Machine Learning Applications for Fast Radio Bursts

Arhum Siddiqui, Fereshteh Rajabi

April 22, 2024

1 Introduction

Fast Radio Bursts (FRBs) are millisecond duration bursts of intense radio waves, ranging from 50 mJy to 100 mJy, whose origins are unknown (Petroff et al., 2019). FRB emission has so far been detected between 400 MHz and 8 GHz (Petroff et al., 2019). The detection of FRBs in 2007 by Lorimer et al. (2007) provides the astronomical community an opportunity to provide insights into new energetic processes. As such, they are an intriguing new mystery of astrophysics. While most FRBs are observed to be a one-time event, a small fraction of FRBs such as FRB 20121102 (Spitler et al., 2016), show repeated bursts with a range of bursting rates. Thus, FRBs can generally be classified into repeating and non-repeating FRBs.

The typical detection of FRBs works on offline data analysis. They are typically performed on high-time resolution data by first accounting for dispersion measure (DM) over many trials. A time series is then generated for the de-dispersed data. The time series represents the radio signal as a function of time. Kernals of various widths are convolved with the time series to look for broad radio pulses. The chunks of data that show potential FRBs are then flagged for human inspection. This process implies that there is room for improvement in detection rates and accuracy (Thakur, 2020).

1.1 Motivation for Review

Conventional offline data analysis restricts the amount of data that can be stored, limiting the sensibility of telescopes. As such, many faint FRBs go unnoticed. A new generation of radio telescopes, however, is under development such as the Square Kilometre Array (SKA), slated for completion in 2027. Detecting FRBs is a primary objective of the SKA. It is expected that the SKA will generate data at a rate exceeding 1 terabyte per second (Czech et al., 2018). This immense data rate makes conventional offline analysis techniques impractical. Hence, there's a need to develop new methods for real-time FRB detection.

Recently, advancements in dedispersion algorithms and the use of modern Graphics Processing Units (GPUs) have made real-time FRB searches possible. However, these searches are challenged by a high rate of false positives, often caused by terrestrial interference, such as TV broadcasts, WiFi signals, and satellite communications (Thakur, 2020). Dealing with this high false positive rate necessitates human data inspection. Various algorithms have been employed to detect Radio Frequency Interference (RFI) to reduce the volume of data that needs human analysis. These include clustering techniques to identify common types of RFI, such as K-Nearest Neighbors (KNN) and Random Forests (Foster et al., 2018; Luo et al., 2022; Wagstaff et al., 2016; Zhu-Ge et al., 2022) and more advanced methods like

Convolutional Neural Networks (CNN) which have effectively distinguished genuine FRB pulses from RFI (Agarwal et al., 2019; Connor & van Leeuwen, 2018; Zhang et al., 2018).

Thus, the objective of this paper is to do a literature review on the different machine learning (ML) models for detecting FRBs.

2 FRB Background

Currently, the catalogue for known FRB sources is limited and the dataset is not large enough to train an ML algorithm. Thus it is necessary to simulate FRBs with the correct properties and then train ML models with this dataset. In this section, we will review the necessary background to simulate an FRB.

2.1 Properties of FRBs

Although there are no strict guidelines in place for defining FRBs as they are relatively new, any radio pulse that meets the criteria is considered to be an FRB. These criteria include the pulse's **dispersion measure**, **pulse time duration**, **brightness**, **broadband characteristics**, etc. This subsection outlines various propagation effects that are relevant to FRB observations. A much more detailed and fundamental description of propagation effects in radio astronomy, in general, can be found in the reviews such as Rickett (1977), and Rickett (1990). A thorough review on FRBs can be found in Petroff et al. (2019).

2.1.1 Dispersion Measure (DM)

Dispersion refers to the phenomenon where different frequencies of FRBs (and light signals in general) travel at varying speeds when passing through an ionized medium, primarily due to the presence of free electrons. The dispersion measure is directly proportional to the column density of these free electrons, which can be found in the interstellar medium, intergalactic medium, or any ionized medium, including the ionosphere of Earth (Petroff et al., 2019). Dispersion is one of the main properties that one looks for when classifying FRBs.

The amount of dispersion is calculated as the measure of the time delay between the highest and lowest frequencies

The time delay Δt is given as:

$$\Delta t = \frac{e^2}{2\pi m_e c}(\nu_{\text{lo}}^{-2} - \nu_{\text{hi}}^{-2})\text{DM} \approx 4.15(\nu_{\text{lo}}^{-2} - \nu_{\text{hi}}^{-2})\ \text{DM (ms)} \tag{1}$$

where Δt is in milliseconds seconds (ms), m_e is the classical mass of the electron and c is the speed of light in a vacuum (Petroff et al., 2019).

The dispersion measure is given as

$$DM = \int_0^d n_e(l)\, dl \tag{2}$$

n_e denotes the electron number density, l is the path length, and d is the distance to the FRB (Petroff et al., 2019).

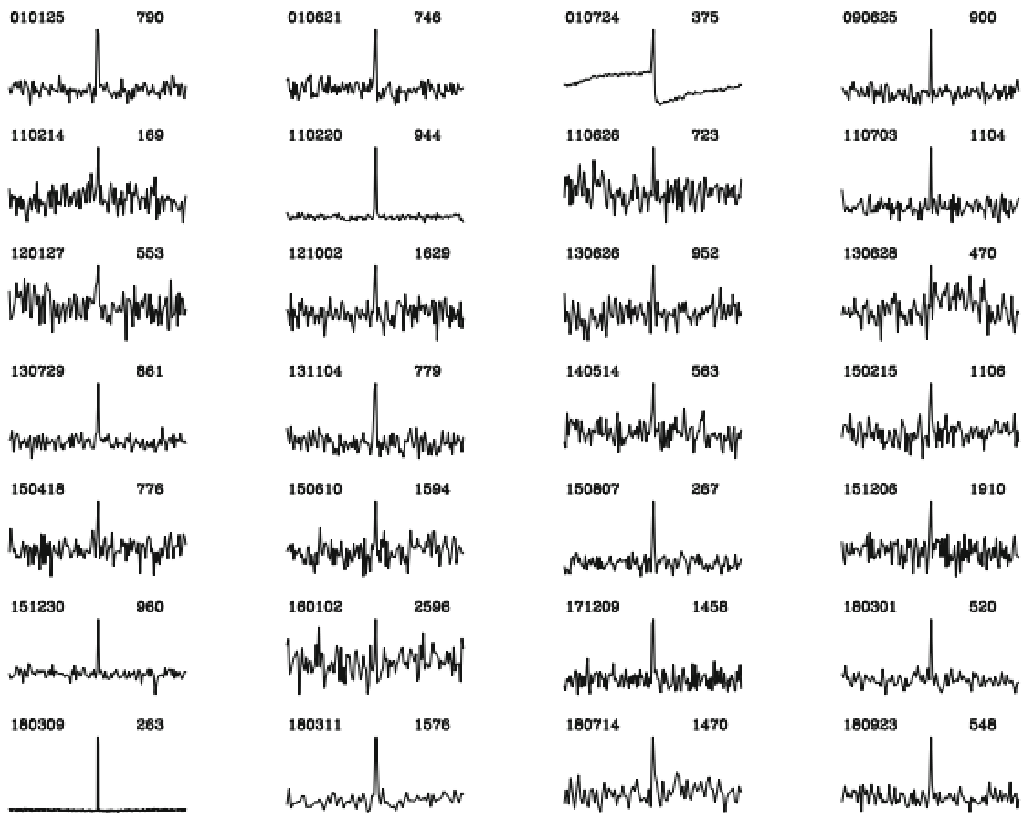


Figure 1: This is a compilation showcasing the initial twenty-eight FRBs detected through observations with the Parkes telescope. The detections are organized chronologically by date of discovery. Each light curve depicts a 2-second interval surrounding the pulse. Following the convention of gamma-ray burst nomenclature, the FRBs are denoted in YYMMDD format, representing the year (YY), month (MM), and day (DD) of detection. Additionally, the observed dispersion measures (DMs) are provided to the right of each pulse, measured in units of cm^{-3} pc (Petroff et al., 2019).

2.1.2 Pulse Width of an FRB

The pulse width W of an FRB is typically considered to be a combination of different contributions, both intrinsic and instrumental. We are generally interested in the top hat pulse. The top hat pulse, also known as the rectangular pulse, is a type of pulse signal in signal processing and telecommunications. It is characterized by having a constant amplitude over a specific duration and zero amplitude outside that duration. The equation of a top hat pulse for an FRB is

$$W = \sqrt{W_{\text{int}}^2 + t_{\text{samp}}^2 + \Delta t_{\text{DM}}^2 + \Delta t_{\text{DMerr}}^2 + \tau_s^2}, \tag{3}$$

where W_{int} represents the inherent pulse duration of the FRB. t_{samp} is the sampling time which is a characteristic of how frequently data points are recorded in the observation. Δt_{DM} accounts for the dispersive delay across individual frequency channels. Δt_{DMerr} represents the dispersive delay due to dedispersion at a slightly incorrect DM. τ_{s} refers to the scattering time scale. This accounts for the pulse broadening caused by the FRB travelling through any turbulent medium such as the interstellar medium. It is important to note that t_{samp}, Δt_{DM}, Δt_{DMerr}, and τ_{s}, are all instrumental spreading contributions (Petroff et al., 2019).

2.1.3 Scattering

Scattering is caused by inhomogeneities and irregularities in the interstellar medium or any other medium that the signal encounters as it propagates. Scattering can alter and deflect the path of the signal, causing it to traverse longer paths. Both scattering and dispersion lead to the temporal broadening of a signal because they cause different components of the signal to arrive earlier or later. However, they are different. In Petroff et al. (2019), the authors discuss the relationship between the decay time τ and frequency ν as the following:

$$\tau \propto \nu^{-4} \tag{4}$$

In Connor & van Leeuwen (2018), the authors simulate FRBs to real background data that have already been dedispersed to a random DM. They calculate the pulse profile at each frequency by convolving a Gaussian with a scattering profile,

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

Where τ_ν is the scattering timescale at frequency ν and is given by,

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

with ν_{ref} acting as a reference frequency.

2.1.4 Scintillation

Due to FRBs inferred small emitting regions and large distances (Gajjar et al., 2018; Tendulkar et al., 2017), FRBs are expected to behave as ideal point sources, suggesting the likelihood of scintillation unless there exists considerable angular broadening of the source (Petroff et al., 2019). Scintillation is the rapid fluctuation in the intensity or amplitude of a signal, whether in radio waves or light, e.g., the twinkling of stars. Scintillation is more pronounced when the source is compact and the signal is coherent, as is the case with FRBs. When a signal is coherent, any interruptions, deflections, and the resulting interferences will be more pronounced. Scintillation has two main characteristics. **Scintillation bandwidth** which is the range of radio frequencies affected by scintillation and **scintillation time** which is how long a scintillation pattern lasts. The scintillation is approximated by the positive half of the cosine function with a random phase:

$$\mathrm{A} = \cos\left(2\pi \mathrm{N}_{\mathrm{scint}} \left(\frac{\nu}{\nu_{\mathrm{ref}}}\right)^{-2} + \phi_{\mathrm{scint}}\right) \tag{7}$$

where A is the amplitude of the pulse, N_{scint} is the number of scintillations (uniformly distributed from 0 to 10), and ϕ_{scint} is some random phase (uniformly distributed from 0 to 1) (Thakur, 2020).

3 Classical Machine Learning Background

Classical ML encompasses traditional techniques and algorithms that have been developed and used before the appearance of deep learning algorithms. These methods typically include supervised and unsupervised ML techniques. Supervised ML involves training models with known outputs whereas unsupervised ML involves training models on unlabelled data to uncover hidden patterns or structures within the data.

3.1 Supervised Machine Learning

This subsection reviews the main supervised ML models prevalent in the literature for classical ML. Supervised ML is a type of ML where the algorithm learns from labelled data, which means that each input data point is associated with a corresponding target label or output. The goal of supervised learning is to learn a mapping function from input variables to output variables.

3.1.1 Simple Decision Tree Learning

Classification and Regression Trees (CART) is a type of simple machine learning algorithm where decision tree learning provides a flowchart-like structure for predicting target values based on input features (Shalev-Shwartz & Ben-David, 2014).

3.1.2 Random forest Algorithm

A crucial topic in machine learning is the bias-variance tradeoff as highlighted by various papers (e.g., Geman et al. (1992); Friedman et al. (1997); Neal (2019). Bias refers to the difference between the predictions for the models to the training input, while variance measures how accurately the model can predict with new data that is not present in the training data. Achieving both low bias and low variance simultaneously is challenging in reality, necessitating a delicate balance between the two. Random Forest, an ensemble method, combines multiple decision trees to enhance accuracy and reduce overfitting (Breiman, 2001). One can simply select random sub-samples of the training set, and train many deep decision trees with the random sub-samples that tend to overfit. This process is called **bagging** (Breiman, 1996).

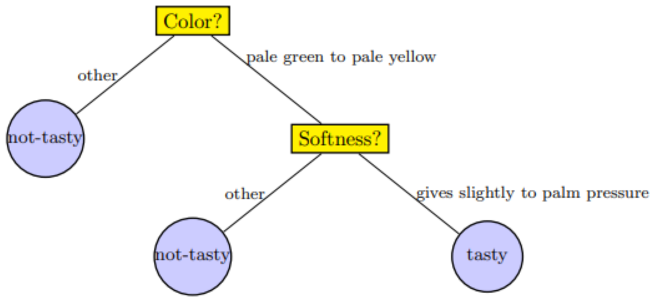


Figure 2: Illustration of a basic decision tree model for determining the tastiness of papayas Shalev-Shwartz & Ben-David (2014).

3.1.3 Boosted Decision Tree Method

A boosted decision tree involves training a series of decision trees, where each tree corrects the errors of the previous one. Common boosting algorithms include AdaBoost, LightGBM, and XGBoost. These algorithms assign weights to the training instances based on their performance, which gives more emphasis to the misclassified occurrences.

3.1.4 Support Vector Machines (SVM)

SVMs are useful for classification and regression tasks. They operate by finding a hyperplane that best separates data points in different classes based on input labels (Cortes & Vapnik, 1995). However, there are many cases where the data is not easily separable by a linear boundary. Thus, when there is no single straight line that can separate the data points into their respective classes, this would then be referred to as non-linearly separable data. To handle such a case, the *kernel trick* can be used (Hofmann et al., 2008). The kernel trick is a technique used in SVMs to implicitly map the input data into a higher-dimensional space where the data becomes linearly separable. Then, SVM can find a hyperplane that separates the data points linearly even if the original data itself is not linear in its original feature space.

3.1.5 Nearest Centroid Method

The nearest centroid method classifies data points based on the similarity to the centroid of each class. The centroid is the average of all the feature vectors belonging to that class (Hastie et al., 2009).

3.1.6 K-Nearest Neighbours Classifier (KNN)

KNN is based on the idea that points in the same class tend to be close to each other in the feature space. Instead of centroids, KNN considers the k-nearest neighbours of a data point (Shalev-Shwartz & Ben-David, 2014).

3.2 Unsupervised Machine Learning Background

While supervised ML techniques require target labels or classified labels as input data, unsupervised ML does not require such a premise. Thus, unsupervised ML techniques usually reveal hidden information from the input data. Unsupervised ML is usually done in two steps. The initial step focuses on dimensionality reduction, where raw data is transformed and visualized from a high-dimensional space to a lower-dimensional space for easier comprehension and evaluation. The second step entails clustering, which is a distinct algorithm that groups clusters or clumps of data points (Zhu-Ge et al., 2022).

Dimensionality Reduction Methods

3.2.1 Principal Component Analysis (PCA)

PCA is one of the most common dimensionality reduction techniques used in ML. PCA reduces the dimensionality of the data by transforming it into a new coordinate system all while maintaining the most important features. The process can be summarized as the following, it starts with standardizing the data and then computing the covariance matrix. The eigenvalues and corresponding eigenvectors of this matrix are then calculated, with the eigenvectors indicating directions of maximum variance and the eigenvalues signifying the magnitude of variance in those directions. By selecting the top eigenvectors based on their eigenvalues, known as principal components, PCA enables the transformation of the original data into a lower-dimensional space. This projection onto the subspace of principal components results in uncorrelated variables, allowing for efficient dimensionality reduction while retaining a significant portion of the original data's variability. A more detailed explanation can be found in (Abdi & Williams, 2010).

3.2.2 t-distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE is a dimensionality reduction technique that is commonly used in ML and data visualization. It is primarily used for visualizing high-dimensional data in lower dimensions while preserving the local structure of the data. The algorithm constructs a probability distribution over pairs of high-dimensional data points. It then aims to find a low-dimensional representation of the data where the Kullback-Leibler (KL) divergence between the original distribution and the distribution of the lower-dimensional points is minimized. One of the key features of this technique is that it emphasizes the relationships between nearby data points while downplaying points that are further apart. This property makes it useful for visualizing complex datasets and identifying their clusters or patterns. Refer to Van der Maaten & Hinton (2008) for a more thorough understanding of this algorithm.

3.2.3 Uniform Manifold Approximation and Projection (UMAP)

UMAP is a dimensionality reduction technique known for preserving local and global data structures. UMAP is constructed from a theoretical framework based on Riemannian geometry and algebraic topology. One key advantage of UMAP compared to other dimensionality reduction techniques like t-SNE is its computational efficiency, particularly for large datasets, while still preserving the global structure of the data. This makes UMAP well-suited for exploratory data analysis, clustering, and visualization tasks. A more in-depth understanding of this algorithm can be viewed in McInnes et al. (2018)

3.3 Clustering Methods

3.3.1 k-means

k-means is a popular clustering algorithm that partitions data points into k clusters based on similarity. The algorithm can be summarized in several key steps. One starts by initializing k initial cluster centroids. Each data point is assigned to its nearest cluster based on some distance metric, typically Euclidean distance. After all data points have been assigned, the centroids of the clusters are updated by computing the mean of all points assigned to each cluster. Assigning and updating data points are iteratively done until convergence is met, which is typically achieved when the centroids no longer change significantly or after a predetermined number of iterations. For a more detailed explanation, one can refer to the following review Kodinariya et al. (2013)

3.3.2 HDBSCAN

HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) is an extension of DBSCAN (Density-Based Spatial Clustering of Applications with Noise). It builds upon the DBSCAN by introducing a hierarchical approach to density-based clustering. It is an algorithm that is used to determine the number of clusters automatically. It is primarily useful when the dataset has varying densities and irregular shapes (i.e. when the concentration of data varies and when the clusters or groupings of data points do not conform to simple geometric shapes). By leveraging the density of points and their proximity to each other, HDBSCAN identifies dense regions as clusters while categorizing outliers as noise and therefore provides a comprehensive understanding of the underlying structure of the data. As such it is used in various fields such as data mining, pattern recognition, and anomaly detection, where uncovering meaningful clusters amidst noisy or high-dimensional data is essential (Campello et al., 2013).

4 Deep Learning Background

This section covers the essential background of deep learning which is covered extensively in Goodfellow et al. (2016). Neural networks are a type of ML algorithm inspired by the biological neuron and their network. A typical neural network consists of many different layers. Each layer is made of many neurons connected to the previous layer's output. The term *deep* comes from the depth of the neural networks, meaning they have multiple layers between the input and output layers. (Goodfellow et al., 2016). Refer to Figure 2 for a schematic of a simple neural network architecture.

Each neuron within a neural network calculates a weighted sum of its inputs (represented by $\mathbf{x}$) and produces an output ($\mathbf{y}$) using the formula

$$\mathbf{y} = f(\mathbf{w} \cdot \mathbf{x} + b) \tag{8}$$

where $\mathbf{w}$ denotes the weights, f is a non-linear activation function determining the output of the neuron, and b is the bias term. During the training process, both the weights and bias are adjusted to optimize performance. There are many different types of activation functions. One example is the Rectified Linear Unit (ReLU) which is commonly used. ReLu is defined as

$$f(x) = \max(0, x) \tag{9}$$

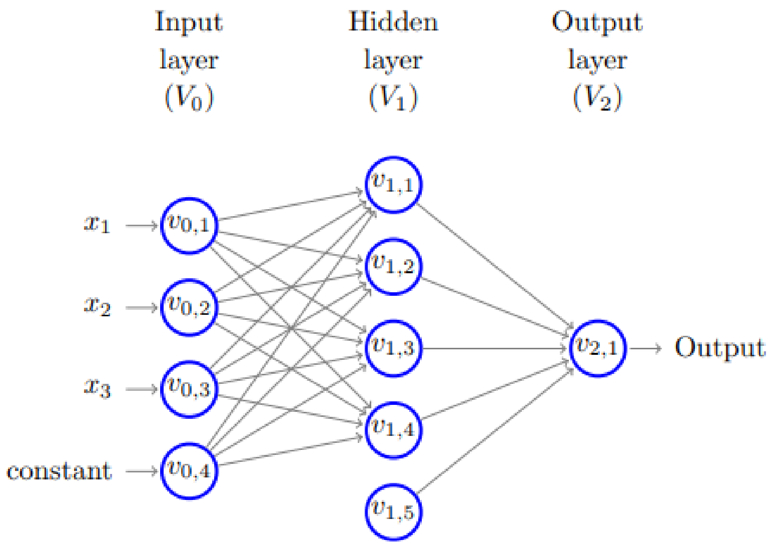


Figure 3: Illustration of a simple neural network (Shalev-Shwartz & Ben-David, 2014).

The arrangement of these neurons into layers defines the neural network's architecture. For instance, a dense layer connects all inputs to all outputs. By iteratively adjusting weights and biases across multiple layers, neural networks can effectively map input data to desired outputs using complex non-linear transformations.

4.1 Convolutional Neural Networks (CNN)

CNNs are a type of artificial neural network that is mostly used for working with images. They use the convolution operator instead of matrix multiplication. The core building blocks of CNNs are convolutional layers. These layers use the convolution operator to scan an image with small filters (kernels) to detect local patterns or features like edges, corners, and textures. This is then followed by an *activation function* (Goodfellow et al., 2016).

4.1.1 Convolution Operation

The convolution operator when performed on two functions produces a third function which describes how they both interact. Specifically, how the second function changes the shape of the first function. It is given by the formula:

$$s(t) = (x * w)(t) = \int x(a)w(t-a)da \tag{10}$$

Where $x(a)$ is the input function and $w(a)$ is the kernel (Goodfellow et al., 2016). When computing, however, we take the discrete case which is given by:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{11}$$

When using a two-dimensional input, the kernel K must also be two-dimensional. Thus the equation is now given by:

$$s(t) = (I * K) = \sum_{m} \sum_{n} I(m, n)K(t - m, t - n) \tag{12}$$

Where m and n are the dimensions of the input (Goodfellow et al., 2016).

4.1.2 Pooling

After convolution, the extracted features undergo a pooling process. This involves pooling layers that are used to reduce the spatial dimensions of the feature maps produced by the convolutional layers. Pooling involves down-sampling the feature maps by taking the maximum or average value in a local region. This helps make the network more computationally efficient and invariant to small translations (i.e. small shifts or displacements of objects or features within an image)in the input (Goodfellow et al., 2016).

4.1.3 Overview of CNN architecture

The combination of convolution and activation layers works together to extract features from the input data. Following this, the extracted features undergo pooling. This sequence of convolution, activation, and pooling layers is repeated multiple times. Eventually, the processed images are reshaped into one-dimensional arrays and connected to a dense layer. In this layer, the extracted features are assigned probabilities indicating their likelihood of belonging to different output classes. A simplified CNN architecture typically comprises of multiple convolutional layers followed by dense layers. The final outcome of the network is usually a probability distribution over the possible classes for the input data. Various hyperparameters, including the total number of layers, the number of filters in the convolutional layers, the units in the dense layer, and the choice of activation function determine the specific structure of the network. The selection of these hyperparameters depends on the particular application at hand.

4.2 Training the Neural Network

For CNNs to make accurate predictions, they are initially trained using labelled data. The filters and dense layers of the network are initialized with random weights. The labelled data is then forwarded through the network, producing classification probabilities in a process known as **forward propagation**. The difference between the predicted values of an ML model and the actual values is quantified using a cost function, which when assessed over the labelled dataset is termed as the loss. To train the network, the loss is minimized using optimization algorithms, which compute the gradients of the cost function with respect to the weights. These gradients, combined with a learning rate, are used to adjust the weights in a process called **backward propagation** (Goodfellow et al., 2016).

The labelled data is typically divided into three sets: training, validation, and test data. The network is trained on the training data, and its performance is evaluated based on the validation set. This iterative process continues until satisfactory performance on the validation data is achieved. Once trained, the network's performance is assessed using the test data. Typically, these datasets consist of thousands of examples, and it's important to note that the validation and test datasets are never used for training the network (Goodfellow et al., 2016).

4.2.1 Deeper layers

A deeper neural network has more layers in it. Thus, it can learn features at various levels of abstractions (Mhaskar et al., 2016). Using multiple layers allows for the network to understand the data better, from basic details to more complex patterns. For a fixed amount of parameters, going deeper allows for the models to capture richer features (Eldan & Shamir, 2016). However, there is a trade-off to consider. Deeper models with a higher number of trainable parameters may encounter overfitting issues, particularly when dealing with limited input datasets (Eldan & Shamir, 2016).

4.2.2 Residual Networks

Residual networks, introduced by He et al. (2016a), are a type of deep CNN. They are designed to facilitate the training of very deep networks by introducing skip connections or shortcuts that bypass one or more layers. This is achieved by adding the original input of a block of layers to the output of the block. These skip connections enable the network to learn residual mappings, i.e., the difference between the input and the desired output, rather than directly learning the mapping. This approach has been shown to make it easier to optimize and train very deep networks. Thus they are best used for very deep networks. The nature of residual networks also allows them to be well suited for tasks that require hierarchical feature learning, such as image classification, object detection, and segmentation. For a more thorough review and understanding of residual networks, refer to He et al. (2016a,b).

4.2.3 Transfer Learning

Transfer learning involves the process of using a pre-trained model that has already been trained on a large dataset for a specific task, such as image recognition. One then takes this pre-trained model and fine-tunes it for a smaller dataset for another task. By doing this process, the model can leverage the features and patterns it learned from the original task to expedite learning and improve performance on the new task. Transfer learning is most applicable when there is a limited supply of target training data, which is the case for FRBs and can help the model to reduce overfitting (Agarwal et al., 2019; Hosna et al., 2022).

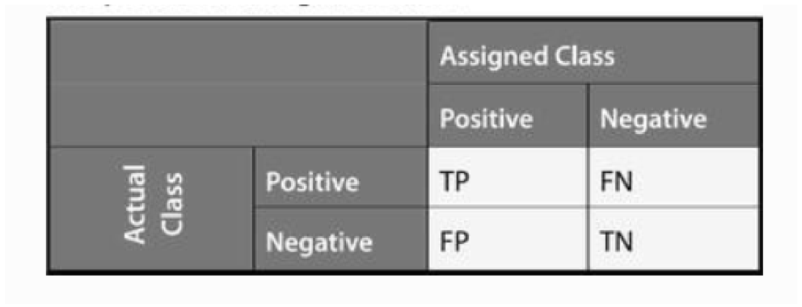
Transfer learning has been used successfully in different fields of astronomy. For instance, identification of supernovae Ia (Vilalta, 2018), detecting galaxy mergers (Ackermann et al., 2018), and galaxy classification schemes (Tang et al., 2019; Pérez-Carrasco et al., 2019; Khan et al., 2019)

5 Comparison Parameters Background

Evaluation metrics, including confusion matrix, accuracy, recall, precision, and score, provide insights into the models performance.

Confusion Matrix The confusion matrix illustrates the relationship between predicted and true classes, including True Positives, False Positives, False Negatives, and True Negatives (Ting, 2010). Refer to Figure 4 for a diagram.

Accuracy Accuracy measures the ratio of correct predictions to the total number of predictions, providing an overall assessment of model correctness.



		Assigned Class	
		Positive	Negative
Actual Class	Positive	TP	FN
	Negative	FP	TN

Figure 4: This figure illustrates a confusion matrix (Ting, 2010)

Recall Recall measures the ability of a classifier to capture all relevant instances of a positive class, considering True Positives and False Negatives.

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{13}$$

Precision Precision assesses the accuracy of positive predictions made by the classifier, considering True Positives and False Positives.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{14}$$

F$_\beta$ Score The F$_\beta$ score is calculated using the formula:

$$F_\beta = \frac{(1 + \beta^2) \times \text{Precision} \times \text{Recall}}{(\beta^2 \times \text{Precision}) + \text{Recall}} \tag{15}$$

Where β is a weighting factor for precision and recall, the F$_\beta$ score has a range of 0 to 1, with $F_\beta = 1$ signalling a perfect model (Sasaki, 2007; Rijsbergen, 1979).

Recall quantifies the proportion of actual positive instances (true positive) that were correctly classified, while precision quantifies the proportion of predicted positive instances that were positive. One can think of precision as the accuracy of positive predictions made by the classifier. The F$_\beta$ score is a metric that combines both precision and recall into a single value.

6 Classifying FRB against RFI

This section reviews different machine learning models in the literature that explore FRB classification against radio frequency interference (RFI). RFI poses a challenge for the model as background radio waves can disrupt the classification of FRBs.

6.1 Application of Supervised ML classification models

Wagstaff et al. (2016) and Foster et al. (2018) have applied a supervised random forest model by extracting data-specific features to classify the candidates into certain pre-defined classes of RFI and FRBs.

In Wagstaff et al. (2016), they made improvements to the V-FASTR fast transient system that was designed to detect rare FRBs within data collected by the Very Long Baseline Array. The resulting event candidates require a large human reviewing time. To tackle this issue, Wagstaff et al. (2016) deploys a random forest model that classifies a pulse from a known pulsar, an artifact resulting from RFI, or a potential discovery. The classifier maintains high reliability by restricting its predictions to those with at least 90% confidence and overall, reduces time spent reviewing candidates and the fraction of interesting candidates. The classifier identifies 80%–90% of the candidates, with an accuracy greater than 98%, leaving only 10%–20% most promising candidates to be reviewed by humans. 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. The model evaluation metrics for both of these papers are summarized in Table 1.

6.2 Application of deep learning for classification

Connor & van Leeuwen (2018) implements deep learning for single pulse classification of FRB candidates. They implement a multi-input CNN model that includes frequency-time data, pulse profiles, DM-time data, and multibeam detections. While some of these data representations can be visualized as 2D images (e.g., frequency-time and DM-time data), others are represented as 1D arrays (e.g., pulse profiles and multibeam detections). The training data primarily consists of simulated FRB data injected into real FRB data. Most of the true positives are simulated but they use only false positives that have already been generated in real surveys and labelled by eye. Two data sets were constructed; one is from using CHIME pathfinder data and the other is based on Apertif data. They also use random forest and SVM models to compare how their CNN model works. Refer to Table 1 for the models' performances. The potential for real-time automated classification is highlighted when using graphical processing units (GPUs). While the text explores the possibility of replacing de-dispersion back ends with a real-time deep neural network classifier, Connor & van Leeuwen (2018) concludes that the latter is sub-optimal due to low signal-to-noise per pixel however can be improved with the advancements and developments of future telescopes and arrays.

Agarwal et al. (2019) utilizes **transfer learning** and CNNs to detect FRBs against RFI. As discussed in previous sections, transfer learning is when a pre-trained model captures essential features and fine-tunes them for a smaller dataset for another task. The classification takes place in the final dense layer. A custom dense layer replaces this layer with the number of units equal to classes of data. The convolutional layers remain frozen such that their weights are not changed during training. The new classification layer can now be trained for the new dataset. This happens because the final dense layer is the only layer that needs to be trained. Therefore, the number of trainable parameters is reduced significantly. Refer to Figure 5 for a sample network architecture used in Agarwal et al. (2019).

Agarwal et al. (2019) use radio frequency-time and dispersion measure-time images as their input data for their CNN models. This study uses simulated FRBs and real RFI candidates from telescopes at the Green Bank Observatory. Figure 6 displays the sample images from the training and test dataset. The authors present 11 models, each having an accuracy and recall above 99.5%. Their test dataset comprises real FRB data and RFI and pulsar candidates. They used FRB data from ASKAP (Shannon et al., 2018), Parkes (5 from (Champion et al., 2016), FRB110220 (Thornton et al., 2013), FRB 150215 (Petroff et

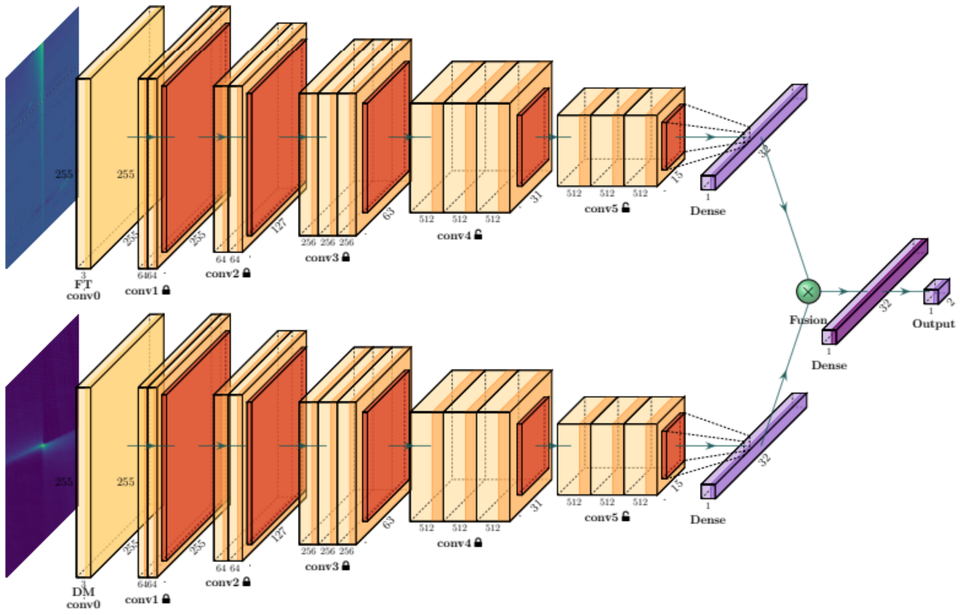


Figure 5: This figure illustrates a sample network architecture. The yellow boxes show the convolutional outputs and are labelled with output sizes. The brown edges represent the ReLU activation. The orange boxes depict the pooling layer. The dense layers are displayed in violet. The green ball represents the element-wise product of the two dense layers. The second last dense layer has a softmax activation function demonstrated by the darker-coloured edge. The lock symbol represents the frozen layers while the unlock symbol shows the unfrozen (i.e. trainable) layers. The arrows show the network connections. This figure was taken from (Agarwal et al., 2019)

.

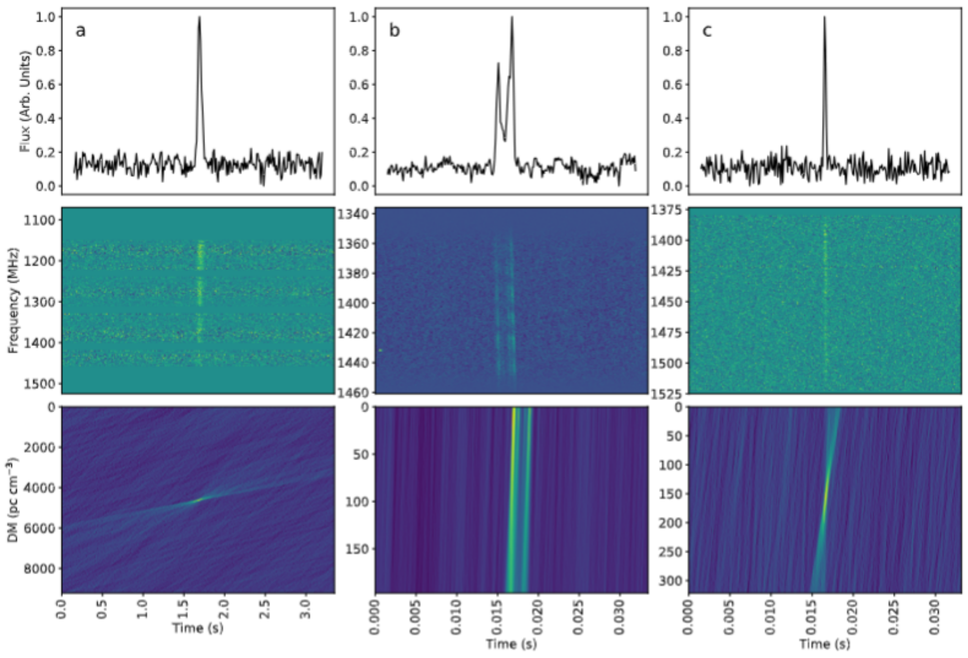


Figure 6: Sample images from the training and test dataset. The top row shows the time-series profile, which is not included in our algorithms but is included for visual reference here. The middle row is the frequency-time image, while the bottom row is the DM-time image. Column (a) corresponds to a simulated FRB with background data from FLAG. The gaps in the frequency-time plots are due to instrumental effects. Column (b) is a real RFI candidate from the 20m telescope at the Green Bank Observatory. Column (c) is a pulsar observed using the FLAG system (Agarwal et al., 2019)

.

al., 2017), and FRB 140514 Petroff et al. (2015)) and FRB121102 data from Breakthrough Listen (Gajjar et al., 2018; Zhang et al., 2018). This paper demonstrates that with the use of transfer learning, the neural network's performance is not significantly affected by variations in observing frequency or differences between telescopes/data acquisition devices.

6.3 Analysis

Of the classical ML models, the most prevalent model used in the literature for classifying FRBs and reducing false positive rates is the random-forest algorithm (Connor & van Leeuwen, 2018; Wagstaff et al., 2016). These papers showcase the effectiveness of feature extraction and classification in identifying FRBs and RFI. The approach significantly reduces the time spent on human review by accurately classifying candidates with high reliability. However, newer papers are now exploring deep learning models, and more specifically CNNs. Table 1 presents a summary of the evaluation metrics in the papers reviewed above. However, to compare ML algorithms fairly, they should be evaluated on a common standardized dataset. Also, as the author in Agarwal et al. (2019) states, supervised ML models such as random forests and SVMs, take advantage of features custom-made for specific telescopes or surveys. Thus, it is not possible to have a standard dataset. However, in general, one can say that random forests and CNNs seem to be the more successful of the models presented in the literature for classifying FRBs in terms of model evaluation scores which are consistent throughout different papers and datasets.

Table 1: Summary of Papers on FRB Classification

Papers	ML Model	Data Source	Accuracy	Precision	Recall
Wagstaff et al. (2016)	Random Forest	V-FASTR	96%	97%	96%
Foster et al. (2018)	Random Forests	ALFA-BURST	$> 99\%$	92%	96%
Connor & van Leeuwen (2018)	Random Forest (freq-time)	Apertif	97%	97%	97%
Connor & van Leeuwen (2018)	SVM (freq-time)	Apertif	96%	94%	98%
Connor & van Leeuwen (2018)	CNNs	CHIME - pathfinder	$> 99\%$	$> 99\%$	$> 99\%$
Connor & van Leeuwen (2018)	CNNs	Apertif	$> 99\%$	$> 99\%$	$> 99\%$
Agarwal et al. (2019)	CNNs	Green Bank Telescope	$> 99\%$	$> 99\%$	$> 99\%$

7 Classifying Repeating and Non-Repeating FRBs

7.1 Supervised and Unsupervised Models

FRBs are categorized into repeating and non-repeating bursts, a classification facilitated by machine learning (ML) techniques. In the papers by Zhu-Ge et al. (2022) and Luo et al. (2022), a range of supervised and unsupervised ML algorithms are explored for the first CHIME/FRB catalogue (Amiri et al., 2021). These papers aim to determine any potential dichotomies with repeating and non-repeating FRBs. They report potentially repeating FRBs that were initially classified as non-repeating. Notably, the authors identify brightness temperature and rest-frame frequency bandwidth as the two most significant distinguishing factors between non-repeating and repeating FRBs.

In this co-study, the authors state that a non-repeating FRB could be a hidden repeating FRB yet to be seen repeating, but a repeating FRB could be a misidentified non-repeating one. it is unknown whether the apparently non-repeating FRBs are actually repeating FRBs whose repetitions are yet to be discovered, or whether they belong to another physically distinct type from the repeating ones Luo et al. (2022). A model that produces some false positives while yielding a few false negatives should not be rejected. Thus, the authors of these papers use $\beta = 2$ to prioritize recall over precision. This metric is referred to as the F_2 score. Tables 2 and 3 summarize the model evaluation metrics presented in the two papers.

Table 2: List of average F2 scores and standard deviations obtained with different supervised models (Zhu-Ge et al., 2022).

Model	F2 Mean	F2 SD
Decision tree (all features)	0.7351	0.0632
Decision tree (TB and ν)	0.7369	0.0499
Random forest	0.7821	0.0643
AdaBoost	0.7666	0.0617
LightGBM	0.7832	0.0647
XGBoost	0.7843	0.0631
SVM	0.8180	0.0483
Nearest centroid	0.7147	0.0614

Table 3: Evaluation metrics for each model. High scores in F2 indicate good performances of the three workflows. FN represents misclassified repeaters and FP represents repeater candidates (Luo et al., 2022).

Method	TP	FN	FP	TN	Recall	Precision	F2
PCA+k-means	85	9	127	373	0.9042	0.4009	0.7227
t-SNE+HDBSCAN	81	13	117	383	0.8617	0.4091	0.7056
UMAP+HDBSCAN	81	13	117	383	0.8617	0.4041	0.7056

7.2 Deep learning

In their study, Zhang et al. (2018) reports the identification of 72 new pulses originating from the repeating FRB 121102. These discoveries were made through Breakthrough Listen C-band (4-8 GHz) observations at the Green Bank Telescope and by employing CNNs. This method significantly surpasses previous detections, which had identified only 21 bursts. Neural network detection with dedispersion verification showcases notable advantages over conventional dedispersion algorithms. These benefits include heightened sensitivity, reduced false positive rates, and accelerated computational speeds, highlighting a substantial advancement in FRB detection techniques.

Compared to everyday objects such as cars or people, the quadratic form of FRBs is relatively straightforward. These basic characteristics are captured in the early layers of the neural network. However, the network must possess adequate capacity to handle high pixel-noise levels, particularly as one aims to detect signals at or below the noise amplitude (*even if the signal is below the noise amplitude, the signal may still have certain characteristics, such as specific frequency patterns or temporal behaviours, that can differentiate it from random noise*). Thus, Zhang et al. (2018) decides to use residual networks because of their ability to handle very deep architectures effectively. In FRB pulse detection, it's important to process signals with high pixel noise and capture low-level features like quadratic bursts efficiently. Residual Networks provide a solution to this challenge by utilizing skip connections, which helps mitigate the vanishing gradient problem and allows for the training of much deeper networks without encountering issues like overfitting. The author reports an overall recall of 88% and precision of 98% for the test set. In this case, recall represents the percentage of signals detected and precision is the percentage of signals that are real signals.

7.3 Discussion

In the papers Luo et al. (2022) and Zhu-Ge et al. (2022), they use classical ML for repeating and non-repeating FRB classification. Both studies highlight the significant features such as brightness temperature and rest-frame frequency bandwidth, in distinguishing between non-repeating and repeating FRBs. The best model that performed the best in Table 2 is the SVM model. The Random forest model also performs quite well which is consistent with other papers' results. The low precision numbers for all models in Table 3 and overall lower F2 scores than Table 2 suggest that unsupervised ML models perform worse than supervised ML models in classifying repeating and non-repeating FRBs. One key distinction between these papers and Zhang et al. (2018), is that Luo et al. (2022) and Zhu-Ge et al. (2022) highlight the features that are important in potentially distinguishing repeating from non-repeating FRBs. Thus, not only do these two papers showcase that ML can classify FRBs, but they also demonstrate that Ml can provide some insight into important and inherent properties of FRBs that may help one distinguish repeating from non-repeating. Zhang et al. (2018), however, demonstrates the effectiveness of deep neural networks and the use of residual networks in capturing low-level features such as quadratic bursts effectively. Deep learning techniques show a high level of potential in enhancing FRB detection sensitivity, reducing false positive rates, and accelerating computational speeds.

8 Conclusion

In conclusion, this literature review provides a comprehensive overview of the current state of research regarding machine learning (ML) techniques for classifying fast radio bursts (FRBs). Of the classical ML models, the random forest algorithm seems to be the most consistent among the literature in terms of high evaluation metrics. In contrast, for deep learning, convolutional neural networks (CNN) are used consistently across the literature. The application of CNN has shown remarkable advancements in FRB detection sensitivity, false positive rate reduction, and computational efficiency. Agarwal et al. (2019) has demonstrated the effectiveness of using transfer learning in tandem with CNNs while Zhang et al. (2018) has displayed the effectiveness of residual networks with CNNs. A lot of the literature uses simulated FRBs, thus future work could involve using more real data on FRBs as more discoveries and advancements of FRBs are made. Other future works could involve exploring methods for fusing and integrating data from multiple radio telescopes and instruments to improve FRB detection sensitivity and localization accuracy.

References

Abdi, H., & Williams, L. J. (2010). Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, *2*(4), 433–459.

Ackermann, S., Schawinski, K., Zhang, C., Weigel, A. K., & Turp, M. D. (2018). Using transfer learning to detect galaxy mergers. *Monthly Notices of the Royal Astronomical Society*, *479*(1), 415–425.

Agarwal, D., Aggarwal, K., Burke-Spolaor, S., Lorimer, D. R., & Garver-Daniels, N. (2019). Towards deeper neural networks for fast radio burst detection. *arXiv preprint arXiv:1902.06343*.

Amiri, M., Andersen, B. C., Bandura, K., Berger, S., Bhardwaj, M., Boyce, M. M., … Zwaniga, A. V. (2021, December). The first chime/frb fast radio burst catalog. *The Astrophysical Journal Supplement Series*, *257*(2), 59. Retrieved from `http://dx.doi.org/10.3847/1538-4365/ac33ab` doi: 10.3847/1538-4365/ac33ab

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123-140.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5-32. Retrieved from

Campello, R. J., Moulavi, D., & Sander, J. (2013). Density-based clustering based on hierarchical density estimates. In *Pacific-asia conference on knowledge discovery and data mining* (pp. 160–172).

Champion, D., Petroff, E., Kramer, M., Keith, M., Bailes, M., Barr, E., … others (2016). Five new fast radio bursts from the htru high-latitude survey at parkes: first evidence for two-component bursts. *Monthly Notices of the Royal Astronomical Society: Letters*, *460*(1), L30–L34.

Connor, L., & van Leeuwen, J. (2018). Applying deep learning to fast radio burst classification. *The Astronomical Journal*, *156*(6), 256.

Cortes, C., & Vapnik, V. (1995). Support vector networks. *Machine Learning*, *20*, 273-297.

Czech, D., Mishra, A., & Inggs, M. (2018). A cnn and lstm-based approach to classifying transient radio frequency interference. *Astronomy and Computing*, *25*, 52–57. doi: 10.1016/j.ascom.2018.07.00

Eldan, R., & Shamir, O. (2016). The power of depth for feedforward neural networks. In *Conference on learning theory* (pp. 907–940).

Foster, G., Karastergiou, A., Golpayegani, G., Surnis, M., Lorimer, D. R., Chennamangalam, J., … others (2018). Alfaburst: A commensal search for fast radio bursts with arecibo. *Monthly Notices of the Royal Astronomical Society*, *474*(3), 3847–3856.

Friedman, N., Geiger, D., & Goldszmidt, M. (1997, 11). Bayesian network classifiers. *Machine Learning*, *29*, 131-163. doi: 10.1023/A:1007465528199

Gajjar, V., Siemion, A., Price, D., Law, C., Michilli, D., Hessels, J., … others (2018). Highest frequency detection of frb 121102 at 4–8 ghz using the breakthrough listen digital backend at the green bank telescope. *The Astrophysical Journal*, *863*(1), 2.

Geman, S., Bienenstock, E., & Doursat, R. (1992, 01). Neural networks and the bias/variance dilemma. *Neural Computation*, *4*, 1-58. doi: 10.1162/neco.1992.4.1.1

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning.* MIT Press. (`http://www.deeplearningbook.org`)

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction, second edition (springer series in statistics).*

He, K., Zhang, X., Ren, S., & Sun, J. (2016a). Deep residual learning for image recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 770–778).

He, K., Zhang, X., Ren, S., & Sun, J. (2016b). Identity mappings in deep residual networks. In *Computer vision–eccv 2016: 14th european conference, amsterdam, the netherlands, october 11–14, 2016, proceedings, part iv 14* (pp. 630–645).

Hofmann, T., Schölkopf, B., & Smola, A. J. (2008, June). Kernel methods in machine learning. *The Annals of Statistics*, *36*(3). Retrieved from `http://dx.doi.org/10.1214/009053607000000677` doi: 10.1214/009053607000000677

Hosna, A., Merry, E., Gyalmo, J., Alom, Z., Aung, Z., & Azim, M. A. (2022). Transfer learning: a friendly introduction. *Journal of Big Data*, *9*(1), 102.

Khan, A., Huerta, E., Wang, S., Gruendl, R., Jennings, E., & Zheng, H. (2019). Deep learning at scale for the construction of galaxy catalogs in the dark energy survey. *Physics Letters B*, *795*, 248–258.

Kodinariya, T. M., Makwana, P. R., et al. (2013). Review on determining number of cluster in k-means clustering. *International Journal*, *1*(6), 90–95.

Lorimer, D. R., Bailes, M., McLaughlin, M. A., Narkevic, D. J., & Crawford, F. (2007, November). A bright millisecond radio burst of extragalactic origin. *Science*, *318*(5851), 777–780. Retrieved from `http://dx.doi.org/10.1126/science.1147532` doi: 10.1126/science.1147532

Luo, J.-W., Zhu-Ge, J.-M., & Zhang, B. (2022, November). Machine learning classification of chime fast radio bursts – i. supervised methods. *Monthly Notices of the Royal Astronomical Society*, *518*(2), 1629–1641. Retrieved from `http://dx.doi.org/10.1093/mnras/stac3206` doi: 10.1093/mnras/stac3206

McInnes, L., Healy, J., & Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Mhaskar, H., Liao, Q., & Poggio, T. (2016). Learning functions: when is deep better than shallow. *arXiv preprint arXiv:1603.00988*.

Neal, B. (2019). *On the bias-variance tradeoff: Textbooks need an update.*

Pérez-Carrasco, M., Cabrera-Vives, G., Martinez-Marin, M., Cerulo, P., Demarco, R., Protopapas, P., . . . others (2019). Multiband galaxy morphologies for clash: a convolutional neural network transferred from candels. *Publications of the Astronomical Society of the Pacific*, *131*(1004), 108002.

Petroff, E., Bailes, M., Barr, E., Barsdell, B., Bhat, N., Bian, F., ... others (2015). A real-time fast radio burst: polarization detection and multiwavelength follow-up. *Monthly Notices of the Royal Astronomical Society*, *447*(1), 246–255.

Petroff, E., Burke-Spolaor, S., Keane, E., McLaughlin, M., Miller, R., Andreoni, I., ... others (2017). A polarized fast radio burst at low galactic latitude. *Monthly Notices of the Royal Astronomical Society*, *469*(4), 4465–4482.

Petroff, E., Hessels, J., & Lorimer, D. (2019). Fast radio bursts. *The Astronomy and Astrophysics Review*, *27*(1), 4.

Rickett, B. J. (1977). Interstellar scattering and scintillation of radio waves. *In: Annual review of astronomy and astrophysics. Volume 15.(A78-16576 04-90) Palo Alto, Calif., Annual Reviews, Inc., 1977, p. 479-504.*, *15*, 479–504.

Rickett, B. J. (1990). Radio propagation through the turbulent interstellar plasma. *Annual review of astronomy and astrophysics*, *28*(1), 561–605.

Rijsbergen, C. J. V. (1979). *Information retrieval* (2nd ed.). Butterworth-Heinemann.

Sasaki, Y. (2007, 01). The truth of the f-measure. *Teach Tutor Mater*.

Shalev-Shwartz, S., & Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms.* Cambridge University Press.

Shannon, R., Macquart, J.-P., Bannister, K. W., Ekers, R., James, C., Osłowski, S., ... others (2018). The dispersion–brightness relation for fast radio bursts from a wide-field survey. *Nature*, *562*(7727), 386–390.

Spitler, L., Scholz, P., Hessels, J., Bogdanov, S., Brazier, A., Camilo, F., ... others (2016). A repeating fast radio burst. *Nature*, *531*(7593), 202–205.

Tang, H., Scaife, A. M., & Leahy, J. (2019). Transfer learning for radio galaxy classification. *Monthly Notices of the Royal Astronomical Society*, *488*(3), 3358–3375.

Tendulkar, S. P., Bassa, C., Cordes, J. M., Bower, G. C., Law, C. J., Chatterjee, S., ... others (2017). The host galaxy and redshift of the repeating fast radio burst frb 121102. *The Astrophysical Journal Letters*, *834*(2), L7.

Thakur, P. (2020). *Comparative analysis of machine learning algorithms for detection of fast radio bursts* (Unpublished master's thesis). Dublin Business School.

Thornton, D. e., Stappers, B., Bailes, M., Barsdell, B., Bates, S., Bhat, N., ... others (2013). A population of fast radio bursts at cosmological distances. *Science*, *341*(6141), 53–56.

Ting, K. M. (2010). Confusion matrix. In C. Sammut & G. I. Webb (Eds.), *Encyclopedia of machine learning* (pp. 209–209). Boston, MA: Springer US. Retrieved from `https://doi.org/10.1007/978-0-387-30164-8`$_1$57 doi: 10.1007/978-0-387-30164-8$_1$57

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, *9*(11).

Vilalta, R. (2018). Transfer learning in astronomy: A new machine-learning paradigm. In *Journal of physics: Conference series* (Vol. 1085, p. 052014).

Wagstaff, K. L., Tang, B., Thompson, D. R., Khudikyan, S., Wyngaard, J., Deller, A. T., ... Wayth, R. B. (2016, jun). A machine learning classifier for fast radio burst detection at the vlba. *Publications of the Astronomical Society of the Pacific*, *128*(966), 084503. Retrieved from `https://dx.doi.org/10.1088/1538-3873/128/966/084503` doi: 10.1088/1538-3873/128/966/084503

Zhang, Y. G., Gajjar, V., Foster, G., Siemion, A., Cordes, J., Law, C., & Wang, Y. (2018). Fast radio burst 121102 pulse detection and periodicity: a machine learning approach. *The Astrophysical Journal*, *866*(2), 149.

Zhu-Ge, J.-M., Luo, J.-W., & Zhang, B. (2022, December). Machine learning classification of chime fast radio bursts – ii. unsupervised methods. *Monthly Notices of the Royal Astronomical Society*, *519*(2), 1823–1836. Retrieved from `http://dx.doi.org/10.1093/mnras/stac3599` doi: 10.1093/mnras/stac3599