The Impact of Domain Knowledge on Applying Machine Learning Methods to Exoplanet Detection

The-Gia Leo Nguyen [1]

Abstract: Exoplanets do not emit electromagnetic waves which makes it challenging to detect them. Based on transit photometry, we trained a neural network on NASA Kepler space telescope data to detect exoplanets based on light intensity curves. We showcase, that with a well designed data pipeline, a small neural network is sufficient to achieve state-of-the-art performance, saving both computation time and hardware cost. The strongest improvement in performance could only be achieved by adding domain specific processing steps to the data pipeline. Domain knowledge was essential in selecting the appropriate machine learning concepts that are beneficial to solving the problem and have a higher impact on the performance than the actual classification method itself. We encourage to consider the data pipeline as an additional component, besides the classification model, that can potentially improve the overall performance.

Keywords: Machine Learning; Neural Networks; Deep Learning; Astrophysics; Exoplanets; Artificial Intelligence

1 Introduction

The idea of exoplanets was proposed early on. Dating back to the 16th century, Giordano Bruno suggested that fixed stars are similar to the Sun and likewise would have planets orbiting them. However, exoplanets do not emit light on their own and given the proximity to their star, it is hardly possible to detect them directly. The first confirmed exoplanet detection only occurred in 1992. According to NASA [2], 4375 exoplanet detections have been confirmed of which over 80% were detected in the last seven years.

Given the enormous amount of stars in the universe, we want to use machine learning methods to recommend interesting stars for further investigation by astrophysicists. Our model will be trained on data from the NASA Kepler missions. This approach is rationalized by two main observations. First, the NASA Kepler space telescope is specifically designed for detecting exoplanets with transit photometry and has detected over 60% of all confirmed exoplanets. Second, 76% of all exoplanet detections have been made by transit photometry. This indicates that training a machine learning model to make predictions based on light intensity curves used for transit photometry might be the most promising approach.

[1] Heidelberg University, Germany
[2] as of 7th of April 2021

Besides achieving state-of-the-art performance, we also analyze in depth the effectiveness of commonly used machine learning approaches to boost performance in the application of exoplanet detection. Our investigation shows the importance of domain knowledge when applying machine learning methods to real world problems. In most cases, machine learning methods will not provide the desired results *out of the box*.

2 Related Works

The idea of applying machine learning and deep learning methods to the problem of exoplanet detection is not new and has been proposed by several researchers. Zucker et al. [ZG18] conducted a feasibility study on the application of Convolutional Neural Networks on exoplanet detection with simulated data. Further works that applied Convolutional Neural Networks to exoplanet detection have been done by [SV18], [YJ21] and [CJ19]. Ensemble models for exoplanet detection have been introduced by Priyadarshini et al. [PP21] with Ensemble-CNNs and Malik et al. [MMO20] with gradient boosting classifiers. Sturrock et al. [SMR] and Schanche et al. [Sc19] analyzed and compared the application of several machine learning methods on exoplanet detection, including Support Vector Machines, K-Nearest neighbor, Random Forest Classifier, Logistic Regression Classifier and Convolutional Neural Networks. Ofman et al. [Of21] applied ThetaRay Fintech algorithms, originally used for anomaly detection in financial institutions, on exoplanet detection. Tsang et al. [TS19] proposed a recurrent neural network autoencoder for unsupervised feature extraction combined with an estimation network for supervised classification and novelty detection. Jara-Maldonado et al. [Ja20] surveyed different machine learning methods for exoplanet detection and propose a model to generate synthetic light curves which can be used as training data. Mislis et al. [MPA18] introduced an unsupervised method for exoplanet detection using the DBSCAN clustering algorithm.

Most of previous research has been focusing on the classification model itself. However, not much attention has been given to the data pipeline that comes before the classification model. In this work, we analyze different pipelines and their effect on the overall performance. In particular, we will investigate the effect of different oversampling methods, data augmentation and domain-specific processing steps. Additionally, we show that most machine learning methods do not work *out of the box* and are highly problem dependent. This makes domain knowledge essential for selecting the appropriate data pipeline for machine learning based methods.

3 Theory

3.1 Transit Photometry

Transit photometry uses the fact that a planet passing directly between its star and the observer would dim the star's light periodically resulting in measurable dips of brightness.

Our goal is to design an algorithm using machine learning techniques to recognizes specific patterns in the light intensity curve of stars accompanied by exoplanets and classify them as such. The probability of a grazing transit or full transit to be observable by a spectator for randomly oriented orbits is given by:

$$P\left(cos(i) < \frac{R_s + R_p}{a}\right) = \frac{1}{2}\int_{-\frac{R_s+R_p}{a}}^{\frac{R_s+R_p}{a}} = \frac{R_s + R_p}{a} \tag{1}$$

where R_p is the radius of the exoplanet, R_s the radius of the hosting star, a the semi-major axis of the orbit and i the angle between the line of sight and the angular-momentum vector of the exoplanet's orbit. Despite transit photometry being the most successful method for detecting exoplanets, the probability of an exoplanet actually orbiting its star in such a way, that it is observable to the spectator, is not exceptionally high. This makes exoplanets one of the most difficult objects to detect in the universe. As an example the transit of Earth is only visible from 0.46 percent of the celestial sphere [BMS16].

3.2 Imbalanced Dataset

Since exoplanets are hard to detect and only 4375 confirmed exoplanet detections [3] have been made so far, we expect our dataset to be highly imbalanced. Assuming a dataset with only 1% of the data points being stars with confirmed exoplanets, an algorithm that classifies every star to not have any exoplanets would already achieve a prediction accuracy of 99%. Even though, accuracy is commonly used as a performance metric in machine learning classification tasks, it will not be very indicative of the real performance in our application. For imbalance datasets, precision [4] and recall [5] are often used instead. In our experiments, we will consider the F1 score as our main performance metric which is the harmonic mean between precision and recall:

$$\text{F1 score} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2}$$

Choosing an appropriate performance metric alone, will not prevent the classification model from always predicting the majority class. A typical approach to deal with an imbalanced dataset, is to oversample the minority class. The most naive approach is random oversampling. By creating copies of random samples from the minority class, our classification model will see those samples more often during the training process. A more sophisticated way to oversample the minority class was introduced by Chawla et al. [Ch11] with the SMOTE algorithm. The main idea is to adjust the class distribution of the dataset by generating new data points of the minority class. To generate a new data point x_{new}, we randomly pick a

[3] as of 7th of April 2021
[4] ratio of true positives against all examples that were classified as true
[5] ratio of true positives against all positive examples

data point $x_i \in \{x_1, ..., x_n\}$, where $\{x_1, ..., x_n\}$ is the set of all data points which belongs to the minority class. Then, we compute the k-nearest neighbors of x_i in the feature space and randomly select one of the k-nearest neighbor as our data point x_{zi}. The last step includes an interpolation, where we generate the new data point x_{new} such that it is located on an imaginary line connecting the data points x_i and x_{zi}:

$$x_{new} = x_i + \lambda \cdot (x_{zi} - x_i) \tag{3}$$

where λ is randomly generated to be in the range $[0, 1]$. Two variants of the original SMOTE algorithm propose to generate new points at the border of the minority class, since those points will make the most impact on the decision boundary. One of the variants proposed by Han et al. [HWM05] uses K-nearest neighbors to identify borderline points. In the following, this method will be referred to as Borderline-SMOTE. The other variant proposed by Nguyen et al. [NCK11] uses support vector machines instead of K-nearest neighbors and will be referred to as SVM-SMOTE. The ADASYN oversampling algorithm proposed by He et al. [He08] follows the same idea as SMOTE. It also emphasizes borderline points, but the main difference is the noise added to the generated points. The idea is that removing the linear dependency from existing data points allows for generation of more realistic samples.

3.3 Data Augmentation

For machine learning methods, the quality and amount of data available plays an important role. Data augmentation is a common approach to boost performance by generating more synthetic data from the existing dataset. For time series data, Guennec et al. [GMT16] have shown in their experiments that *window sliding* and *time warping* are viable data augmentation methods for time series classification tasks using neural networks. The main idea of *window sliding* is to slide a window of smaller size over the time series creating new shorter time slices. These slices carry the same label as its source time series. At testing time, we will slide a window over the testing time series and perform classification on each window. The final prediction is then decided by majority vote. The main idea of *time warping* is to select random slices which will be either stretched or compressed. Additionally, Gaussian noise will be added to randomly selected data points.

3.4 Fully-connected Neural Network

The goal of our neural network is to learn an approximated mapping f between the light curves x from NASA Kepler Telescope and the labels y denoting if the star has an exoplanet. A neural network can be interpreted a directed acyclic graph, where each node n is a computational unit often referred to as a *neuron*. Each node n takes an input vector I and

performs the following computation with its internal weights (W, b) [6] to output a single output value o:

$$o = \phi(W^T \cdot I + b) \tag{4}$$

where ϕ is an arbitrary non-linear function called *activation function*. In our neural network architecture the ReLU [NH10] is used as an activation function for all neurons. Each layer in a neural network is constructed by stacking multiple neurons. As a result, the output of each layer is a vector, where every entry is the output scalar value o_i of the corresponding i-th neuron in the layer. This output vector can then be given as input for the next neural network layer. Using *softmax* and *cross entropy* as our loss function, we can compare the neural network output $\hat{y} = f(x, \theta)$ [7] with the ground truth label y. Adam optimization [KB14] will be used to minimize our loss function by optimizing the weights θ and is often referred to as *training*. At testing time, our model is evaluated on previously unseen data, not used during training, on both accuracy and F1 score to determine its generalization ability.

4 Methodology

4.1 Dataset

For our experiments, we will be using a Kaggle dataset [8] which contains data from NASA Kepler space telescope Campaign-3 mission taken from Mikulski Archive for Space Telescopes [9]. According to the creator of the Kaggle dataset, Campaign-3 was picked because it is unlikely to contain any undiscovered exoplanet-stars and therefore would minimize the probability of wrongly labeled data points. Additionally, the number of stars with exoplanets was boosted by including data from other Kepler Campaigns. Even after boosting, the dataset is still highly unbalanced with over 99% of the data being observations of stars without exoplanets. The trainset consists of 5078 flux diagrams of which only 37 are confirmed exoplanet-stars. The testset consists of 570 flux diagrams of which only 5 are confirmed exoplanet-stars. However, we will not use the train-test-split provided by Kaggle for our experiments, but instead perform stratified k-fold cross-validation. The main reason is that the train-test split provided on Kaggle with a ratio of 90-10 is prone to overfitting and selection bias, which might not represent the true generalization ability of our model. Instead with stratified k-fold cross-validation, we will split our dataset into 10 equal subsets with approximately the same ratio between confirmed exoplanet stars and non-exoplanet stars. The training and testing procedure will then be executed 10 times, where each time another subset is chosen as the testset and the remaining 9 subsets will be used as training data. The performance is then calculated by taking the mean of all 10 runs.

[6] W is a vector and b is a scalar

[7] θ are the weights of the whole neural network consisting of all (W, b) of all neurons combined

[8] https://www.kaggle.com/keplersmachines/kepler-labelled-time-series-data

[9] https://archive.stsci.edu/

4.2 Architecture and Training

The classification model will be kept relatively small with a 3-layer neural network [GBC16] of hidden size 100 with dropout [Sr14], batch normalization [IS15] and ReLU activation functions [NH10]. This model was design such that it can be trained in reasonable time on a CPU [10] without the need for any GPUs. As a comparison the AstroNet by Shallue et al. [SV18] consists of 14 convolutional layers, 5 fully-connected layers and 7 pooling layers. A smaller network has a lower representation ability, but requires significantly less resources to train and evaluate. We initialize the weights with Xavier initialization. For training we use Adam optimization [KB14] with a learning rate of 0.01 and a batch size of 1024 trained for a total of 100 epochs. We employ adaptive learning by reducing the learning rate by a factor 0.5 at epoch 20, 40, 55, 70, 80, 90 and 95. Further implementation details can be deducted from our code [11].

4.3 Domain specific pre-processing

On every flux diagram, we will independently apply normalization and standardization, since we are only interested in the periodically reoccurring dips and not in the light curves amplitude. Note that the normalization is applied per flux diagram and not over the whole dataset. Besides normalization and standardization, we will apply domain specific pre-processing steps using domain knowledge. Analyzing its performance impact can give insight into the significance of domain knowledge for applying machine learning methods to real world problems.

In transit photometry, we are looking for periodically reoccurring dips whenever an exoplanet passes the line of sight between star and telescope. We propose to apply a Fourier transform [Ob07] to the flux curves. For an input vector $(x_0, ..., x_{2n-1})$ with a dimension of $2n$ the discrete Fourier transformed output vector $(f_0, ..., f_{2n-1})$ is given by:

$$f_m = \sum_{k=0}^{2n-1} x_k \cdot exp\left(-\frac{2\pi i}{2n} mk\right) \tag{5}$$

where $m \in \{0, ..., 2n-1\}$. This means that the light curves, which are functions of time, will be transformed into functions of frequency. In other words, those Fourier transformed functions measure how prominent certain frequencies appear in the original light curve. We argue that this frequency domain makes it much easier for the classification model to detect the periodical dips, since periodicity and frequency are inherently connected. Furthermore, by only considering the absolute value of the Fourier transform, we also compensate for possible time shifts between data from different stars, since phase information is ignored.

[10] under 2 minutes on a AMD Ryzen 7

[11] https://github.com/Xenovortex/exoplanet_detection

Based on transit photometry, we expect exoplanets to create periodically occurring dips of flux in the negative direction. Therefore, we will only remove outliers in the positive direction with a deviation higher than 5σ to avoid removing sensible effects introduced by the presence of an exoplanets. Note that the outlier removal only applies to points within a flux diagram. The total number of flux diagrams stays the same and all flux diagrams in the dataset were used in the conducted experiments. Since the obtained data after Fourier transform is still quite noisy, we will apply a Gaussian filter afterwards.

5 Results

5.1 Domain Knowledge and Oversampling

We evaluate the performance of all oversampling techniques with different pre-processing settings as presented in table 1. The best performing oversampling method depends on the pre-processing steps performed on the data. However, we find that overall the performance depends more on the pre-processing pipeline than the actual oversampling method.

The main improvements can be observed through pre-processing, where Fourier Transform plays an essential part. Without Fourier Transform, the F1 score will fall under 0.2 irrespective of other included pre-processing steps. However, for optimal performance both Fourier Transform and Gaussian Smoothing are required independent from the oversampling method used. These results are aligned with our expectations. Based on transit photometry, we were looking for periodic dips in the flux curves. This makes the Fourier frequency space the perfect feature space to train our model on and at the same time acts as a normalization on the flux curves.

In table 2, we replace our neural network classification model by other machine learning methods such as K-nearest Neighbor, Support Vector Machine and Random Forest Classifiers and show that our proposed inclusion of Fourier transform and Gaussian Smoothing into the data pipeline can significantly improve performance for a variety of machine learning based classification models. With a performance increase of about 25% to 60% in F1 Score, the classical machine learning methods show less improvements than for the fully-connected neural network model. We hypothesize that our neural network model has a higher representation power than the classical machine learning methods and thereby is more prone to overfitting by picking up unimportant details in the data, that are not related to the classification task. Both Fourier Transform and Gaussian Smoothing will help remove those unrelated details by changing the feature space, since the frequency domain directly represent the periodicity we are looking for.

Oversampling Technique	Test Accuracy	Test F1 Score
Without Pre-processing		
Without Oversampling	0.993	0.000
Random	0.987	0.100
SMOTE	0.986	0.141
Borderline-SMOTE	0.988	0.165
SVM-SMOTE	0.987	0.102
ADASYN	0.987	0.110
Normalization + Outlier Removal		
Without Oversampling	0.992	0.040
Random	0.992	0.033
SMOTE	0.991	0.123
Borderline-SMOTE	0.991	0.135
SVM-SMOTE	0.992	0.062
ADASYN	0.992	0.124
Normalization + Outlier Removal + Fourier Transform		
Without Oversampling	0.996	0.655
Random	0.995	0.510
SMOTE	0.995	0.470
Borderline-SMOTE	0.995	0.492
SVM-SMOTE	0.995	0.573
ADASYN	0.995	0.528
Normalization + Outlier Removal + Gaussian Smoothing		
Without Oversampling	0.992	0.000
Random	0.993	0.100
SMOTE	0.991	0.089
Borderline-SMOTE	0.991	0.000
SVM-SMOTE	0.993	0.089
ADASYN	0.993	0.186
Normalization + Outlier Removal + Fourier Transform + Gaussian Smoothing		
Without Oversampling	0.998	0.810
Random	0.999	0.843
SMOTE	0.999	0.857
Borderline-SMOTE	0.998	0.824
SVM-SMOTE	0.998	0.790
ADASYN	**0.999**	**0.905**

Tab. 1: Mean Test Accuracy and Mean Test F1 Score for different pre-processing pipelines and oversampling techniques using stratified k-fold cross-validation with $k = 10$

Method	With FT and GS		Without FT and GS	
	Test Accuracy	Test F1 Score	Test Accuracy	Test F1 Score
K-nearest Neighbor	0.995	0.725	0.970	0.581
Support Vector Machines	0.998	0.868	0.992	0.535
Random Forest	0.996	0.731	0.895	0.494

Tab. 2: Mean Test Accuracy and Mean Test F1 Score comparison on different machine learning methods with and without Fourier transform (FT) + Gaussian Smoothing (GS) using stratified k-fold cross-validation with $k = 10$

5.2 Data Augmentation

Table 3 shows our results for the window sliding experiment. We observe a significant performance decrease, when using window sliding irrespective of the chosen window size. Despite window sliding being a valid data augmentation technique in the context of machine learning, we have to consider domain knowledge when applying machine learning methods to real world problems. From the physics perspective, we know through transit photometry that we are looking for periodical dips in flux when an exoplanet is between the observer and the star. This requires the star to be observed for at least a multiple of the exoplanet's orbital period. For context, the orbital period of Earth is roughly 1 year and for Neptune 165 years. Using window sliding, the neural network will only look at a small window, making all exoplanets with orbital periods larger than the window size undetectable.

The results for time warping are summarized in table 4. Time warping will be applied with a given probability to randomly selected segments during training. We observe a performance decrease for higher probabilities. Therefore, we conclude that there is no apparent advantage in using time warping, since it only worsen the performance. From the physics perspective, we assume that randomly stretching and compressing parts of the time series will interfere with the periodicity of the dips.

Window Size	Test Accuracy	Test F1 Score
Without WS	**0.999**	**0.905**
100	0.976	0.003
500	0.993	0.000
1000	0.993	0.000

Tab. 3: Mean Test Accuracy and Mean Test F1 Score for Window Sliding (WS) using stratified k-fold cross-validation with $k = 10$ and the full pre-processing pipeline (Normalization + Outlier Removal + Fourier Transform + Gaussian Smoothing)

5.3 Comparison to other Kaggle solutions

For the comparison with other Kaggle solutions, we will employ normalization, outlier removal, Fourier transform and Gaussian smoothing as pre-processing steps with ADASYN

Probability	Test Accuracy	Test F1 Score
0	**0.999**	**0.905**
0.1	0.998	0.813
0.3	0.997	0.649
0.8	0.995	0.378

Tab. 4: Mean Test Accuracy and Mean Test F1 Score for Time Warping using stratified k-fold cross-validation with $k = 10$ and the full pre-processing pipeline (Normalization + Outlier Removal + Fourier Transform + Gaussian Smoothing)

for oversampling. We do not use any data augmentation, since our experiments show that they only negatively impinge the performance. Since most of the solutions on Kaggle do not use stratified k-fold cross-validation, but use the train-test split provided on Kaggle, we additionally evaluate our model on the given Kaggle split. Table 5 shows our test performance in comparison with other top solutions retrieve from Kaggle. Our method beats every solution on both accuracy and F1 score achieving state-of-the-art performance. Even our stratified k-fold cross-validation results still beat all other methods. This showcase that using domain knowledge to design appropriate pre-processing pipelines can greatly improve the performance of machine learning models.

Approach	Test Accuracy	Test F1 Score
Juan Felipe (XGBoost)	–	0.88
Travillion (SVC)	–	0.73
Gabriel Garza (SVC)	–	0.71
Alexei D. (SVC)	–	0.67
Rahul Singh (Random Forest)	0.991	0.5
Peter Grenholm (CNN)	0.998	–
Rahul Misal (Naive Bayes)	0.991	–
Amajid Sinar (CNN)	0.991	–
Keyur Paralkar (MLP)	0.990	–
Ours (stratified k-fold)	**0.999**	**0.905**
Ours (Kaggle split)	**1.000**	**1.000**

Tab. 5: Performance Comparison of our method and top solutions on Kaggle

6 Conclusion

Through this work, we have demonstrated the importance of domain knowledge when applying machine learning methods to real world problems. Most of the time, established machine learning methods are very case dependent and can not be applied *out of the box* to every problem. This was apparent in our data augmentation experiments. Even though, *window sliding* and *time warping* are sensible approaches in the machine learning context, they are not constructive when considering the astrophysics behind exoplanet detection with transit photometry. In contrast, having domain knowledge on transit photometry allows us to

identify effective processing steps such as Fourier transform that allows for state-of-the-art performance with a smaller neural network architecture. In summary, we successfully applied machine learning techniques on the recent research topic of exoplanet detection. Our model achieves state-of-the-art performances beating all current approaches on the same dataset. As a use case, we see your model as a pre-selection algorithm that can recommend interesting star for further inspection by an expert astrophysicist.

7 Acknowledgment

We thank Paul Hill, Ullrich Köthe and Tsungnan Lin for the inspiration to this project. We thank Holger Fröning for bringing the SKILL conference to our attention. Furthermore, we thank the reviewers for the constructive criticism and recommendations.

Bibliography

[BMS16] Bozza, Valerio; Mancini, Luigi; Sozzetti, Alessandro: Methods of Detecting Exoplanets: 1st Advanced School on Exoplanetary Science. 01 2016.

[Ch11] Chawla, N. V.; Bowyer, K. W.; Hall, L. O.; Kegelmeyer, W. P.: SMOTE: Synthetic Minority Over-sampling Technique. arXiv e-prints, p. arXiv:1106.1813, June 2011.

[CJ19] Chintarungruangchai, Pattana; Jiang, Ing-Guey: Detecting Exoplanet Transits through Machine-learning Techniques with Convolutional Neural Networks. Publications of the Astronomical Society of the Pacific, 131(1000):064502, June 2019.

[GBC16] Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron: Deep Learning. MIT Press, 2016. `http://www.deeplearningbook.org`.

[GMT16] Guennec, Arthur Le; Malinowski, Simon; Tavenard, Romain: , Data Augmentation for Time Series Classification using Convolutional Neural Networks, 2016.

[He08] He, Haibo; Bai, Yang; Garcia, Edwardo; Li, Shutao: ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning. pp. 1322 – 1328, 07 2008.

[HWM05] Han, Hui; Wang, Wen-Yuan; Mao, Bing-Huan: Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. volume 3644, pp. 878–887, 09 2005.

[IS15] Ioffe, Sergey; Szegedy, Christian: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv e-prints, p. arXiv:1502.03167, February 2015.

[Ja20] Jara-Maldonado, Miguel; Alarcon-Aquino, Vicente; Rosas-Romero, Roberto; Starostenko, Oleg; Ramirez-Cortes, Juan: Transiting Exoplanet Discovery Using Machine Learning Techniques: A Survey. Earth Science Informatics, 09 2020.

[KB14] Kingma, Diederik P.; Ba, Jimmy: Adam: A Method for Stochastic Optimization. arXiv e-prints, p. arXiv:1412.6980, December 2014.

[MMO20] Malik, Abhishek; Moster, Benjamin P.; Obermeier, Christian: Exoplanet Detection using Machine Learning. arXiv e-prints, p. arXiv:2011.14135, November 2020.

[MPA18] Mislis, D.; Pyrzas, S.; Alsubai, K. A.: TSARDI: a Machine Learning data rejection algorithm for transiting exoplanet light curves. Monthly Notices of the Royal Astronomical Society, 481(2):1624–1630, December 2018.

[NCK11] Nguyen, Hien M.; Cooper, E.; Kamei, K.: Borderline over-sampling for imbalanced data classification. Int. J. Knowl. Eng. Soft Data Paradigms, 3:4–21, 2011.

[NH10] Nair, Vinod; Hinton, Geoffrey: Rectified Linear Units Improve Restricted Boltzmann Machines Vinod Nair. volume 27, pp. 807–814, 06 2010.

[Ob07] Oberst, Ulrich: The Fast Fourier Transform. SIAM J. Control and Optimization, 46:496–540, 01 2007.

[Of21] Ofman, Leon; Averbuch, Amir; Shliselberg, Adi; Benaun, Idan; Segev, David; Rissman, Aron: Automated identification of transiting exoplanet candidates in NASA Transiting Exoplanets Survey Satellite (TESS) data with machine learning methods. arXiv e-prints, p. arXiv:2102.10326, February 2021.

[PP21] Priyadarshini, Ishaani; Puri, Vikram: A convolutional neural network (CNN) based ensemble model for exoplanet detection. Earth Science Informatics, 14:1–13, 06 2021.

[Sc19] Schanche, N.; Collier Cameron, A.; Hébrard, G.; Nielsen, L.; Triaud, A. H. M. J.; Almenara, J. M.; Alsubai, K. A.; Anderson, D. R.; Armstrong, D. J.; Barros, S. C. C.; Bouchy, F.; Boumis, P.; Brown, D. J. A.; Faedi, F.; Hay, K.; Hebb, L.; Kiefer, F.; Mancini, L.; Maxted, P. F. L.; Palle, E.; Pollacco, D. L.; Queloz, D.; Smalley, B.; Udry, S.; West, R.; Wheatley, P. J.: Machine-learning approaches to exoplanet transit detection and candidate validation in wide-field ground-based surveys. Monthly Notices of the Royal Astronomical Society, 483(4):5534–5547, March 2019.

[SMR] Sturrock, George Clayton; Manry, Brychan; Rafiqi, Sohail: Machine Learning Pipeline for Exoplanet Classification. SMU Data Science Review, 2(1 , Article 9).

[Sr14] Srivastava, Nitish; Hinton, Geoffrey; Krizhevsky, Alex; Sutskever, Ilya; Salakhutdinov, Ruslan: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research, 15(56):1929–1958, 2014.

[SV18] Shallue, Christopher J.; Vanderburg, Andrew: Identifying Exoplanets with Deep Learning: A Five-planet Resonant Chain around Kepler-80 and an Eighth Planet around Kepler-90. Astronomical Journal, 155(2):94, February 2018.

[TS19] Tsang, Benny T. H.; Schultz, William C.: Deep Neural Network Classifier for Variable Stars with Novelty Detection Capability. Astrophysical Journal Letters, 877(2):L14, June 2019.

[YJ21] Yeh, Li-Chin; Jiang, Ing-Guey: Searching for Possible Exoplanet Transits from BRITE Data through a Machine Learning Technique. Publications of the Astronomical Society of the Pacific, 133(1019):014401, January 2021.

[ZG18] Zucker, Shay; Giryes, Raja: Shallow Transits - Deep Learning. I. Feasibility Study of Deep Learning to Detect Periodic Transits of Exoplanets. Astronomical Journal, 155(4):147, April 2018.