

IDENTIFYING EXOPLANETS FROM TRANSIT SURVEY DATA USING NEURAL NETWORKS

Joseph Bell, Maria Harris, and James Salem

University of California San Diego, La Jolla, CA 92093-0238,

ABSTRACT

In this paper, we discuss three methods for using machine learning to determine if a star has an exoplanet from transit survey data. The three models include a Convolutional Neural Network (CNN), a CNN autoencoder, and a Fully Connected Neural Network (FCNN). We used data from the Kepler Space Telescope to train the models, however, since there were very few exoplanet samples represented in the data, we used Synthetic Minority Oversampling Technique (SMOTE) to over-sample the exoplanet class. The results were promising for the supervised learning methods, CNN and FCNN, however, the unsupervised method, CNN autoencoder, did not produce meaningful results

Index Terms—Convolutional Neural Network, CNN, autoencoder, Fully Connected Neural Network, FCNN, Transit Survey, Exoplanet

1. INTRODUCTION

Searching for exoplanets is important because it can teach us a lot about our universe. For example, each exoplanet discovered helps us gain a better understanding of how common it is for planets to orbit other stars. Once an exoplanet is identified, scientists can study it further to determine the type of planet it might be (earth-like, gas giant, etc.) and how close it is to its star. The information that can be gained from discovering exoplanets is useful for learning about other solar systems and for finding earth-like planets that could harbor extraterrestrial life.

The most common method of searching for exoplanets is the transit survey method. A transit survey light curve consists of time-series light intensity measurements of a star. If an object like a planet travels between the star and the telescope taking measurements, one would expect to see a decrease in measured light intensity values as some of the light from the star was being blocked. In other words, when a planet travels in front of a star being observed, it casts a shadow that would be seen as a dip in light intensity on a light curve.

While finding dips in a light curve sounds like a simple task, it is often difficult because light curves can be fairly noisy and planets often do not block much light as they are generally significantly smaller than the stars they orbit. We

think machine learning, and particularly neural networks, would be a useful tool to help recognize the underlying light curve patterns that indicate whether or not a star has an exoplanet (or exoplanets).

In this paper, we compare three methods for classifying light curves using deep neural networks. The first method uses Principle Component Analysis (PCA) to reduce dimensions and a deep CNN as a classifier. The second method involves a deep CNN autoencoder used to detect anomalies. The last method uses a Fast Fourier Transform (FFT) as feature extraction followed by PCA and a FCNN. These three methods will be discussed in more detail in section 4.

2. RELATED WORK

Exoplanet detection using machine learning has been tackled by a variety of approaches that can be grouped into: 1) Classical Machine Learning Solutions 2) Deep Learning Solutions. Sturrock 2019 [1] made a comparison between three classical ML models: SVM, K-NN and Random Forests in detecting exoplanets. Among the three, they concluded that Random forests had the best results in terms of robustness to high dimensional data and prediction performance. Yip and Nikolaou [2] used a Generative Adversarial Network to create a synthetic direct imaging dataset and then used ConvNet’s to discover exoplanets. Their model performed well (>97% accuracy) for inputs that had low signal-to-noise measurements. Another approach was adopted by Shallue and Vanderburg [3], where they used a ConvNet to detect exoplanets in the NASA Exoplanet Archive: Autovetter Planet Candidate Catalog dataset. Their model had achieved an accuracy of 98.8%.

3. DATASET AND FEATURES

3.1. Data

To train and test our networks, we used the "Exoplanet Hunting in Deep Space" dataset from Kaggle [4]. The training set contains 5050 non-exoplanet samples and 37 exoplanet samples while the test set contains 565 non-exoplanet samples and 5 exoplanet samples. Each sample is a light curve measured from a different star consisting of 3197 time-series light intensity values, $x \in \mathbf{R}^{3197}$. An example of a light curve is shown in Fig. 1.

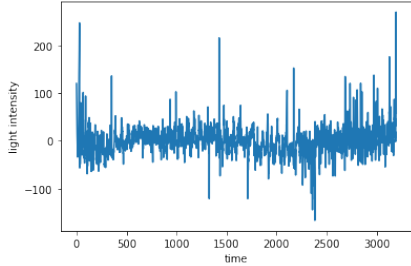


Fig. 1: A raw sample of a light curve from a star containing an exoplanet. It is noisy and there is no obvious indication that this star has an exoplanet.

3.2. Training with an Imbalanced Dataset

Exoplanet samples are clearly underrepresented in the dataset. If we trained our models on the dataset as is, the models would see very few exoplanet examples and have trouble learning the difference between exoplanet and non-exoplanet light curves. In order to combat this under representation of exoplanet samples, we used Synthetic Minority Oversampling Technique (SMOTE).

SMOTE generates synthetic minority class samples by linearly combining a sample with some of its nearest neighbors [5]. For example, with a minority class sample, x_1 , its nearest neighbor, x_2 , and a random number between 0 and 1, α , a synthetic sample can be generated as follows.

$$x_{new} = x_1 + \alpha(x_2 - x_1) \quad (1)$$

We used the SMOTE function from the python imblearn library [6]. By using SMOTE, we were able to balance the training data, resulting in 5050 non-exoplanet samples and 5050 exoplanet samples.

3.3. An Explanation of Principal Component Analysis

Principal Component Analysis (PCA) is a method that is often used for feature extraction and dimension reduction of high-dimensional data. At a high level, PCA works by mapping the data into a new feature space such that the components of this feature space maximize the variance of the data. These components are what we call principal components, and they can also be represented by the unit eigenvectors of the covariance matrix. The eigenvalues correspond to the magnitude of variance explained by each eigenvector, so the eigenvector with the largest eigenvalue will be the first principal component, the eigenvector with the second largest eigenvalue will be the second principal component, and so on. Since the covariance matrix is symmetric, it has orthogonal eigenvectors. The orthogonal nature of the principal components de-correlates the data and is what gives this method the ability to extract so much data variance in so few features. The number of principal components was selected such that 100 percent of the

variance of the original dataset is accounted for. PCA was performed using the Scikit-learn python library [7].

3.4. Fast Fourier Transform

Since our data consists of time varying signals, one form of feature extraction we wanted to try is the Discrete Fourier Transform (DFT), which transforms a signal into frequency space. A DFT transforms a signal into a finite sum of sines and cosines by computing the coefficient, f_k , for each frequency, k .

$$F_n = \sum_{k=0}^{3197-1} f_k e^{-2\pi i n k / 3197} \quad (2)$$

To perform DFT we used the FFT algorithm implemented in Numpy [8]. The input to the FFT is a vector $x \in \mathbf{R}^{3197}$ and the output is a vector $z \in \mathbf{C}^{3197}$. We then take the absolute value of each of the complex elements of z to get the magnitude for each frequency, $z \in \mathbf{R}^{3197}$. An example of a normalized light curve after FFT has been performed is shown in Fig. 2. Because there is no longer a relationship between neighboring inputs after the FFT, this feature extraction method will only be used with the FCNN model.

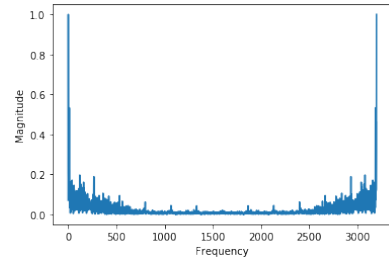


Fig. 2: An example of a light curve signal after FFT has been performed.

4. MACHINE LEARNING METHODS

4.1. Convolution Neural Networks

Convolution Neural Networks are a class of deep neural networks that are most commonly applied to visual imagery for classification purposes because of their state-of-the-art performance. At its core, CNN's are built using three main types of layers: The Convolution Layer, Pooling Layer, and Fully-Connected Layer. The architecture of CNN's follow that first the raw data (normally an image) is passed through the convolution layers where each layer contains a stack of filters; each filter performs in a sliding window manner a dot product between its weights and a portion of the input to which these weights are locally connected to. This operation is known as convolution. These layers act as automatic feature extractors, i.e., the weights associated with each filter are automatically learnt during the training process. After convolution, the next

step involves in applying an activation function to the data and then the data is downsampled in the pooling layer. Lastly, the processed data is flattened and passed through the fully-connected layers for classification.

Our project required the classification between exoplanets and non-exoplanets in a time-series dataset. Since CNNs consider the spatial relationship between the neighbouring data values, one-dimensional CNNs were used. The data was standardized and 30 principal components were used as the input $\in \mathbf{R}^{30}$ to our CNN model. The CNN model consisted of two 1-D convolution (3x3 64 and 128 filters) + 2x2 maxpooling layers stacked together followed by 4 Fully-Connected Layers (100, 50, 25 and 1 neurons). Binary Cross-Entropy was used as the loss function and ADAM as the optimizer using keras [9]:

$$L(x, y) = -\frac{1}{N} \sum_{i=0}^N (y \log(\bar{y}_i) + (1 - y) \log(1 - \bar{y}_i)) \quad (3)$$

Where, $y \in \mathbf{R}$ is the true value and $\bar{y}_i \in \mathbf{R}$ is the predicted value.

4.2. Convolutional Autoencoder

A convolutional autoencoder is a convolutional neural network that aims to learn features of a dataset in an unsupervised manner. There are various types of autoencoders, such as a variational autoencoder, but the one implemented in this project is what some call a vanilla autoencoder. A convolutional autoencoder has two components: an encoder and a decoder. The encoder consists of the input layer and a series of convolutional layers that progressively decrease the number of feature maps extracted. The last layer consists of the encoded data, which is a lower dimensional representation of the input layer that is referred to as the latent space. The decoder takes the latent space as input, and progressively increases the number of feature maps, with the last layer being a single feature map with the same dimensions as the input layer. Padding of the convolutional layers is needed in order to ensure the output layer has the same dimensions as the input layer. The goal of the autoencoder is to recreate the input data as accurately as possible, which is measured by the mean-squared error cost function shown below.

$$MSE = \frac{1}{n} \sum_{n=1}^n (y_n - \bar{y}_n)^2 \quad (4)$$

Where y is the input, \bar{y} is the output/recreation, and n is the number of data points. In this project, the autoencoder is used for anomaly detection. The non-anomalous data consists of stars without exoplanets and the anomalous data consists of stars with exoplanets. The autoencoder is trained on the non-anomalous data so that it learns the important features necessary to recreate this type of data with a low MSE. When it has been trained and is asked to recreate anomalous data,

Model	Optimizer	Batch Size	Learning Rate	Dropout	Epoch
CNN	ADAM	150	0.003	0.4	100
Conv AE	ADAM	32	0.003	0.3	50
FCNN	ADAM	50	0.001	0.5	50

Table 1: Final (Hyper) parameters

the theory is that the autoencoder will struggle to recreate the input and generate a high MSE. A threshold is set such that inputs that generate a MSE outside the threshold are classified as anomalies. For this report, a distribution is generated from non-anomalous MSE values. When testing, any inputs that generate a MSE outside two standard deviations from the mean of the non-anomalous MSE distribution are classified as anomalies (i.e. a star with exoplanets).

4.3. FFT with FCNN

Since each data sample is a time varying signal, we wanted to try using the frequency space transformation of the signal as the input to a neural network model. Essentially, we used a FFT to extract a new set of features from the original data. As discussed in section 1.3, we then use PCA with 100 principle components to reduce the dimensions of the data to $z \in \mathbf{R}^{100}$ before inputting to the model.

Instead of a CNN, we chose this model to be a FCNN since the input data is no longer time series and therefore has no particular relationship between neighboring elements of the input. Each unit of a FCNN is connected to all of the units of the subsequent layer. Due to the non-linear activation functions, a FCNN is able to learn complicated functions to map the input to the output. Our FCNN model takes as input a vector, $z \in \mathbf{R}^{100}$ and has two hidden layers, one with 30 units followed by one with 10 units, each with ReLU activation functions. The model has one output unit which uses a sigmoid activation function because the model is being used for binary classification. Dropout at 50% is used on the first and second layers to help regularize the model and reduce overfitting. The model was trained with the binary cross entropy loss function and the ADAM optimizer.

5. DISCUSSION AND RESULTS

5.1. Hyper Parameters

Table 1 summarizes the final hyperparameters that were used for the models after extensive tuning and testing:

5.2. Model Results

Table 2 summarizes the results for the three models against three evaluation metrics: Precision/Recall/Accuracy

CNN: The results of the CNN model using 30 Principal Components was satisfactory. In Fig.3(a), the model was

Model	Precision %	Recall %	Accuracy %
CNN	27.27	60	98.24
Conv AE	0	0	96.67
FCNN	19.04	66.66	96.84

Table 2: Performance measure against evaluation metrics

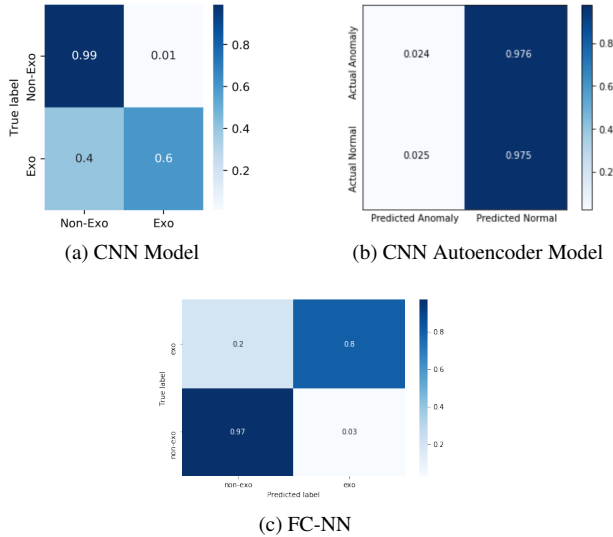


Fig. 3: Normalized Confusion matrices on test set

able to detect 60% (3/5) exoplanets in the test set, and non-exoplanets with a very high accuracy of 99% (557/565). A lot of tuning was carried out to make the model detect greater than three exoplanets, however, this CNN model achieved its maximum performance at 3/5 exoplanets.

CNN Autoencoder: The results of the CNN autoencoder were unsatisfactory as the encoder failed to learn features that differentiated the non-anomalous (no exoplanets) data from the anomalous data (exoplanets). When testing, as seen in Figure 4, the anomalous and non-anomalous data were recreated with practically the same MSE.

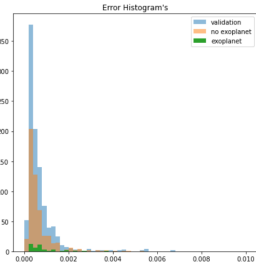


Fig. 4: Distribution of MSE errors generated during testing.

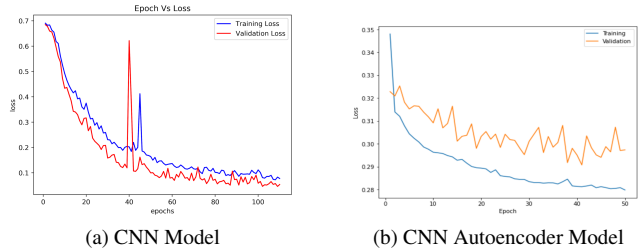
Practically every input, whether it was non-anomalous or anomalous, was classified as non-anomalous. This can be seen in the confusion matrix shown in Figure 3(b).

FFT with FCNN: The results for the FCNN that used FFT as feature extraction were fairly promising. As shown

in Fig. 3(c), four out of the five exoplanet samples (or 80%) were correctly identified. As well, 97% of the non-exoplanet samples were correctly identified. These results suggest that perhaps the frequency space representation of the data reveals patterns that otherwise might not be noticed by a neural network model.

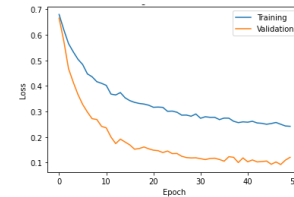
Comparing the two supervised machine learning methods shows that there may be a trade-off. The FCNN was able to correctly identify four out of five exoplanets while the deep CNN was only able to identify three out of five. On the other hand, the FCNN model only identified 97% of non-exoplanets compared to the 99% correctly identified by the deep CNN model. So the FCNN results in more true positives but at the cost of having more false positives while the deep CNN has less true positives but also less false positives.

Determining which model is "better" depends on the application. For exoplanet hunting, it is probably better to have more true positives at the cost of having more false positives. There are other exoplanet finding methods that can be used to help rule out false positives but one would not want an exoplanet being missed because it was a false negative.



(a) CNN Model

(b) CNN Autoencoder Model



(c) FC-NN

Fig. 5: Training loss plots.

6. CONCLUSION

Had we had more time, we would have measured Recurrent Neural Network's performance on this dataset. In summary, the results show that supervised learning prevailed over unsupervised learning in this application. We also learned the importance of data pre-processing in deep learning, as standardization and feature extraction dramatically boosted results in the supervised learning cases. Also, without SMOTE, the results achieved with the supervised cases would not have been achievable as the imbalance of data between the two classes was far too great. As space exploration progresses and we are able to extract more data, machine learning will be able to have an even greater impact in teaching us about our universe.

7. REFERENCES

- [1] George Clayton Sturrock, Brychan Manry, and Sohail Rafiqi. Machine learning pipeline for exoplanet classification. *SMU Data Science Review*, 2(1), 2019.
- [2] Kai Hou Yip, Nikolaos Nikolaou, Piero Coronica, Angelos Tsiaras, Billy Edwards, Quentin Changeat, Mario Morvan, Beth Biller, Sasha Hinkley, Jeffrey Salmond, Matthew Archer, Paul Sumption, Elodie Choquet, Remi Soummer, Laurent Pueyo, and Ingo P. Waldmann. Pushing the limits of exoplanet discovery via direct imaging with deep learning. *arXiv:1904.06155 [astro-ph.EP]*, 2019.
- [3] Christopher J. Shallue and Andrew Vanderburg. Identifying exoplanets with deep learning: A five planet resonant chain around kepler-80 and an eighth planet around kepler-90. *Astronomical Journal*, 155(94), 2018.
- [4] Exoplanet hunting in deep space. <https://www.kaggle.com/keplersmachines/kepler-labelled-time-series-data>, 2017.
- [5] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Phillip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16(321 – 357), 2002.
- [6] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017.
- [7] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011.
- [8] S. van der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science Engineering*, 13(2):22–30, 2011.
- [9] François Chollet et al. Keras. <https://keras.io>, 2015.

Individual Contributions

Joseph

CNN Autoencoder and testing of hyperparameters, data pre-processing, and one-hot encoding to try improve CNN model results.

James

Feature extraction with Fast Fourier Transform, tuning and testing of Fully Connected Neural Network

Maria

Applying PCA, SMOTE, data pre-processing, tuning and testing the CNN model

Response to Critical Reviews (only 2 reviews received)

Critique of **group 22** presentation - **Exoplanet Hunting in Deep Space**

Critiques by **group 6**.

Group 22 gives a very comprehensive discussion on detecting exoplanets. The whole process is well-explained from acquiring the data to network structuring, the confusion matrices were also pretty helpful in quickly determining the success of the models.

Some questions/suggestions:

- Since you are comparing three very different structures here, we would like to hear more about the reasoning why you choose them and why you think they intuitively or analytically make sense, which includes:
 - Autoencoder seems to only reduce the amount and precision of the information. What is your reasoning why an autoencoder could probably help in extracting the useful features?
We decided to implement the autoencoder as the problem of detecting stars with exoplanets can be seen as an anomaly detection problem. We had recently done an anomaly detector in our homework assignment using an autoencoder and wanted to see how well it would perform in this case. However, 1D convolutions didn't seem to work too well at extracting features and if we had more time we would have liked to incorporate LSTM.
 - FFT + FCN gives the best results. However it is not very clear whether the improvement is due to the FFT or FCN. It would be helpful in answering this question to compare it with the performance of Time series data + FCN.
Agreed, this would be a good comparison. We tested it early on but did not record the results. Since we are already comparing three models, we do not want to expand the scope of the report too much.
 - It seems that the most important goal is to distinguish exoplanets and higher false positive rates (detecting non-exo as exo) seems acceptable. Considering this fact, what change do you think that can be made to raise the true positive rate, even if it may sacrifice others?
Excellent point. We address this topic in our report when discussing how to judge which model is "better". Oversampling with SMOTE is one way we tried to increase the exoplanet true positives. If our models saw very few exoplanet samples, then they would be more likely to think everything is a non-exoplanet. One way perhaps to get more true positive exoplanets might be to purposely imbalance the dataset in favor of exoplanets or to undersample the non-exoplanet samples.
- Though you talked about SMOTE briefly, it would be helpful to see how the extra data samples were generated in the slides.
 - Good point. A more detailed explanation is provided in the report along with an equation showing an example of how samples are generated
- Do you have an idea why there was a sharp jump in training and validation loss?
This could be due to the random nature of dropout which is used to regularize the models. With dropout, each epoch is essentially training a different network so spikes could occur.

- A table summarizing your results with the different models would be helpful for a quick comparison of performance. Would be a good addition in your Results/Conclusions section.

We agree! We are working on a table for the final report that compares different aspects of the models.

Critique of **group 22** presentation - **Exoplanet Hunting in Deep Space** Critiques by **group 44**.

This project focuses on detecting whether a planet is an exoplanet given time-series time-series light intensity/flux observations. Group 22 uses 3 models to detect: PCA with CNN, PCA with convolutional auto-encoder and FFT, PCA with a FC network. Also they use the SMOTE technique to overcome the training data imbalance problem.

Possible improvements:

- Since it's an imbalance dataset and there're only 5 anomaly in validation set, I strongly suggest using precision and recall or F1 score which combines those 2 together during the training (instead of accuracy). We don't care about how many normal planets that are detected, instead we should focus more on what fraction of exoplanets are detected.

Good point! We are going to implement precision and recall scores into our final report.

- Since it's a time-series dataset, using CNN might not be able to catch the temporal feature. Maybe it's why the anomaly detection is lower for CNN model. You may try Time-CNN or LSTM.

We agree, there definitely were issues handling the temporal features for the autoencoder. We had actually found an implementation of an LSTM autoencoder that we wanted to try for this project. However, we spent too much time playing with hyperparameters and different data pre-processing methods and ended up not having enough time to implement a fourth model. In hindsight, we should have just accepted the vanilla autoencoder wasn't going to work and tried incorporating LSTM.

- Since you mention SMOTE sampling, it would be better to provide a comparison

between training using original dataset and training using SMOTE sampling. It might be able to demonstrate that SMOTE is a crucial technique for the training.

Yes we think this would be a good idea for the presentation/report. We did train on the data without SMOTE, but the results were terrible so we didn't bother to record it. However, providing the results from training without SMOTE would have been helpful to emphasize the impact of SMOTE as opposed to us just saying that SMOTE helped.