forthcoming publication. For empty cells, metrics could not be calculated from the published material. Values below 99.0% are rounded to the closest integer percentage.

5.2. Comparison against Other Single-pulse ML Approaches

In a domain where the ratio of apparently worthwhile candidates can be easily over 10^5:1, a 0.1% improvement in classifier performance (especially precision) can save multiple hours of human inspection for a single 1 hr observation. Even with its on-sky time of about 400 hr yr^{-1}, equal to a 5% duty cycle, the LOTAAS[7] survey on LOFAR already produced an estimated more than 1 billion (10^9) single-pulse candidates (Michilli et al. 2018b). For a 24/7, 100% duty cycle survey such as Apertif, precision below 98% already makes it impossible for a single person to keep up with all candidates ranked True. In Table 1, we thus compare performance metrics for existing ML implementations against the classification presented in this work.

The results in Table 1 offer a useful but imperfect comparison between various ML efforts in single-pulse classification. Each algorithm was applied to a different data set from different survey environments and with disparate signal-processing back ends. For example, V-FASTR used a low-S/N cutoff of 5, which could potentially deflate their accuracy. To remedy this, we hope to carry out a multi-collaboration benchmarking test in future work, in which different algorithms could be applied to one or multiple common data sets, allowing for improvement of all classifiers. This is discussed further in Section 7.

While it is beyond the scope of this paper to implement other groups' algorithms on our own data directly, we have tested more traditional ML algorithms against our DNNs. We applied a random forest with 100 decision trees to the same data set of Apertif dynamic spectra that was used for the DNN tests, described in Section 4. Of the 21,246 candidates, 75% were randomly selected to train on, and the remaining nonoverlapping 25% were used as a test set. The random forest achieved roughly 97% in precision, recall, and accuracy. We also used an SVM that scored 94%, 98%, and 96% in the same three metrics, respectively. This performance is reasonably good, but the accuracy of 99.8% seen by our DNN on the same data corresponds to significant time saved if one has millions of triggers. We would also argue that even if such algorithms achieved the same results as our model, they should not necessarily be preferred over deep learning. Since no human expert has a full understanding of what separates an FRB from a non-FRB post-dedispersion, all ML models are fundamentally opaque at some level. Therefore, whatever algorithm is applied, its value will mainly come from its speed, ease of use, and performance as verified by cross-validation. For our purposes, `TensorFlow` was very easy to implement and is highly optimized—especially to our hardware—and faster at inference than the other classifiers we tried.

5.3. Validation with Pulsars

One must be cautious when including simulated events in an ML training set, particularly when the model will be used on real data. For this reason, we carried out tests in which classifiers trained only on simulated FRBs were used to predict the labels of real single pulses from Galactic pulsars. We did this for both Apertif and CHIME Pathfinder data independently, using their respective hand-labeled false positives in combination with simulated bursts to train their DNNs. The models were then used on separate data sets containing hundreds of Crab giant pulses and B0329+54 single pulses. In Figure 8, we show the output of our pipeline for the CHIME Pathfinder data set. For the CHIME Pathfinder, a recall of ~99% can be achieved. Our Apertif model was trained on ~20,000 candidates and applied to several hundred Galactic single pulses. It was able to recover 99.7% of these events.

5.4. Speed

For a given processor, training is always slower than classification, since evaluation only requires one forward propagation through the neural network. For the applications described in previous sections, neither is prohibitively slow, even on CPUs. This is due to the modest sizes of the neural networks we have used (described in Section 3). However, for real-time FRB surveys like Apertif, ASKAP, UTMOST, and CHIME, it may be necessary to "decide" on triggers quickly and without human inspection, for example, if voltages are to be saved or an alert is to be sent to another telescope. Therefore, low-latency classification will be required. Our measurements of execution times for training and classification, shown in Figure 9, indicate that both can easily be done in real time on either CPUs or GPUs. A single GTX Titan X can classify 10^4 candidates in under 1 s. For Apertif, the ARTS cluster contains 164 GTX 1080 Ti GPUs, which are each about twice as fast. Each Apertif compound beam (Figure 10) is reduced on a dedicated four-GPU server. The dedispersion and detection routines (`AMBER`[8]; Sclocco et al. 2016) require the

[7] www.astron.nl/lotaas/

[8] https://github.com/AA-ALERT/AMBER

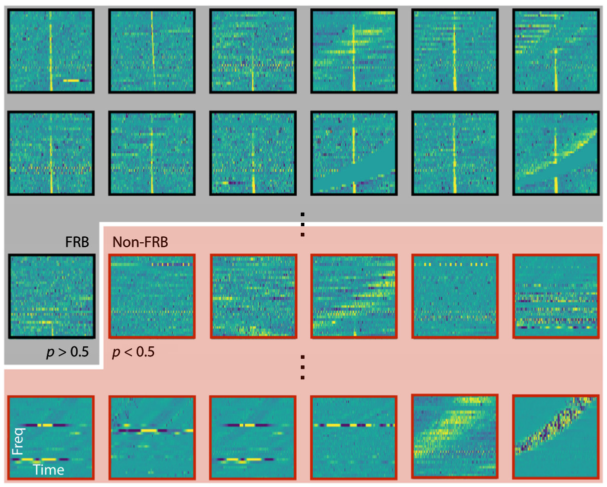


Figure 8. Ability of a model trained on simulated bursts to correctly identify Galactic pulsars. Our DNN classifier's output is a list of triggers ranked by their probability of being an FRB. Shown are frequency (ordinate) vs. time (abscissa) arrays of the test triggers, identical to the top panels in Figure 5. The set includes real pulsars. Events that the classifier thinks are FRBs ($p > 0.5$) are boxed in black, and non-FRBs are boxed in red. The most likely events are in the top two rows; the marginal events are in the middle row, where the predicted labels transition from "FRB" to "RFI"; and the bottom row shows the events least likely to be true positives. In this case, we trained on 4850 known false positives from the CHIME Pathfinder and 4850 simulated FRBs. The test data are a separate set of several hundred triggers consisting of known false positives, single pulses from B0329+54, and giant pulses from the Crab. The Pathfinder classifier gets fewer than 1% wrong. Our classifier trained on Apertif data achieves 99.7% recall.

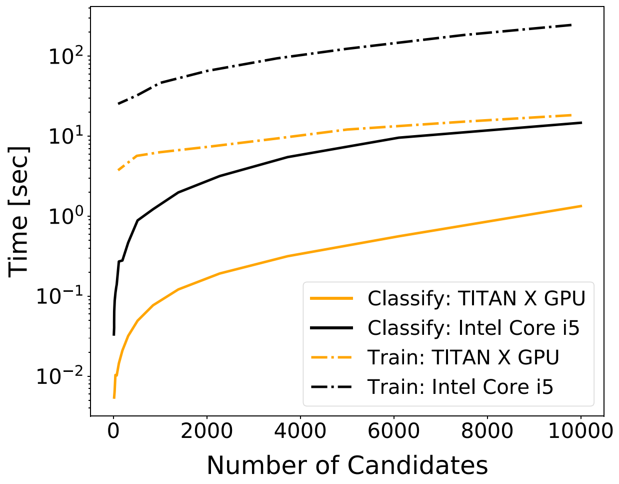


Figure 9. Classification and training time as a function of the number of dynamic spectra candidates. On a GTX Titan X GPU, classification takes ~100 μs per frequency–time array, using the network architecture described in Section 3 with the `TensorFlow` back end of `keras`. On a 2.9 GHz Intel Core i5-5287U CPU, the classification time is a few ms per candidate.

usage of only two of these. After RFI mitigation, the trigger levels in the single-pulse S/N can be set in AMBER. Allowing of order 10 candidates to be marked as interesting per second and per compound beam would amount to >10 million candidates per day, among which may be of order a single FRB. Even this liberal false-positive strategy could easily be further classified by the hybrid network running on a single GPU on the central ARTS server and VOEvent issuer.

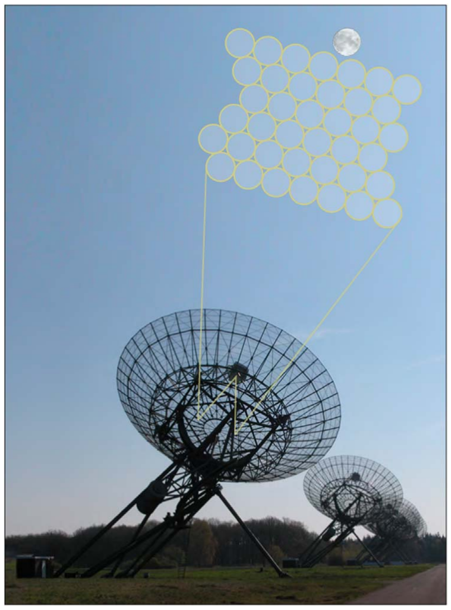


Figure 10. Layout of the 39 compound beams that can maximally be formed at full bandwidth. This pattern provides the most uniform sensitivity possible over the celestial sphere (K. Hess 2018, private communication).

5.5. Phased-array Feed Simulation

To test the efficacy of including multibeam detection information in our DNN, we simulated the on-sky response within the 39 compound beams in the Apertif phased-array feed (Figure 10). We randomly scattered 10,000 FRBs within this multibeam setup, currently planned for the imaging and time-domain surveys with Apertif.[9]

Events were drawn from a Euclidean flux distribution, and "detections" greater than 6σ were recorded. Though the fraction of multibeam detections depends on the FRB brightness distribution, we found no significant differences when reasonable non-Euclidean values were used. The RFI was assumed to show up in a random number of beams ranging from 1 to 39 following a lognormal distribution such that 75% of events are detected in five or more beams. The training set was then taken to be 15,000 simulated length-39 S/N vectors. The feed-forward neural network was then tested on the remaining 5000 events, with an accuracy of ~75% and a slightly higher recall, when using the described configuration, shown in Figure 7. This is much worse than the dynamic spectra CNN (accuracy above 99%), but that is to be expected. Seeing an event in only one beam does not preclude it being a false positive, in the same way multibeam detection does not guarantee that the event was RFI. The multibeam data add orthogonal, complementary information to pulse shape and frequency structure and were shown to be a valuable part of the hierarchical hybrid neural network (bottom row of Figure 4). A real-world example is the detection of perytons at Parkes, which looked very similar to pulses dispersed by cold plasma but showed up in all beams because they were generated locally (Petroff et al. 2015b). Our DNN would likely classify them as real on a per-beam basis, but the multibeam information would offer evidence that they were terrestrial.

[9] http://www.astron.nl/radio-observatory/apertif-surveys

6. Replacing Dedispersion with DNNs

Advances in signal processing allow the data volumes of future radio surveys to grow at considerable rates. This data deluge will likely outpace the ability of end-user astronomers to study all results. Interferometers like the Square Kilometre Array (SKA; Smits et al. 2009) and next generation Very Large Array (ngVLA)[10] will not be able to search the newly available regions of parameter space simply by increasing person power—new techniques must be developed. It is therefore reasonable to ask how advances in ML might aid this endeavor.

We investigate whether a real-time DNN classifier could be used for transient detection. For example, might it be possible to completely replace dedispersion back ends with a pretrained neural network? Even though the classifier could learn to identify arbitrary signals and not just ν^{-2} sweeps, comparing to dedispersion provides a useful benchmark.

We can start by checking whether any deep classifier could even keep up with the high data rates involved in real-time transient detection. The data rates on modern multipixel radio telescopes are enormous, meaning data must either be searched in real time or binned down to lower resolution for offline processing. The scale of this challenge is exemplified by the ARTS, which continuously produces 225 Gbps of 300 MHz, 41 μs data that need to be searched in real time. A massive dedicated cluster with 164 GPUs (GTX 1080 Ti) is required to keep up with this data rate.

In the case of real-time classification of FRBs, the idea would be to train on large numbers of dispersed pulses, such that the model would learn to look for ν^{-2} sweeps, independent of their DM and with enough translational invariance to be insensitive to arrival times. This would supplant the need for dedispersion back ends, which calculate an S/N after collapsing in frequency for multiple trial DMs.

In the past, offline processing and lower data rates meant that brute-force dedispersion was sufficient. The brute-force algorithm requires summing N_f frequency channels for $N_{\rm DM}$ DM trials for all N_t time samples. Its computational complexity is $\mathcal{O}(N_f N_t N_{\rm DM})$ (Magro et al. 2011; Barsdell et al. 2012; Sclocco et al. 2014). Tree dedispersion applies a divide-and-conquer technique by taking advantage of the redundancy in dedispersion for nearby frequency channels (Taylor 1974). By using an FFT-like approach, the problem is reduced to a tree with $\log_2 N_f$ branches, allowing for a $\mathcal{O}(N_f N_t \log_2 N_f)$ complexity. A highly optimized CPU-based version of this algorithm has been implemented for CHIME's FRB search (Kaspi & CHIME/FRB Collaboration 2017). Other algorithms, like the fast DM transform (FDMT), exploit the same redundancy and attempt to maintain optimality (Zackay & Ofek 2017). This algorithm is used in ASKAP's FREDDA pipeline.

For a neural network like the one shown in Figure 3, forward propagation is simply a series of convolutions and matrix multiplications. The two computational bottlenecks are the input layer, in which the $N_f \times N_t$ array is convolved with n_{k_1} kernels, and the first fully connected layer. A fully connected layer with n inputs and m outputs scales as $\mathcal{O}(nm)$. This is because the output is given by

$$z = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}, \tag{12}$$

where $\boldsymbol{x}$ is the n-element input vector, $\boldsymbol{b}$ is a vector containing offsets of the m neurons in that layer, and $\boldsymbol{W}$ is an $m \times n$ matrix whose elements w_{ij} give the connection between the jth input and the ith neuron.

[10] http://ngvla.nrao.edu/

In the final convolutional layer, n_{k_2} arrays are created, one for each kernel in the second convolution. The final pooling step takes these n_{k_2} matrices and reduces them in size by a factor of $p_x\, p_y$ by mapping each box of dimensions p_x by p_y to a single pixel in the subsequent layer. Therefore, the input of the first fully connected layer is an unraveled vector of length

$$n_l = \frac{N_f N_t}{p_x^2 p_y^2} n_{k_2}, \tag{13}$$

since the original $N_f \times N_t$ array has been reduced in size twice by a factor of $p_x\, p_y$ through pooling. With n_{d_1} neurons in the first fully connected layer, calculating the activations of this component scales as

$$\mathcal{O}\left(\frac{N_f N_t}{p_x^2 p_y^2} n_{k_2} n_{d_1}\right). \tag{14}$$

This means that if the network's parameters are such that

$$\frac{n_{k_2} n_{d_1}}{p_x^2 p_y^2} < N_{\rm DM}, \tag{15}$$

then this layer can be computed faster than brute-force dedispersion. In the model we used for classification of already-dedispersed single pulses, p_x and p_y were both 2, $n_{k_2} = 64$ kernels were used in the last convolutional layer, and the first fully connected layer had $n_{d_1} = 128$ neurons. Thus, the inequality in Equation (15) would be satisfied for most brute-force searches, since the left side in our current model would be 512, whereas $N_{\rm DM} \sim 10^{3-4}$.

For dedispersion algorithms that are more optimized, such as subband or tree dedispersion, the balance in Equation (15) may be different; for tree dedispersion, the right-hand side term is $\log_2 N_f$, which is of order 10. So, in contrast to the brute-force approach, its computational intensity may be less than the DNN. That does not immediately imply, however, that the real-life performance of these optimized dispersion algorithms is proportionally faster. Dedispersion is a memory-bound algorithm for real-world parameters. Through data reuse, brute-force dedispersion can approach the execution time of the more optimized algorithms (Sclocco et al. 2016). Yet the fact that the matrix multiplications underlying the DNN are compute-bound can give the classifier a further real-life advantage on compute-biased accelerators, such as GPUs.

Forward propagation through the first layer of our CNN amounts to computing n_{k_1} convolutions. Convolution can be slow for two arrays of similar size. Using the brute-force method, this operation scales as $\mathcal{O}(N^{2D})$, where N is the input array's length and D is the number of dimensions. By invoking the convolution theorem, FFTs allow for a speed-up, scaling as $\mathcal{O}(N^D \log_2^D N)$. However, our case is different from these, since our first layer requires convolving an $N_f \times N_t$ array with a much smaller array, often with kernels of size 3×3 or 5×5. The convolutions can be lowered to matrix multiplications, which are highly optimized on GPUs, allowing routines like `cuDNN` and `cuda-convnet2` high arithmetic intensity and efficiency (Chetlur et al. 2014). If we have a filter tensor that consists of n_{k_1} kernels of size $n \times n$, then that can be reshaped to an array of dimensions $n_{k_1} \times n^2$. With batches of N_b data arrays, each of whose images are $N_f \times N_t$, the data tensor can be reshaped to an $n^2 \times N_b N_f N_t$ matrix. The convolution can then be computed as a matrix multiplication, which scales as

$$\mathcal{O}(n_{k_1} n^2 N_b N_f N_t). \tag{16}$$

Therefore, each frequency–time array takes, on average, $n_{k_1} n^2 N_f N_t$ computations after dividing out the number of arrays per batch. In our case, with 16 or 32 length-3 square kernels, our most expensive convolutional layer is faster than brute-force dedispersion, since $n_{k_1} n^2 \approx 10^2 < N_{\rm DM} \approx 10^4$. There are further techniques that allow a large DNN to be approximated by a smaller one, called compression. The deeper and/or wider model would be trained offline, and its compacted version could be applied in real-time classification at a faster speed but with similar accuracy.

More than purely its speed, the sensitivity an algorithm provides is highly important when aiming to discover weak sources. Thus, despite the somewhat surprising fact that a moderately deep CNN could search raw intensity data faster than the brute-force dedispersion algorithm, we argue that replacing dedispersion with deep learning will only be valuable in instances where the S/N per pixel is not too low. This is because algorithms like brute-force dedispersion, the FDMT, and tree dedispersion are either optimal or near-optimal in signal recovery. With a 2D CNN of only a dozen layers, the model is more successful if the S/N does not fall significantly below $\sim$1 pixel^{-1}. The universal approximation theorem states that a finite feed-forward neural network can approximate arbitrary functions, meaning that a sufficiently large network could, in principle, mimic optimal dedispersion (Cybenko 1989). However, the theorem says nothing about such a network being reasonably sized or about its learnability. Still, by demonstrating the low theoretical complexity of classification, the radio community can consider the problems for which real-time deep-learning classifiers might be suited, including dedispersion in some cases. In the following section, we discuss this further.

7. Discussion

In this work, we have found it sufficient to simulate FRBs based on several parameters drawn from wide distributions. However, if one wanted to improve the realism of true positives in one's training set, there are new techniques that can be employed. Generative adversarial networks (GANs) are a class of deep-learning algorithms that could generate realistic FRB candidates. They consist of two adversarial networks: one that generates realizations and another that attempts to discriminate real from simulated data (Goodfellow et al. 2014). The generator's goal is to "fool" the discriminator, eventually resulting in a high error rate in classification. This has allowed for the creation of photorealistic images based on drawings (Shrivastava et al. 2016). Guo et al. (2017) found that a standard deep CNN hit a performance ceiling for pulsar searching using real pulsars, so they used a deep convolutional GAN to build a collection of candidates. If such techniques were developed further, they might be useful for generating simulated RFI. In general, RFI is very difficult to model, but with an adversarial network trained on unlabeled real data, the problem of parameterizing it by hand could be overcome.

The black-box problem is another general concern about using DNNs in place of more explicit modeling. While we consider this a genuine issue for other problems, in the case of false-positive sifting, our hybrid artificial net is no more opaque than a human scientist's biological neural network. A human scientist knows some basic facts about dedispersed FRBs—they are roughly broadband, narrow in time, etc.—and then gets a "feel" for what false positives look like by inspecting 10^3–10^4 triggers. We never really know which features the expert has deemed salient, whereas in Figure 3, we show the actual activations inside of our neural network for a given input. Therefore, if our goal is simply to save time by accurately filtering out false positives, the black-box problem is not a major consideration. In the future, this may not suffice. As AI takes a larger role in the transient detection process, simply plotting activations like in Figure 3 will not provide enough insight into the neural network. The ML community has attempted to remedy the problem of better understanding classifiers. Using instance-specific tools such as "saliency maps" (Simonyan et al. 2013) and "prediction differential analysis" (Robnik-Šikonja & Kononenko 2008; Zintgraf et al. 2017), one can learn what a model considers to be statistical evidence for or against a given class. If transients like FRBs are to be detected in raw data by large DNNs and with almost no human in the loop, astronomers will have to become comfortable with these tools.

Having an ML classifier that can keep up with real-time triggers will be useful for a number of reasons. Even if all candidates are to be written to disk, the number of false positives may end up being prohibitively large for email notifications, outriggers, voltage dumps, or VOEvents. Because our neural network assigns a probability to each candidate, groups can set up a confidence threshold, below which triggers are saved but do not effect an alert.

In terms of the standard classification metrics, ours is the best-performing model. However, the problem with this comparison is that five of the eight models in Table 1 ran on different data sets. None of the data sets are presently available. Better would be to compare these models on the same data, in an open and collaborative platform, using the same data, truth criteria, and metrics. In a number of data-intensive disciplines, such benchmarks are emerging (Wu et al. 2018). Benchmark setup requires significant time, effort, and expertise. Together, SURF[11] and the Netherlands eScience Center[12] are developing a platform that includes software and hardware infrastructure for such research benchmarks (A. Mendrik & M. Hester 2018, in preparation). This is something the FRB community could take advantage of in the case of ML in single-pulse search.

We have also discussed the possibility of not only sifting through high-significance dedispersed candidates in real time but actually searching raw data in place of dedispersion back ends. Beyond the optimized routines we described in Section 6, training of and classification with deep neural nets is being made faster by tailored GPU hardware. Nvidia has released Tensor Cores in their Volta-based Tesla V100 that provide almost an order-of-magnitude speed-up in matrix multiplication for large arrays over the Pascal-based P100 GPU. Google has also responded to the increased use of DNNs by building a custom application specific integrated circuit (ASIC) that they have called "tensor processing units" (TPUs; Jouppi et al. 2017). These TPUs cannot yet help train neural networks but were built specifically for classification, ideal for what we have described in Section 6.

[11] https://www.surf.nl/en/innovationprojects/open-science/enlighten-your-research.html
[12] https://www.esciencecenter.nl/

We showed that the computational complexity of a single forward propagation through a modest CNN can be significantly less than that of brute-force dedispersion. Furthermore, dedispersion algorithms tend to have low arithmetic intensity, which means that they are memory-bound and not ideal for GPUs. Classification using neural networks amounts to a series of matrix multiplications accelerated by previously discussed hardware. However, we argue that a CNN could not reach the level of statistical optimality of known dedispersion algorithms without making the network so large that gains in speed were lost.

Applying deep learning to real-time transient detection may still be useful in upcoming surveys. Dedispersion algorithms search for signals with ν^{-2} sweeps caused by the differential group velocity of light in cold dense plasmas. Deviations from such a quadratic dispersion relation can come from relativistic plasmas or electrons whose plasma frequency is close to the observing frequency. Unusual polarization signatures can be induced by propagation, which can be searched for (Kennett & Melrose 1998). SETI might also find these techniques useful in searching for bright, structured signals from extraterrestrial civilizations. But history teaches us that the most exciting discoveries in transient astronomy come from "unknown unknowns," usually by searching a parameter space that was not previously accessible. The SKA and ngVLA will offer such data sets, and their availability may coincide with great advances in unsupervised or semisupervised learning.

8. Conclusions

We have applied deep learning to the problem of single-pulse classification, with large real-time FRB surveys in mind. Using Google's `TensorFlow`, we developed several independent DNNs, as well as a hybrid neural network that takes FRB candidate diagnostic data, such as dynamic spectra, the DM–time intensity array, and multibeam information, and returns a probability of the event being real. Models can be trained offline but applied in real time, allowing for low-latency classification if outriggers, VOEvents (Petroff et al. 2017), or voltage dumps are to be triggered. These tools are available on github.[13]

The possibility of replacing dedispersion back ends with a single DNN classifier was investigated. Although statistical optimality to purely quadratically dispersed signals may not be achievable without cumbersome multilayer models, we showed that forward propagation could be done more efficiently and quickly than brute-force dedispersion on modern hardware. Thus, deep-learning classification of signals more diverse than dispersion is feasible on raw data in real time.

We thank Emily Petroff for helpful comments on the manuscript, as well as our anonymous referee for valuable feedback. We thank the CHIME Collaboration for allowing the usage of data from its Pathfinder instrument and the Apertif Survey Team for the use of the ARTS false-positive data set. We also thank Jorn Peters, Yunfan (Gerry) Zhang, and Folkert Huizinga for useful discussions. The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007–2013)/ERC Grant Agreement no. 617199 and from the Netherlands Research School for Astronomy (NOVA4-ARTS).

[13] https://github.com/liamconnor/single_pulse_ml

ORCID iDs

Liam Connor https://orcid.org/0000-0002-7587-6352

References

Amiri, M., Bandura, K., Berger, P., et al. 2017, ApJ, 844, 161
Aulbert, C. 2007, arXiv:astro-ph/0701097
Bannister, K. W., Shannon, R. M., Macquart, J.-P., et al. 2017, ApJL, 841, L12
Barsdell, B. R., Bailes, M., Barnes, D. G., & Fluke, C. J. 2012, MNRAS, 422, 379
Breiman, L. 2001, Machine Learning, 45, 5
Caleb, M., Flynn, C., Bailes, M., et al. 2017, MNRAS, 468, 3746
Chetlur, S., Woolley, C., Vandermersch, P., et al. 2014, arXiv:1410.0759
Coenen, T., van Leeuwen, J., Hessels, J. W. T., et al. 2014, A&A, 570, A60
Connor, L. 2018, liamconnor/single_pulse_ml: First release of single pulse machine learning code, Zenodo, doi:10.5281/zenodo.1442657
Cybenko, G. 1989, Mathematics of Control, Signals and Systems, 2, 303
Devine, T. R., Goseva-Popstojanova, K., & McLaughlin, M. 2016, MNRAS, 459, 1519
Foster, G., Karastergiou, A., Golpayegani, G., et al. 2018, MNRAS, 474, 3847
Goldberg, Y. 2015, arXiv:1510.00726
Goodfellow, I., Bengio, Y., & Courville, A. 2016, Deep Learning (Cambridge, MA: MIT Press)
Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., et al. 2014, arXiv:1406.2661
Guo, P., Duan, F., Wang, P., Yao, Y., & Xin, X. 2017, arXiv:1711.10339
Jouppi, N. P., Young, C., Patil, N., et al. 2017, arXiv:1704.04760
Kaspi, V. M. & CHIME/FRB Collaboration 2017, AAS Meeting, 229, 242.19
Kennett, M., & Melrose, D. 1998, PASA, 15, 211
Lecun, Y., Bengio, Y., & Hinton, G. 2015, Natur, 521, 436
Lin, H. W., Tegmark, M., & Rolnick, D. 2017, JSP, 168, 1223
Lorimer, D. R., Bailes, M., McLaughlin, M. A., Narkevic, D. J., & Crawford, F. 2007, Sci, 318, 777
Lyon, R. J., Stappers, B. W., Cooper, S., Brooke, J. M., & Knowles, J. D. 2016, MNRAS, 459 arXiv:1603.05166
Maan, Y., & van Leeuwen, J. 2017, arXiv:1709.06104
Maas, A. L., Hannun, A. Y., & Ng, A. Y. 2013, in Proc. ICML 30
Magro, A., Karastergiou, A., Salvini, S., et al. 2011, MNRAS, 417, 2642
Marcote, B., Paragi, Z., Hessels, J. W. T., et al. 2017, ApJL, 834, L8
Masui, K., Lin, H.-H., Sievers, J., et al. 2015, Natur, 528, 523
Michilli, D., Hessels, J. W. T., Lyon, R. J., et al. 2018b, MNRAS, 480, 3457
Michilli, D., Seymour, A., Hessels, J. W. T., et al. 2018a, Natur, 553, 182
Ng, C., Vanderlinde, K., Paradise, A., et al. 2017, arXiv:1702.04728
Oppermann, N., Connor, L. D., & Pen, U.-L. 2016, MNRAS, 461, 984
Petroff, E., Bailes, M., Barr, E. D., et al. 2015a, MNRAS, 447, 246
Petroff, E., Barr, E. D., Jameson, A., et al. 2016, PASA, 33, e045
Petroff, E., Houben, L., Bannister, K., et al. 2017, arXiv:1710.08155
Petroff, E., Keane, E. F., Barr, E. D., et al. 2015b, MNRAS, 451, 3933
Ravi, V. 2017, arXiv:1710.08026
Ravi, V., Shannon, R. M., Bailes, M., et al. 2016, Sci, 354, 1249
Robnik-Šikonja, M., & Kononenko, I. 2008, IEEE Transactions on Knowledge and Data Engineering, 20, 589
Rosen, R., Swiggum, J., McLaughlin, M. A., et al. 2013, ApJ, 768, 85
Scholz, P., Spitler, L. G., Hessels, J. W. T., et al. 2016, ApJ, 833, 177
Sclocco, A., van Leeuwen, J., Bal, H. E., & van Nieuwpoort, R. V. 2016, A&C, 14, 1
Sclocco, A., Van Nieuwpoort, R., & Bal, H. E. 2014, in Exascale Radio Astronomy, AAS Topical Conference Series Vol. 2, 203.01
Shrivastava, A., Pfister, T., Tuzel, O., et al. 2016, arXiv:1612.07828
Simonyan, K., Vedaldi, A., & Zisserman, A. 2013, arXiv:1312.6034
Smits, R., Kramer, M., Stappers, B., et al. 2009, A&A, 493, 1161
Spitler, L. G., Cordes, J. M., Hessels, J. W. T., et al. 2014, ApJ, 790, 101
Spitler, L. G., Scholz, P., Hessels, J. W. T., et al. 2016, Natur, 531, 202
Taylor, J. H. 1974, A&AS, 15, 367
Tendulkar, S. P., Bassa, C. G., Cordes, J. M., et al. 2017, ApJL, 834, L7
Thornton, D., Stappers, B., Bailes, M., et al. 2013, Sci, 341, 53
van Leeuwen, J. 2014, in The Third Hot-wiring the Transient Universe Workshop (HTU-III), ed. P. R. Wozniak, 79
van Leeuwen, J., & Stappers, B. W. 2010, A&A, 509, 7
Wagstaff, K. L., Tang, B., Thompson, D. R., et al. 2016, PASP, 128, 084503
Wu, Z., Ramsundar, B., Feinberg, E. N., et al. 2018, Chem. Sci., 9, 513
Zackay, B., & Ofek, E. O. 2017, ApJ, 835, 11
Zhu, W. W., Berndsen, A., Madsen, E. C., et al. 2014, ApJ, 781, 117
Zintgraf, L. M., Cohen, T. S., Adel, T., & Welling, M. 2017, arXiv:1702.04595