Machine Learning Engineer Nanodegree

Capstone Project

Suzanne Taylor
July 10, 2018

I. Definition

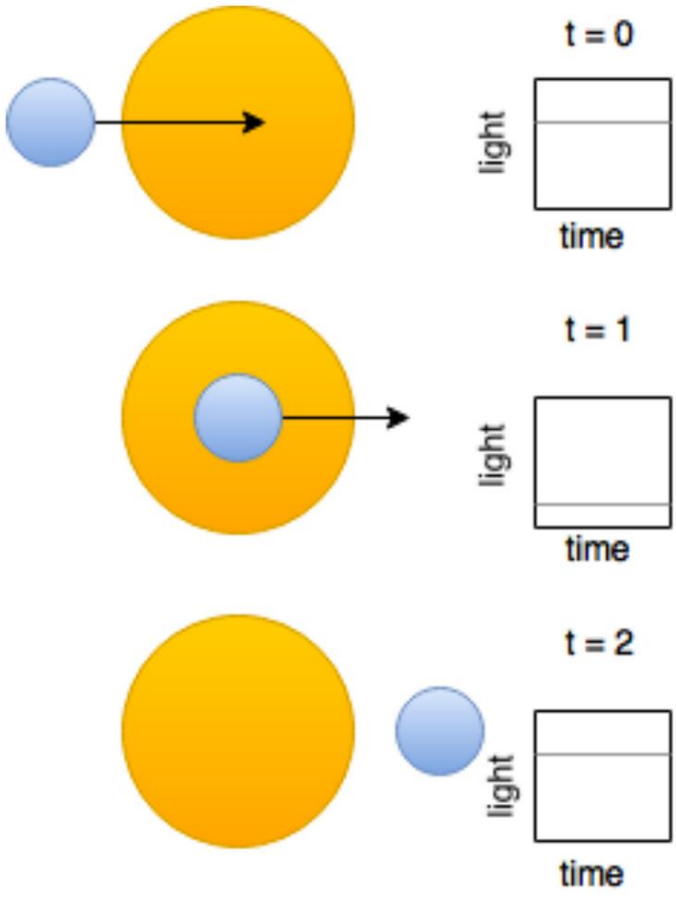
Project Overview

Kepler is a NASA built telescope launched in 2009 for the purpose of detecting planets orbiting other stars. Designed to survey a portion of the Milky Way to discover Earth-size exoplanets in or near habitable zones of those stars, Kepler's sole scientific instrument is a photometer that continually monitors the brightness of approx. 150,000 main sequence stars in a fixed field of view. This data is transmitted to Earth and the light curves resulting from the observations is used to determine if a planet has transited the star. As of 2 June 2018, there are 3,786 confirmed planets in 2,834 systems, with 629 systems having more than one planet.[1]

Due to the large amount of data relying on human judgment to produce a planet candidate is a time-consuming process and so this would be a good candidate for machine learning. As noted by Shallue et al. (2017)[2] there have been a number of projects to automate the process of detecting planets including Robovetter and Autovetter. In 2017 Shallue et al.[2] described a method of using a deep neural network to automatically vet Kepler threshold crossing events. This model has a good accuracy for distinguishing between transiting planets and other false positives.

Problem Statement

Planets do not emit light but when a planet transits a star the light intensity (flux) of the star dims. If a star is observed over several months or years a pattern of regular dimming may occur which could indicate a planet is orbiting the star.



[3]

As shown above at time t0 before the planet (blue) starts it's transit the light intensity is high. At time t1 the planet is halfway across the star and the light intensity drops. At time t2 the planet has completed its transit and the light intensity has returned to normal.

The project will build a CNN and train it on a provided dataset of flux values for 5087 stars. Each star in the training set has been labeled to indicate whether the star has exoplanets or not. Once the model has been trained it will then be tested on a separate dataset to attempt to predict if any of the stars in the test dataset have exoplanets.

Metrics

The metrics to evaluate the model on are recall, precision, and F_1 score. F_1 score is a good measure of the test's accuracy. Accuracy would not be a good metric as the dataset is highly imbalanced with none exoplanet stars.

TP: true positive

TN: true negative

FP: false positive

FN: false negative

Recall = TP / (TP + FN)

Precision = TP / (TP + FP)

	Predicted		
Actual		Non-exoplanet	Exoplanet
	Non-exoplanet	TN	FP
	Exoplanet	FN	TP

Precision is the number of correct results out of the total number of results. Recall is the number of correct results divided by the number of correct results that should have been returned.

F_1 = 2 x ((Recall x Precision) / (Recall + Precision))

II. Analysis

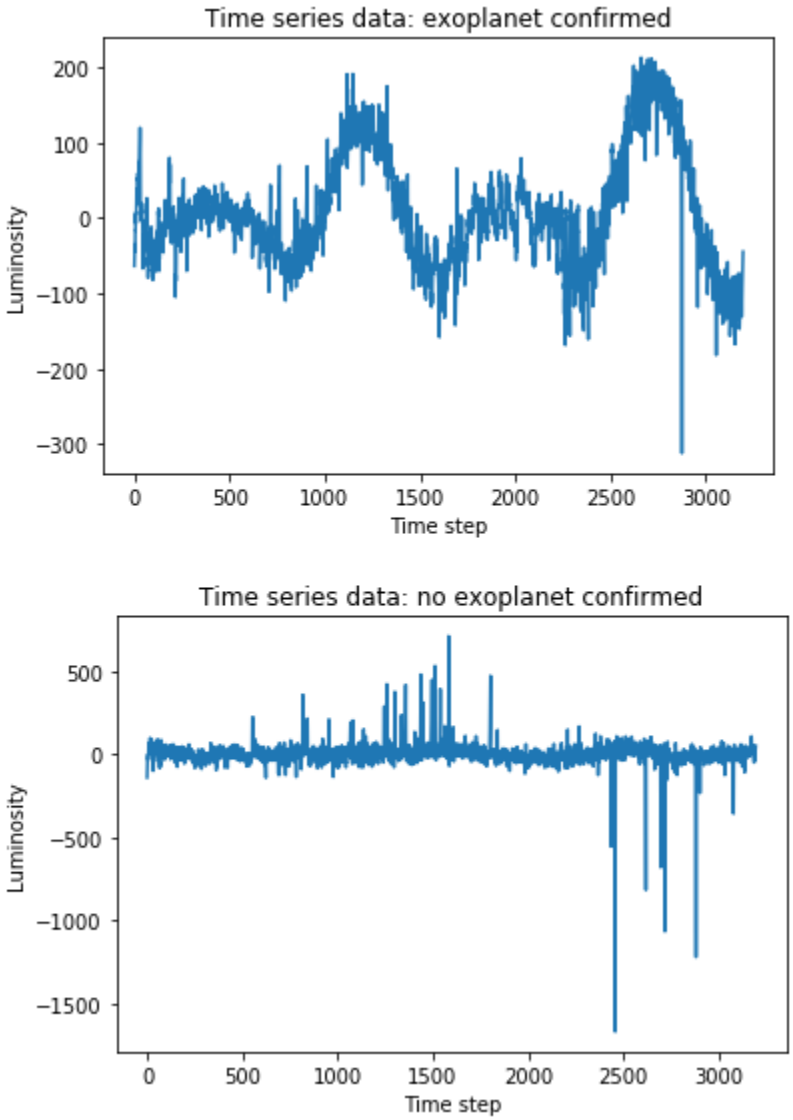
Data Exploration

The dataset was obtained from Exoplanet Hunting in Deep Space on Kaggle.com [3]. The data has been cleaned and is derived from observations made by the NASA Kepler space telescope. 99% of the data originates from campaign 3 and the dataset was prepared in late summer 2016

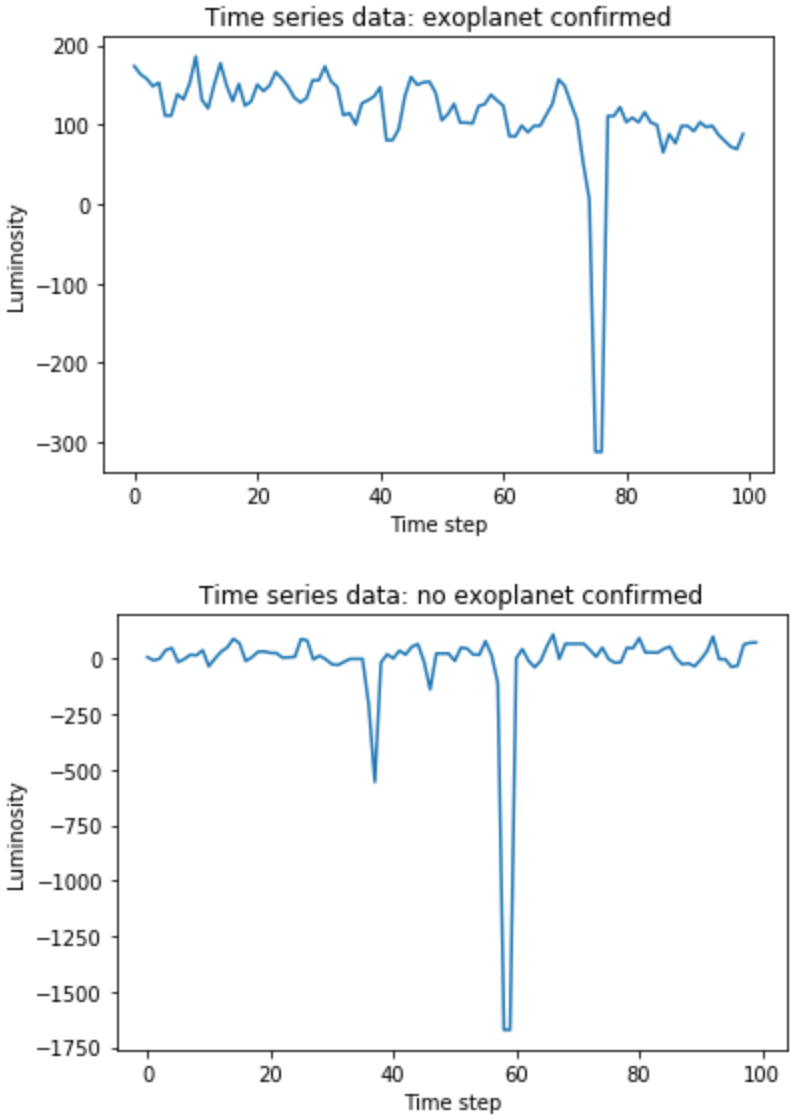
The data describes the change in light intensity (flux) of stars. Each start has been labeled 1 or 2 with 2 indicating the star is confirmed to have at least one exoplanet. The data is in 2 sets one for training and one for testing. The training set contains 5087 rows of observations (stars) with 37 confirmed exoplanet stars and 5050 non-exoplanet stars. The test set contains 570 rows of observations with 5 confirmed exoplanet stars and 565 non-exoplanet stars. Each row has 3198 features. Column 1 is the label indicating whether the star has exoplanets or not. Column 2 - 3198 are the flux values of the star.

The both datasets are heavily imbalanced with the training set having 0.73% of the records as confirmed exoplanets and the test dataset having 0.88% of the records as confirmed exoplanets.

Exploratory Visualization



In the first example (confirmed exoplanet) we have a star with variation in intensity; the second (no planet) shows why this is such a difficult task – a drop in luminosity can be caused by other phenomena. Taking a closer look:



The top figure appears to show a transit at around x=2875. However, a sharp dip in brightness alone does not indicate a transit - as shown in Figure 2. The network will have to look for subtle features.

Algorithms and Techniques

Since we are dealing with only 2 types of results, confirmed exoplanets and non-exoplanets, the data has been loaded the labels on the data will be changed from 1 to 0 for non-exoplanets and 2 to 1 for confirmed exoplanets this will help with the binary classification of the data. Both datasets will then be normalized.

The training set will be randomly split into two pieces 80% for training and 20% for cross validation. Because the datasets are heavily imbalanced the training data will have SMOTE applied to it to increase the number of confirmed exoplanets to a 50-50 mix.

The research performed by Shallue et al. (2017)[2] used a 10-layer CNN and the model created is based on the parameters specified in the paper. As in the paper the filer sizes are 16, 32, 64, 128, and 256 and the ReLU activation function is used. The output layer uses a sigmoid function.

Training used the Adam method with $\alpha = 10^{-6}$, $\beta 1 = 0.9$, $\beta 2 = 0.999$, and $\varepsilon = 10^{-8}$. The α value was chosen to slow down the learning rate to prevent the gpu learning too fast and not obtaining good results.

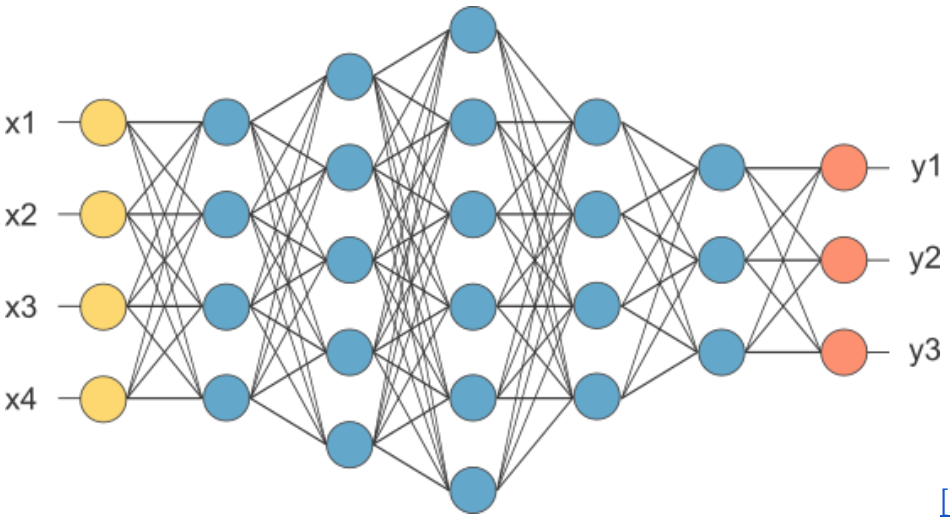
The original paper used batches of 64 over 50 epoch and I have kept the batch size the same but increased the epochs to 100 as this seems to result in a better outcome.

A convolutional neural network (CNN) is a class of deep feed-forward artificial neural networks which were inspired by biological processes. The connectivity patterns between neurons resembles the organization of the animal cortex. CNNs require relatively little pre-processing compared to other image classification algorithms which means the network learns to filter the data whereas traditionally algorithms would need the filters hand engineered.

A CNN is built from convolution layers, pool layers, and fully connected layers. The convolution layer is the core of the CNN. It consists of a set of learnable filters which have a small receptive field but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input. Stacking the activation maps for all filters along the depth dimension forms the full output volume of the convolution layer. Every entry in the output volume can be interpreted as an output of a neuron that looks at a small region in the input and shares parameters with neurons in the same activation map. The convolution layers extract features from the input and preserves the relationship be points by learning features using small squares of input. The input matrix is multiplied by a filter matrix to produce a feature map. As the layer calculates the values for the feature map it moves the input matrix across the row in a stride. Stride is the number of pixels the input matrix is shifted over. [6]

The purpose of the activation function is to introduce non-linearity into the CNN. I will use ReLU it is known to have better performance than tanh and sigmoid. Rectified Linear Units (ReLU) is a layer which applies the non-saturating activation function $f(x) = \max(0, x)$ which increases the nonlinear properties of the decision function without affecting the receptive fields of the convolution layer. [6] Other functions which increase non-linearity are tanh and sigmoid. ReLU is often preferred because it trains the neural network faster without a significant penalty to generalization accuracy.

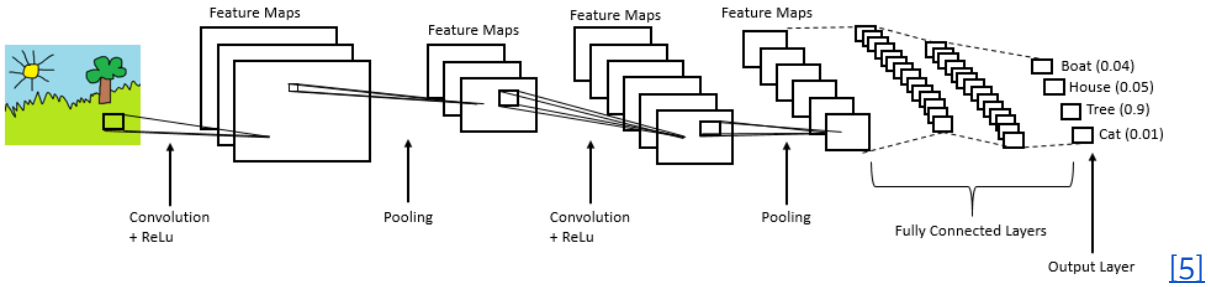
Pooling layers are a form of non-linear down sampling the most popular of which is max pooling. It partitions the input into a set of non-overlapping rectangles and outputs the maximum value for the rectangle. The pooling layer progressively reduces the size of the representation to reduce the number of parameters and hence also controls overfitting. Once we have passed the data through the convolution layers we flatten the data into a vector and feed it into the fully connected layers.



[5]

After pooling layer, flattened as FC layer

The final layer we use is the output layer with an activation function such a SoftMax or sigmoid. This final layer is used to classify the output.



[5]

The above image illustrates the layers in a CNN

A CNN is a good candidate for this problem as it is known to work well with very little pre-processing. This dataset is complex because a star does not produce a constant light intensity over time. The transit of a planet will lower the intensity of the of the star but other events such as eclipsing binary stars will lower the intensity of the observations. Noise in the signal might also change the intensity of the star producing readings which look like transits but are not.

Benchmark

My approach is based in the Mystery Planet (99.8% CNN) kernel by Peter Grenholm on Kaggle[4]. This approach achieved a 99.8% accuracy and my desired result will be to obtain accuracy close to that.

III. Methodology

Data Preprocessing

In the original data came from a larger NASA dataset which contained more labels than those used in the training and test datasets. The label in the provided datasets are 1 for non-exoplanets and 2 for confirmed exoplanets, these labels will be changed to 0 and 1 respectively.

As noted in the paper feature normalization is important for a CNN to function properly so we will normalize both datasets.

Both datasets are heavily imbalance, the test dataset will not be touched but the training set will be augmented to it to increase the ratio of confirmed exoplanets to 5050.

The training set will be randomly split into 2 pieces 20-80. 20% of the records will be used for cross validation and 80% will be used for training. This training set will have SMOTE applied to it to increase the number of confirmed exoplanets to a 50-50 mix.

Implementation

The language used for this model is Python 3.6 and uses the keras, matplotlib, pandas, NumPy, imbalanced-learn, scikit-learn packages. Keras is a high-level neural networks API capable of running on top of TensorFlow, CNTK, or Theano. Scikit-learn is a simple and efficient tool for data mining and data analysis. Imbalanced-learn is an API with a variety of methods for oversampling a dataset. Matplotlib is a 2D plotting library used to produce graphs. Pandas is a library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. NumPy is used to handle the n-dimensional dataset. Python was used due to its available scientific packages and Jupyter notebook was used to develop the model due to the ease the code could be produced.

The model will be a CNN built with a Sequential model using CONV1D with filters of 16, 32, 64, 128, and 256 a kernel of 11 and using the ReLU activation function. The input shape will be the size of the dataset. It will have 2 sections split by a maxpooling layer with strides of 8 batch normalization. These 2 sections will be followed by a maxpooling layer and then flattened before going through 4 densely connected layers and finally the output layer using the sigmoid function

The optimizer used for training will be Adam with $\alpha = 10^{-6}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$

This is the final output shape of the CNN

```
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_71 (Conv1D)           (None, 3187, 16)          192
_________________________________________________________________
conv1d_72 (Conv1D)           (None, 3177, 32)          5664
_________________________________________________________________
conv1d_73 (Conv1D)           (None, 3167, 64)          22592
_________________________________________________________________
conv1d_74 (Conv1D)           (None, 3157, 128)         90240
_________________________________________________________________
conv1d_75 (Conv1D)           (None, 3147, 256)         360704
_________________________________________________________________
max_pooling1d_15 (MaxPooling (None, 394, 256)          0
_________________________________________________________________
batch_normalization_8 (Batch (None, 394, 256)          1024
_________________________________________________________________
```

```
conv1d_76 (Conv1D)          (None, 384, 256)         721152
_________________________________________________________________
conv1d_77 (Conv1D)          (None, 374, 128)         360576
_________________________________________________________________
conv1d_78 (Conv1D)          (None, 364, 64)          90176
_________________________________________________________________
conv1d_79 (Conv1D)          (None, 354, 32)          22560
_________________________________________________________________
conv1d_80 (Conv1D)          (None, 344, 16)          5648
_________________________________________________________________
max_pooling1d_16 (MaxPooling (None, 43, 16)           0
_________________________________________________________________
flatten_8 (Flatten)          (None, 688)              0
_________________________________________________________________
dropout_8 (Dropout)          (None, 688)              0
_________________________________________________________________
dense_36 (Dense)             (None, 512)              352768
_________________________________________________________________
dense_37 (Dense)             (None, 512)              262656
_________________________________________________________________
dense_38 (Dense)             (None, 512)              262656
_________________________________________________________________
dense_39 (Dense)             (None, 512)              262656
_________________________________________________________________
dense_40 (Dense)             (None, 1)                513
=================================================
Total params: 2,821,777
Trainable params: 2,821,265
Non-trainable params: 512
_________________________________________________________________
```

Since the dataset was pre-cleaned by NASA there were no problems in implementing the model.

Refinement

The original paper suggested a kernel of 3 or 5, a stride of 2, 50 epochs and $\alpha = 10^{-5}$. After many trials and comparisons of result sets the final parameters I settled on were a kernel = 11, stride = 4. Since the gpu appeared to be learning too fast and remaining dumb I reduced $\alpha = 10^{-6}$ and increased the epoch to 100, This slowed down the learning rate and increased the number of training sessions which produced more desirable results from the test set.

IV. Results

Model Evaluation and Validation

Training validation set performance
AUPRC: 0.60654756746017

Train Set Error 0.02062868369351667

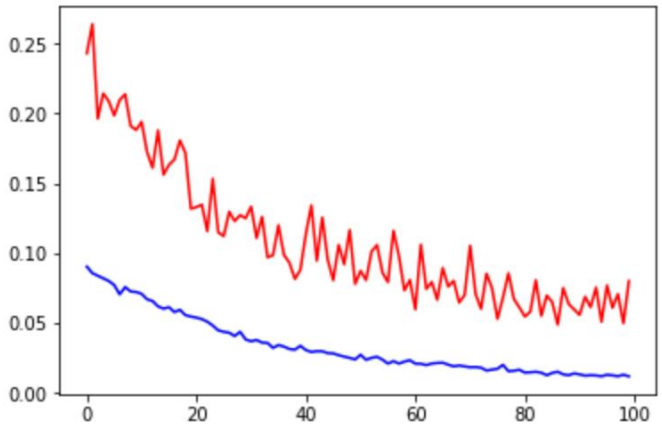
```
------------
Precision - Train Set 0.20833333333333334
------------
Recall - Train Set 0.7142857142857143
------------
Confusion Matrix - Train Set
[[992  19]
 [  2   5]]
------------

------------
classification_report_train
             precision    recall  f1-score   support

          0       1.00      0.98      0.99      1011
          1       0.21      0.71      0.32         7

avg / total       0.99      0.98      0.98      1018
```

First, we look at the performance of the training validation set and as we can see confusion matrix shows 19 false positives and 2 false negatives. The recall is 71.4 % and precision is 20.8%. If we look more closely we find the model is very good at classifying the non-exoplanets with recall at 98% and precision at 100% but the exoplanets have a recall and precision of 71% and 21% respectively. The F_1 score for exoplanets is 32%, which is quite low, indicating the model does not do very well predicting exoplanets.

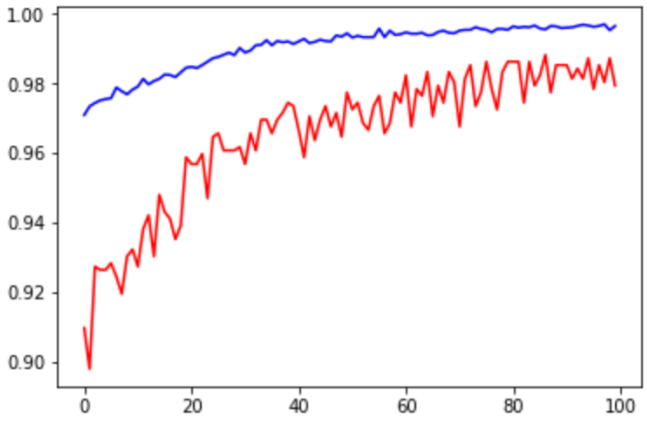
Test set performance

```
AUPRC: 0.9183333333333332

Test Set Error 0.007017543859649145
------------
Precision - Test Set 0.5555555555555556
------------
Recall - Test Set 1.0
------------
Confusion Matrix - Test Set
[[561   4]
 [  0   5]]
------------

------------
classification_report_test
             precision    recall  f1-score   support

          0       1.00      0.99      1.00       565
          1       0.56      1.00      0.71         5

avg / total       1.00      0.99      0.99       570
```

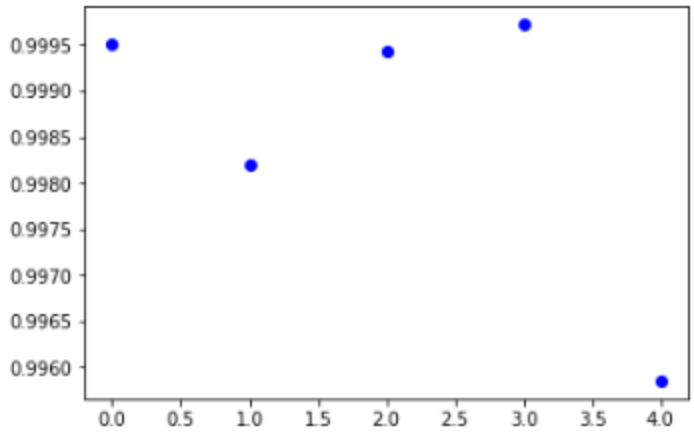
The second set of results we look at the performance of the test set and as we can see confusion matrix shows 4 false positives and no false negatives. The recall is 100 % and precision is 55.5%. If we look more closely we find the model is very good at classifying the non-exoplanets with recall at 99% and precision at 100% but the exoplanets have a recall and precision of 100% and 56% respectively. The F_1 score for exoplanets is 71%, which is higher than the training validation set, indicating the model does a decent job predicting exoplanets.

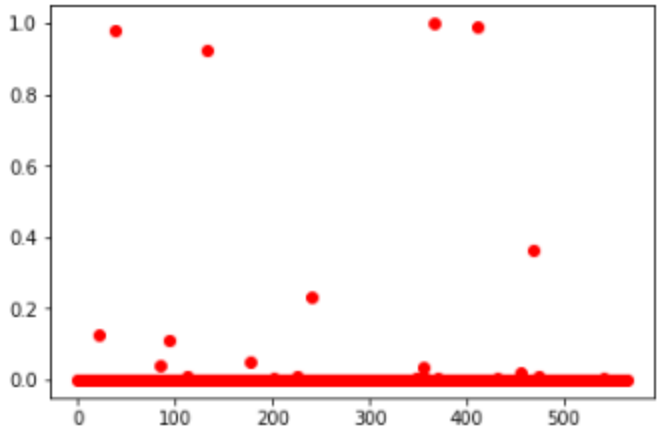


These above graph shows the convergence of the loss (blue) and the validation loss (red)

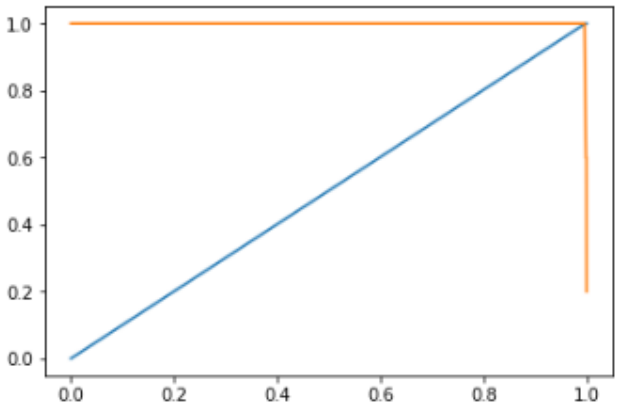


The above graph shows the convergence of the accuracy (blue) and the validation accuracy (red)

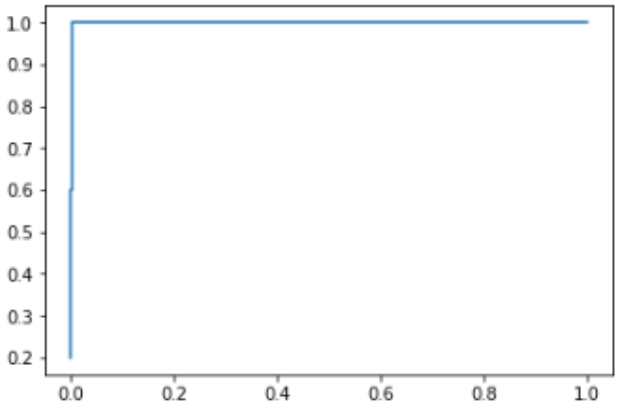




These graphs show that the five positive (blue) examples all get 0.996-1.00 score. Also, almost all negative (red) examples get score close to zero, except a few in the 0.9-1.0 range.




```
Crossover at 1.00 with specificity 1.00
```

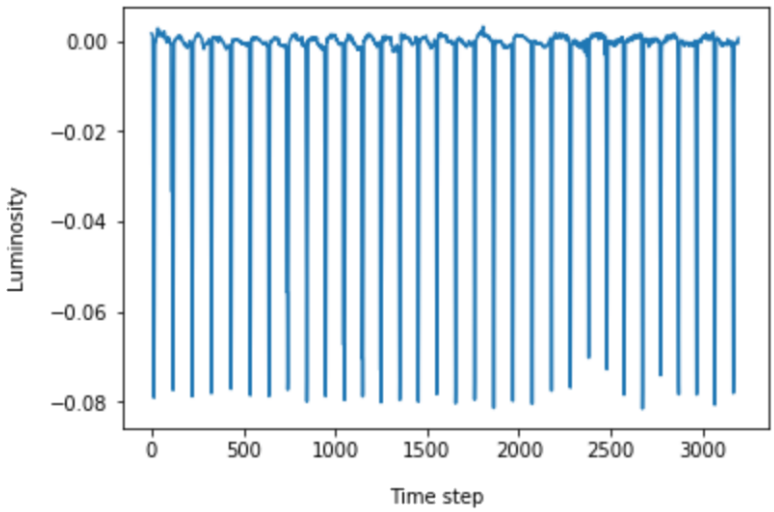



```
ROC area under curve is 1.00
```

The above graph shows us the optimal cutoff value for false positives.

Based on this value we find of the 4 false positives there is only one false positive that needs to be examined. The following graph is the normalized data for that point.

327

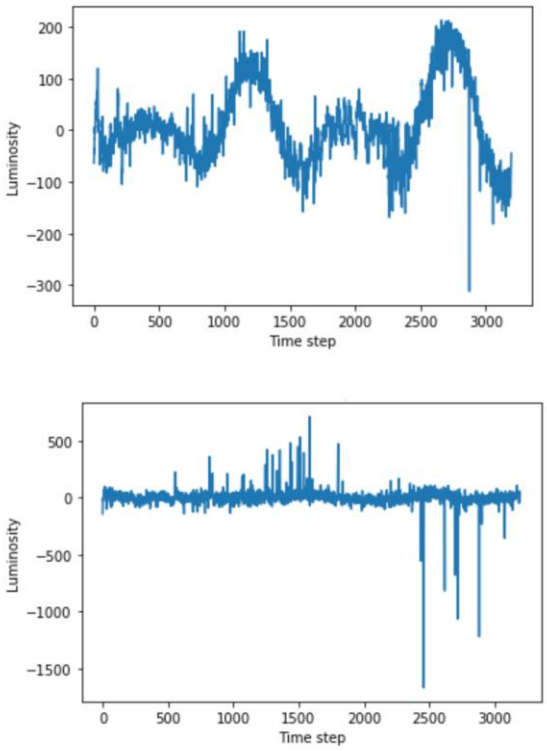


Justification

The chosen model was able to obtain similar results to the benchmark. The benchmark was 99.8% and f_1 score for this model was 99%. This would indicate the final solution is able to solve the problem. The model does not appear to be sensitive to small changes in the data or outliers in the data set. It appears to handle unseen data well as shown by the results of the test dataset.

V. Conclusion

Free-Form Visualization



The above graphs show the time series flux data for 2 separate stars. The model is able to take that data and determine if a planet is in orbit around that star.

Reflection

The process to analyze this dataset involved normalizing both the test and training sets to better work with a CNN. The training set was split into an 80-20 mix for training and cross validation and the training set was augmented with SMOTE to reduce the imbalance. After many test runs and tweaking of parameters the final shape of the model was obtained.

It was interesting to see that changing the configuration of the layers, kernel size, stride, and learning rate in the model produced different results. If a kernel =5, stride=2, and learning rate of 10^{-5} was used the model was only able to positively identify 3 exoplanets.

It was also interesting to see that changing the learning rate affected the ability of the model to predict exoplanets.

The final solution model does solve the problem but might be considered overly complicated for the problem.

Improvement

Flux readings are actually a wave pattern of intensity over time and an improvement to the solution would be to pre-process the data with Fourier and Gaussian transformations to reduce some of the noise in the signal.

References

1. Kepler (spacecraft) https://en.wikipedia.org/wiki/Kepler_(spacecraft)
2. Christopher J. Shallue & Andrew Vanderburg IDENTIFYING EXOPLANETS WITH DEEP LEARNING: A FIVE PLANET RESONANT CHAIN AROUND KEPLER-80 AND AN EIGHTH PLANET AROUND KEPLER-90 https://arxiv.org/pdf/1712.05044.pdf
3. https://www.kaggle.com/keplersmachines/kepler-labelled-time-series-data
4. https://www.kaggle.com/toregil/mystery-planet-99-8-cnn
5. https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148
6. https://en.wikipedia.org/wiki/Convolutional_neural_network#Convolutional_layer