DE MONTFORT UNIVERSITY LEICESTER

MSc Project Report

Title: Exoplanet Transit Detection using Deep Neural Networks

Author: Dinis Marques Firmino (P13240786)

Programme: Intelligent Systems MSc

Year: 2018

Section 1 of 7

FOT

FACULTY OF TECHNOLOGY

Exoplanet Transit Detection using Deep Neural Networks

Dinis Marques Firmino

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Master of Science (MSc)

of

De Montfort University.

De Montfort University

May 11, 2018

I, Dinis Marques Firmino, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

NASAs Kepler Space Telescope amongst other similar telescope surveys, was designed as a statistical mission to determine the presence of Earth-sized planets in orbit of stars in the Milky Way galaxy. Determining the presence of such planets requires vetting procedures that can correctly specify that a transit in a potential candidate solar system is indeed caused by an exoplanet, and not a false positive such as eclipsing binaries, stellar variability or instrumental artefacts. In this research, two deep learning architectures are presented as automated vetting methods for the accurate detection of exoplanets in the Kepler data. The final model configurations were able to correctly classify Kepler Threshold-Crossing Events (TCEs) $\approx 93.5\%$ of the time. In the future, automated vetting procedures like the methods proposed in this research will be commonplace in the Astronomy domain due to the large influx of data from new telescope surveys with the goal of observing celestial objects excluding exoplanets and phenomena in the universe.

Contents

Chapter 1

Introduction

Space exploration is concerted with the discovery and understanding of the many types of celestial objects and natural phenomena that exist in the universe with the ultimate goal of advancing our scientific understanding and interpretation of the laws of nature. The impact of space exploration on society, economy and politics is far-reaching [28] as the scientific and engineering advances required to perform such exploration can usually be applied to improve many aspects of daily life. There are two main forms of space study. One is carried out physically by Astronauts and robotic spacecraft, and the other by astronomers through the use of telescopes. The latter is the observational branch of Astronomy and is the problem domain field explored in this thesis.

Over the last decade, the volume of data that a single telescope survey can gather in a short period of time as steeply increased from $\approx$ 10GB to $\approx$ 90TB per day [20], pushing Astronomy into the big data domain. A few popular telescope surveys are/were the Very Large Telescope (VLT), Sloan Digital Survey (SDSS), Visible and Infra-red Telescope for Astronomy (VISTA), Kepler, K2, Large Synoptic Survey Telescope (LSST) and the recently launched Transiting Exoplanet Survey Satellite (TESS) [26]. Contrary to other surveys, NASA's Kepler, K2 and TESS space telescopes were assigned missions to specifically look for transiting exoplanets in orbit around the vast number of stars in the Milky Way. The Kepler and K2 telescopes alone have observed and processed photometric data on approximately 200,000 stars with high precision [19, 9] and using the Kepler pipeline correctly

classified approx. 4000 transiting planets [27, 17, 29], and dispositioned thousands more systems as potential candidates [33]. Candidates systems are defined by NASA as Threshold-Crossing Events (TCEs), that is, a periodic transit-like event that may or not be an exoplanet. These candidate TCEs are usually manually reviewed by seasoned experts to remove false positives caused by eclipsing binaries, or more commonly, stellar variability and instrumental artefacts.

Statistical learning techniques such as the deep learning architectures examined in this study, have been applied to help solve a vast array of problems in a wide array of domains, including Astronomy [30, 8, 25, 22]. The abundance of data from satellite surveys demands that there be automated analysis techniques to sift through these incoming data and reliably classify interesting phenomena and celestial objects. The goal is for these methods to one day eliminate the need for humans to perform manual and time consuming analysis of future data for accurate detection. The main aim of this research development project is to discover the effectiveness of deep learning architectures at modelling time series photometric data for the detection of exoplanets. A series of questions are proposed to help define the goal of the research, they are as follows:

- What is the future potential of the application of machine learning techniques to solve problems in the Astronomy domain?
- What are the most suitable deep learning architectures for modelling the Kepler data for detecting exoplanets?
- What is the importance of automation in this domain?
- How accurate are these models when compared to traditional techniques?
- What is the current state of the art in this field?
- Does the raw Kepler data require pre-processing, if so, what are the necessary techniques?

Chapter 2 provides a solid overview of the background of the thesis project by presenting a state of the art on the recent literature regarding the use of deep learning techniques in the field of astronomy. The aim was to examine and critically evaluate the variety of approaches already proposed in this domain in order

to obtain leads on potential answers to the research questions and to find a potentially novel approach that this thesis could contribute to the problem of exoplanet detection. Chapter 3 thoroughly explains the role of the Kepler pipeline and how data is retrieved, analysed and processed to ensure it is fit for modelling. Chapter 4 explains the procedure which selects the best configuration of hyper-parameters for the model architectures, and provides details on the structure of the selected topologies. Chapter 5 presents the results of the best performing model configurations and details the metrics and plots used to aid in articulating the performance of the final models in the main discussion. Chapter 6 provides an in depth analysis and critical review of the entire research development process. It aims to use the findings from the accomplished research objectives to answer the initially proposed research questions. Finally, Chapter 7 wraps up the dissertation with an honest conclusion.

Chapter 2

Literature Review

The literature review will be organized as follows. Firstly, the research papers will be grouped according to several criteria. These criteria are the machine learning approach used i.e model architecture, the problem domain within Astronomy e.g. classification of exoplanets, supernovae..., and the type of problem e.g. time-series, image recognition, ... Some papers are cited purely for helping describe the general background of the problem and to support the reason for undertaking this project, therefore, they will not be grouped in any criteria. The criteria table will draw a line between the differences in the papers and help determine if the content within is relevant to the research objectives.

2.1 Architectures

2.1.1 Fully Connected Neural Networks

Fully connected neural networks (a.k.a Multilayer Perceptron or feed-forward neural networks), are made up of a series of inter-connected layers made up of scalar units called neurons. The output of the neurons are calculated by the activations of the neurons in the previous layer and the type of activation function used, e.g. sigmoid, tanh, relu... The results are fed into the inputs of all the neurons in the next layer. The first layer is called the input layer represents the input data, and the last layer is called the output layer where the activations are the predictions for the supplied input data. The layers in between are called hidden layers. Below is a figure showing the topology of a generic fully connected network with 2 hidden

layers.

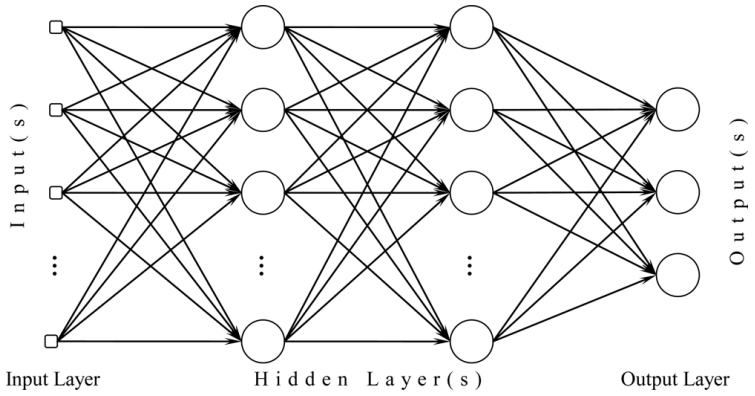

Figure 2.1: Fully Connected Neural Network [**?**]

The activation of the hidden layers of the network can be defined by the equation:

$$\mathbf{a}_i = \phi(\mathbf{W}_i\mathbf{a}_i + \mathbf{b}_i)$$

where $\mathbf{a}_i$ is a vector of length n_i of the activations of all neurons in layer i, $\mathbf{W}_i$ is an $n_i \times n_{i-1}$ matrix of trained weight parameters, $\mathbf{b}_i$ is the trained bias parameter and ϕ is the non-linear activation function.

2.1.2 Convolutional Neural Networks

Convolutional neural networks [34], CNN for short, have been an increasingly popular architecture in the domain of deep learning due to their capability to solve various pattern recognition tasks such as those seen in computer vision and voice recognition [12]. CNNs use weight-tying techniques to create multiple copies of the same neuron to model large amounts of information, while also keeping the total number of trainable parameters low. However, in CNNs, neurons are grouped in what is called a convolutional layer. In the case of 1 dimensional photometric data input, each group of neurons (A) learns from a particular segment of the input data (X). The size of the segment differs depending on the problem, but the goal is to extract local properties of such data such as the intensity of light at a given point or at the start/middle/end of the sequence, trends in the intensity in that particular segment, etc... The features each group represents are usually passed into a fully-connected layer/s (F) as seen in the diagram below.

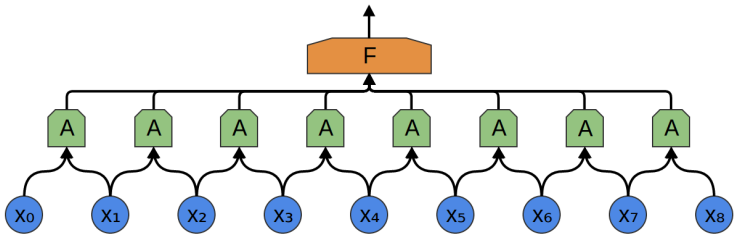

Figure 2.2: 1D CNN -

Convolutional layers (*A*) can be stacked on top of each other to allow the CNN to represent more abstract features in the data. The most popular use of CNNs is in the computer vision domain, and as such, the convolution kernels can be extended to convolve 2D, or even 3D data in cases such as videos which have a temporal component. The diagram below shows a visualization of a 2D CNN with 2 convolutional layers *A* and *B*.

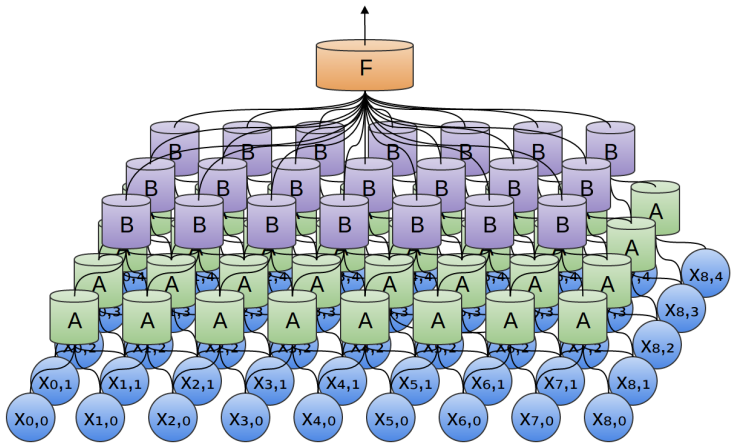

Figure 2.3: Visualisation of 2D CNN

2.1.3 Recurrent Neural Networks

Recurrent neural networks (RNN for short) differ from other architectures in the way data is fed through the network. Unlike feed-forward architectures, RNNs have a recursive mechanism that is capable of creating a persistence of data it has seen in the past. This allows the network to ”remember” what kind of input it has received in the previous time steps. This architecture is therefore ideal for sequential data such as natural language and time series as knowledge of previous data in the sequence is crucial to understand the full context of the problem in the present,

allowing the better future predictions.

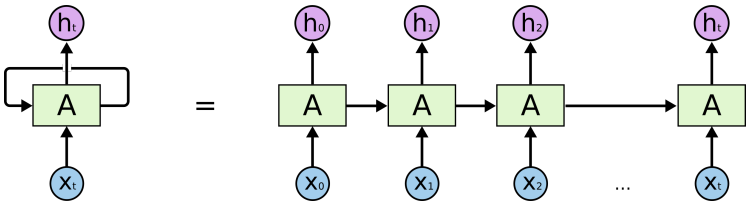

Figure 2.4

The diagram above shows an unfolded RNN, which essentially represents multiple copies of the RNN cell for each time step of the input array. The main issue with RNNs was examined in [3] which states that when the input sequences are too long, vanilla RNN suffers from vanishing gradients that limit how far back in time it can remember the context of the sequence.

A solution to this long-term dependency limitation was proposed in [14] with the introduction of Long Short Term Memory networks (LSTM). The key differences between the architectures is the inclusion of a cell state which carries information throughout the entire chain of LSTM cells. The cell states are tightly controlled by gates which determine what information is added and removed from the state at each time step.

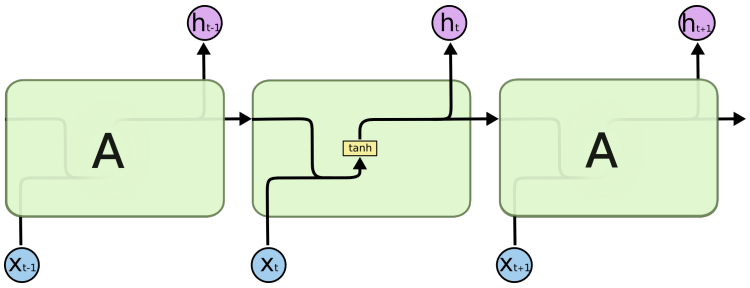

Figure 2.5: Vanilla RNN cell

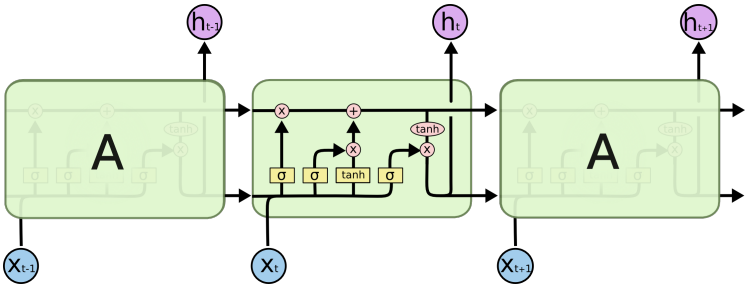

Figure 2.6: LSTM cell

The diagrams above represent the layers contained within a single RNN and LSTM cell. Vanilla RNN is usually made up of a single tanh activation layer, as opposed to the LSTM cell which contains 4 layers that interact with each other in a very unique manner. In short, the first layer in LSTM cell is sigmoid (σ) and determines what information to forget from the current cell state. The next two layers decide which values to update (σ activation layer), and generate a vector of candidate values (tanh activation layer). The final layer determines what to output to the next cell.

2.2 Applications

2.2.1 Exoplanet Classification

The Robovetter [10], is a decision tree based model that incorporates manual vetting procedures used by astronomers to classify transits to improve its prediction capabilities. The system was responsible for the first fully automated catalogues on the archive [10]. Autovetter [7], use random forest models to classify TCEs using statistical information derived from the Kepler pipeline.

More recently, approaches exploring the application of deep neural networks to the problem of classifying kepler lightcurves have been proposed [25, **?**] Research in [25] demonstrates various machine learning techniques to classify exoplanets transients from the Kepler mission photometric time series data. The objective was to learn how much more effective deep learning techniques were compared to traditional approaches to the problem. The approaches discussed were SVM and deep neural network architectures such as multi-layer perceptron and 1D CNN.

The Least-Squares Box Fitting was also developed to provide a baseline traditional approach. The research concluded that non-linear systems such as deep learning neural network architectures were substantially more effective at capturing the variability of shapes and sizes of planet transient light signals. Moreover, the 1D CNN outperformed all other techniques, and due to utilizing 1D convolution filters to extract local features that describe the relationships between the time series data points, no manual feature engineering was necessary to accomplish the results.

Reaching similar conclusions, [30] also demonstrate a novel CNN model that outperforms a fully-connected architecture. The Kepler data is processed in such a way that two independent input representations, a global and local view of the input data are generated for each light curve fed into a multi input network. The best performing configuration was comparable in performance to the Autovetter and Robovetter systems on simulated transit data created by NASA for the purpose of tuning the Kepler pipeline. The researchers successfully discovered and validated two new planets Kepler-80g and Kepler-90i using their proposed CNN architecture.

2.2.2 Gravitational Waves

In the detection of gravitational waves (GW), the CNN architecture as shown to be very effective when combined with wavelet packet techniques used to decompose the input data into a more approachable representation [22]. In addition, [11] also proposed an effective CNN model for the classification and clustering of LIGO glitches which interfere with the process of detecting GWs. The model was pre-training on Imagenet (to learn generic features) and the weights of the final layers were fine tuned with the limited labelled glitch data available to the researchers.

2.2.3 Supernovae Classification

Research in [8] revolves around the application of deep recurrent neural networks to process sequential photometric data captured by the Large Synoptic Survey Telescope (LSST) in order to classify supernovae. The aim was to create two classifiers, one binary (type-Ia supernovae or non-type-Ia) and one with 3 output categories (supernovae types -I, -II and -III). The authors of [8] experiment with many differ-

ent recurrent neural network architectures including uni and bi-directional LSTM (Long-short term memory) networks and GRU (Gated Recurrent Unit) which are ideal for dealing with the influx of photometric light data recorded over many days (weeks-months). This is because to accurately classify these phenomena, the networks need to discern a pattern in how the light spectrum changes over time.

In the computer vision domain, [5] demonstrates Deep-Hits, a rotational-invariant CNN for the classification of supernovae transient images obtained from the Dark Energy Camera (DECam). An extensive comparison with a feature engineered random forest model concludes with the CNN achieving substantially better accuracy while simultaneously eliminating the need to manually extract features from the data.

2.3 Summary

There are a few important insights gained from reviewing the current literature related to Astronomy. Mainly, the realisation that deep learning approaches are, in some cases [30, 25], outperforming traditional methods and coming close to being state of the art at tackling problems in this domain. In particular, the CNN architectures proposed in the various works reviewed above seemed to be the most effective, especially on time series or image classification tasks which CNNs have been demonstrated to excel [18, 22, **?**]. LSTMs on the other hand have received less attention than CNNs in the Astronomy domain, however, they achieved promising results at classifying supernovae [8] suggesting they could perhaps be effective at other classification tasks. There is a strong indication that research into applying deep learning approaches to Astronomy problems has only just begun, as is reflected by the review criteria table in Appendix A which shows that most papers were published in the last 2 years. This has been a pattern in the past before where deep learning approaches have reached state of art performance in a very short period of time, perhaps the same is on the way in the Astronomy domain.

Chapter 3

Kepler Data

The Kepler data processing pipeline [16] consists of several components that convert raw pixel data from the telescope sensors into photometric light curves over multiple timely quarters of ≈ 90 days. In brief, the pipeline process is as follows. Raw data is converted into pixel information via the CAL module [16]. The result are a series of pixel images which are processed by the Photometric Analysis (PA) module to create a photometric time-series of fluxes via Simple Aperture Photometry (SAP) associated to the number of integrated photoelectrons captured during each observation which could be 1 minute long or 30-minute depending on the cadence. Each observation is time stamped with the Julian date at the midpoint of the observation. The produced light curves are further processed using the Presearch Data Conditioning (PDC) module [31]. This procedure corrects the original light curve flux by removing various systematic errors caused by instrumental errors, and flux discontinuities (cosmic rays, solar flares, etc...) which could potentially occlude planet transits. The light curve data produced by the Kepler pipeline is hosted on the Mikulski Archive for Space Telescopes (MAST), whereas information regarding stellar variability of each host star and planetary transit data such as period, depth, radius, mass, eccentricity, e.g..., is presented in interactive tables for public access through the NASA Exoplanet Archive [1].

Radial Velocity [**?**] methods were the most successful at detecting exoplanets prior to the Kepler missions, but were more effective at detecting Neptune-Jupiter sized planets [33] and therefore lacked the precision of Kepler for detecting small

planets e.g moon sized [2]. In Appendix B, figure B.1 shows a histogram of the number of detected planets using popular detection techniques. It is clear that transit detection methods are the reason behind an elevated amount of successful exoplanet detection in the last decade. Similarly, figure B.2 displays a scatter graph confirming that transiting methods of detection have been more successful at detecting planets with smaller masses. Other techniques such as micro-lensing, imaging and pulsar timing also fall short of the success of transit detection techniques which is clearly seen in both plots.

3.1 Training Data

The retrieval of data is the initial procedure before any data analysis, pre-processing or modelling can take place. In this project, this was the retrieval of the raw light curve data from the MAST servers and storing it in a format which is appropriate to be analysed and processed further if need be, and finally fed into machine learning models. To start, labelled data from the Q1-Q17 Data Release 24 catalogue [7] was obtained. The DR24 catalogue is composed of $\approx$ 15740 Threshold-Crossing Events (TCEs). TCEs are any transit-like event that are detected and cross a predefined threshold degree of certainty, and are then subjected to further analysis by the Kepler team to determine if the transit is caused by an exoplanet or other objects/phenomena. The labelled data are available from the **av training set** column of the DR24 TCE Table which were the labels used to train the Kepler teams Autovetter system. The column can have 3 possible TCE labels: planet candidate (PC), astronomical false positive (AFP) and non-transiting phenomenon (NTP). There are a total of 3600 PCs, 9596 AFPs, and 2541 NTPs. Since the problem is binary classification, the labels are organised into two classes, positive i.e contains an exoplanet transit (PCs) and negatives i.e transit is caused by false positive or other (AFPs and NTPs).

Once the labelled data was ready, the light curves for each target star (Kepler ID) in the TCE table were retrieved from the MAST servers using WGET commands. The light curve data is captured in various quarters because due to orbital

motion the telescope needs to recalibrate the lenses by looking away from the target star. However, a single quarter suffices as a TCE signal for each host star. Every attribute was discarded from the selected quarter except PDCSAP FLUX. Using just the PDCSAP FLUX attribute narrows the problem from a multivariate to univariate time series format, thus reducing the complexity of the problem and therefore the training times and computational power required.

3.1.1 Partitioning

The retrieved Kepler data is split into a training (90%) and testing (10%) sets. The training set is used for cross validation to learn the internal model parameters on each training split and optimize the hyper-parameters to find the best performing configuration on each validation split. Finally, the testing dataset is used to evaluate the final model configuration and to check how well it generalizes the data it has been trained with to classify unseen data. This procedure ensures that no information about the testing dataset is leaked into the model fitting and selection stages.

3.2 Analysis

Analysis of the raw Kepler data was paramount to ensure competent final performance of the deep learning models was achieved. The interpretation of the characteristics of the data beforehand helps to determine possible approaches on how best to prepare the input data for the models. To do this, scatter plots are used to visualise the 2 different classes of samples (Planet: 1 - No Planet: 0) in order to distinguish between the differences and similarities of the respective TCEs in each class, and hopefully shed light on any undesired features.

Figure C.1 and C.2 in Appendix C displays a range of scatter plots showing the distribution of data points over time for both classes. At first glance, it is clear that the raw distributions vary in shape significantly, even between samples from the same class. There could be a variety of reasons as to what causes the brightness fluctuations, however, stellar variability and transiting companions i.e planets and eclipsing binaries, are the most probably reasons. The large visible brightness fluctuations can be pinpointed to emerge from physical processes specific to each

star such as pulsations, star spots, solar flares, e.g... a.k.a Stellar astrophysics [**?**]. Pulsations tend to create quasi-periodic signals (Regular peaks with an element of unpredictability) such as those seen in Figure B.3 attached to Appendix B. Star spots can also cause complex variations as the star spins and the spots come in and out of view due to the surface temperature differences. The periodicity of the fluctuations varies from a few days to several depending on how rapidly the star is spinning which determines how frequently the spots come into view. An example of a regular (non pulsating star) but with variability caused by star spots can be seen in Figure B.4 attached to Appendix B Solar flares are also responsible for creating sharp brightness spikes in the signals.

Another finding is that not all transits are clear in the light curve signals containing planets. In figure C.1 some samples have clear downward spikes which may indicate a planet transit, however, a fair amount show no distinct features related to transits even though they are present. The stellar astrophysics processes mentioned above play a role in occluding the planet transits, as well as the transit specific properties such as depth, period and signal-to-noise ratio. This is a major obstacle when trying to classify both classes of samples correctly as the machine learning models need to be able to extract properties that are unique to a particular class of data, and if transits are difficult to observe, the models will evaluate a high number of actual transits as false negatives which is undesirable.

3.3 Pre-Processing

One of the main aims of the study is to further back the findings of previous research on exoplanet transient detection using deep learning techniques [30, 25] by analysing the potential for accurate detection without the need any kind of pre-training feature engineering techniques which are typically required with traditional learning techniques. As such, only necessary preprocessing steps are applied to the data in order to extract relevant feature information and remove unnecessary noise from the TCE signals which can have a detrimental effect on the final classifier performance.

3.3.1 Flattening

The stellar astrophysics phenomena explored during the analysis were found to add a large amount of unwanted variability and noise into the Kepler data, as a result, increasing the complexity of the data and making the task of classifying the TCE signals more difficult. To remove the low frequency variability, a procedure called light curve flattening was applied to each TCE sample. Flattening de-trends the light curve of the low frequency variability while preserving valuable transit information by fitting a piecewise polynomial spline to the light curve and dividing the original light curve by the spline.

In Appendix D, Figure D.1 displays the original PDCSAP Flux of KIC 12557548 retrieved from the Exoplanet Archive, and Figure D.2 shows the de-trended version. It is clear that the variability as been removed entirely from the light curve. For a better view, figures D.5 and D.8 show 10 flattened samples of positive and negative TCEs respectively.

3.3.2 Folding

After the light curves are flattened to remove low frequency variability caused by stellar astrophysics, a procedure called phase folding is applied to the flattened light curve to fold the light curve transit period around the centre of the transit event. The transit period is the interval between planetary transits, and the event centre corresponds to the time of the centre of the first detected transit in Barycentric Julian days [1]. Both attributes (tce_period, tce_time0bk) are retrieved from the TCE table for each light curve.

For most samples, this procedure helps expose the transit features from the rest of the light curve, as can be seen by comparing the flattened version of KIC 12557548 in Appendix D, Figure D.2 and the flattened phase folded version in Figure D.3. For a better view, figures D.6 and D.9 show 10 phase folded samples of positive and negative TCEs respectively.

[1] `https://exoplanetarchive.ipac.caltech.edu/docs/API_tce_columns.html`

3.3.3 Binning

The folded light curves are binned by defining a set number of bins, each with a specific width along the time axis. Each bin represents the median value of the data points that fall within the boundaries. The total number of bins used was b where $b = 2001$. Each bin was given a width of $p \times f$ where p is the transit period obtained from the TCE table and f is a width factor of $1/b$. The result is a 1D array of size b that represents the binned light curve. This procedure reduces the total number of points and eliminates unnecessary scatter in the light curves. For a better view, figures D.7 and D.10 show 10 binned samples of positive and negative TCEs respectively.

The flattening, folding and binning procedures are derived from the research in [30].

3.3.4 Missing Values

Missing values are a very common occurrence in most raw data such as those from the Kepler pipeline. Most retrieved light curve signals featured a fair amount of *NaNs* which are incompatible with the input required by most machine learning models, especially CNNs which have no input masking support. A myriad of techniques were attempted to remove missing variables. The simplest was to remove every time step which had a missing value, however, this was not suitable as it would mean destroying chunks of the signal, thus creating unrealistic samples. A better technique was to impute the missing values with the mean value of the full time series. The resulting signals retained much better overall structure when compared to removal of NaNs method and before imputation.

A more advanced and computational expensive method was to use the functionality in the R package (ImputeTS) to fit an Auto-Arima model to each time series and use the trained model to forecast the missing values. However, this was an unsuccessful experiment as the imputations were very inaccurate and introduced far more noise into the signal compared to simply imputing with the mean value in the time series.

3.3.5 Standardization

The ranges of values in the raw time series observations tend to vary significantly from one sample to the next. This can cause certain observations to have more influence on the classifier due range being larger than others. This problem can also cause gradient descent based optimization methods to converge much slower, and even cause exploding gradients [21]. Feature scaling techniques such as normalization and standardization, in conjunction with Batch normalisation layers, are commonly used to resolve this problem. The dataset is standardized to have a mean of 0, and unit variance of 1. To ensure that details about the distribution of the test data doesn't leak into the training data, standardization is applied to each individual subset separately.

3.3.6 Reshaping

Reshaping involves shifting the dimensionality of the data before it is fed into the models. The input layers for both the CNN and LSTM architecture require that the correct shape of the input data is specified. The correct shape is a 3D dimensional representation of the input, such that $X = (Samples, Timesteps, Features)$. In our case, samples is the number of rows in the data, time steps is the total number of columns (2000) and the number of features (output dimensionality) is 1 since the time series is univariate.

Chapter 4

Modelling

4.1 Configuration Selection Procedure

The selection of the most appropriate model to solve the classification problem can be categorized as a search optimization problem. The search space in the case of machine learning, are the range of values to search for each hyper parameter. Hyper parameters differ from the internal parameters learned during training in that the values need to be set before the learning commences. The number of hyper parameters to tune varies depending on the model, but the most crucial ones involve the topology of the network (number of layers and hidden units), learning rate, momentum, etc... Hyperopt [1] was used to define an objective function with the goal of maximizing the AUC score during each evaluation of a model configuration obtained from the defined search space. The search optimization algorithm used is called Tree of Parzen estimators (TPE) [4] which is a form of Bayesian optimization. The search spaces for each model architecture are defined in the tables below.

[1] https://github.com/hyperopt/hyperopt

Table 4.1: CNN Parameter Search Space

Parameter	Distribution	Space
Number Blocks	Choice	[0,1,2,3]
Filters	Choice	[8,16,32,64,128,254]
Kernel Size	Choice	[3, 5, 7, 9]
Pooling Type	Choice	[Max, Average]
Pooling Size	Choice	[2,3,4]
Pooling Strides	Choice	[2,3,4]
Conv Dropout	Uniform	[0.0 - 0.35]
FC Dropout	Uniform	[0.0 - 0.6]
FC Units	Choice	[32,64,128,254]
Batch Size	Choice	[16,32]
Learning Rate Mult	Log Uniform	[-0.5 - 0.5]
Momentum	Uniform	[0.0 0.5]
Batch Normalisation	Choice	[32,64,128,254]
Number Epochs	Uniform	[10.0 - 50.0]
Activation	Choice	[ReLU, PReLU]

Table 4.2: LSTM Parameter Search Space

Parameter	Distribution	Space
Number of LSTM Layers	Choice	[0, 1]
Number of LSTM Units	Choice	[2, 5, 10, 15]
Number of Fully Connected Layers	Choice	[0, 1, 2]
Number of Fully Connected Units	Choice	[32, 64, 128]
Dropout	Uniform	[0.0 - 0.5]
Batch Size	Choice	[16, 32]
Learning Rate Mult	Log Uniform	[-0.5 - 0.5]
Momentum	Uniform	[0.0 - 0.5]
Batch Normalisation	Choice	[32, 64, 128, 254]
Number Epochs	Uniform	[10.0 - 50.0]
Activation	Choice	[ReLU, PReLU]

During the evaluations, to mitigate variance and bias in the results, and prevent under/over-fitting the selected model configuration to the dataset, stratified k-fold cross validation is performed during each evaluation of a hyper-parameter configuration. This procedure splits the dataset into k subsets (where $k = 5$) and the model is trained and validated on the subsets k times. The performance of the model during each evaluation is averaged across all trials. The main advantage of this is that each sample in the dataset gets to be in the validation set at least once and in the training set $k-1$ times. This as been shown [**?**] to reduce both variance and bias in the final model as the totality of the data is being used in the fitting and validation procedures, as opposed to the traditional hold-out method where a fraction of training, validation and testing data is lost. Stratification was employed as the sampling strategy to ensure an even distribution of each target class was represented in each split of the data. This was appropriate because of the slight imbalance between the positive and negative TCE samples in the dataset.

After the optimization, the best configurations are evaluated on unseen test data to confirm the cross validation performance.

4.2 Selected Configurations

After the model selection procedure described in the previous section, the final result is single best performing model configuration for each architecture. The CNN and LSTM optimizations ran for 50 and 20 evaluations respectively. This section aims to explain in detail, the topologies of the best neural network configurations.

4.2.1 CNN

The most optimal configuration determined by the optimization contained 2 convolutional blocks. Each block in the convolutional architecture is made up of the following layers (in order):

1st Layer: Convolutional layer with 128 filters and a kernel size of 5

2nd Layer: Batch Normalisation Layer

3rd Layer: Max Pooling layer with window size of 3 and stride of 2

4th Layer: Dropout layer used to drop 0.086% of the weights.

5th Layer: PreLU activation function

After the convolutional blocks, the extracted features are passed into 4 fully connected blocks which are made up of the following layers (in order):

1st Layer: Fully Connected layer with 64 hidden units

2nd Layer: Batch Normalisation Layer

4th Layer: Dropout layer used to drop 0.265% of the weights.

5th Layer: PreLU activation function

The model was trained for a total of 36 epochs using the Stochastic Gradient Descent optimization algorithm with a learning rate of 0.00134341939565237, momentum of 0.25 and decay of 0.0001. Training samples where fed into the network in batches of 32.

A diagram of the network topology can be found in Appendix E.

4.2.2 LSTM

The most optimal configuration determined by the optimization has 1 LSTM block. Each block in the LSTM model is made up of the following layers (in order):

1st Layer: LSTM layer with 10 hidden units

2nd Layer: Batch Normalisation Layer

4th Layer: Dropout layer used to drop 0.298% of the weights.

5th Layer: PreLU activation function

After the LSTM block, the extracted features are passed into 2 fully connected blocks which are made up of the following layers (in order):

1st Layer: Fully Connected layer with 64 hidden units

2nd Layer: Batch Normalisation Layer

4th Layer: Dropout layer used to drop 0.298% of the weights.

5th Layer: PreLU activation function

The model was trained for a total of 43 epochs using the Stochastic Gradient Descent optimization algorithm with a learning rate of 0.00164341939565237, momentum of 0.25 and decay of 0.0001. Training samples where fed into the network in batches of 16.

A diagram of the network topology can be found in Appendix E.

4.2.3 Layer Explanation

Batch Normalisation layers [15] are used to reduce the internal covariate shift between layers. This phenomenon is caused as a result of changes of the input distributions between different layers during the learning/training of internal model parameters. A remedy for this problem is to incorporate normalization on a per batch basis into the models architecture. BN layers have been shown to largely reduce the trainings times of models by allowing larger learning rates and acting as a regularizer for over-fitting, and in some cases removing the need for Dropout layers [15].

Max Pooling layers are also used in conjunction with convolution layers in the CNN architecture to enable down sampling of the feature maps extracted by convolutions in the previous layers, and enable representation of higher level features in the input sequence data.

Dropout layers are stacked on top of every convolutional and fully connected layer in the network. Dropout is a regularization technique that randomly initialises a percentage of the weights in the previous layers to 0. This removes certain connections from the network and has been shown /cite to reduce the possibility of over fitting the data

The activation function used throughout the network is the Parametric rectified linear unit (PreLu) [13]. PreLu activation functions have been shown to

Binary classification requires that the model outputs a single value i.e 1 or 0. Therefore, the final layer of the model uses a sigmoid activation function that outputs a probability value between 0 and 1. The loss function to minimize for this problem is binary cross-entropy, and the optimization algorithm used for both models is Stochastic Gradient Descent.

Chapter 5

Results

5.1 Evaluation Metrics

Proper assessment of the model architectures on the test dataset requires the use of metrics and visualizations that portray honest, and not biased assumptions about the classification performance and generalization of the model. Simply using classification accuracy as a primary metric would be a wrong choice as the dataset suffers from slight class imbalance due to there only 4600 positive samples when compared the 10000 negative samples representing astronomical false positives. The metrics used are as follows:

- Accuracy: The fraction of correct predictions over the total number of predictions made.
- Precision: The fraction of observations classified as containing transiting planets that actually have transits (Reliability). $Precision = TP/TP + FP$
- Recall: The fraction of positive samples over all the samples that should have been predicted a positive (Completeness). $Recall = TP/TP + FN$
- AUC (Area Under the Curve): Probability that a randomly selected positive TCE sample (exoplanet) is scored higher than a randomly selected negative TCE sample (no exoplanet).

The accuracy, precision and recall metrics are affected by the probability threshold used to determine how a TCE is classified. The threshold used is 0.5, which means

a probability score above that equates to a TCE being classified as containing a transiting planet, and any score below means the TCE is not caused by a planet.

5.2 Best Performing Configurations

5.2.1 CNN

The performance of the best configuration after hyper-parameter optimization is displayed in Table 5.1.

Table 5.1: CNN Results

Dataset	Accuracy	AUC	Precision	Recall	F1
Flattened, Folded and Binned	0.93529	0.96453	0.93839	0.93529	0.93629

The confusion matrix generated from the predictions on the unseen test data is attached as Figure F.1 in Appendix F.

The AUC ROC plot is attached as Figure F.2 in Appendix F.

The Precision-Recall plot is attached as Figure F.3 in Appendix F

The training and validation epoch loss and accuracy are plotted in figures F.4 and F.5.

5.2.2 LSTM

Table 5.2: LSTM Results

Dataset	Accuracy	AUC	Precision	Recall	F1
Flattened, Folded and Binned	0.93464	0.9652	0.93673	0.93464	0.93539

The confusion matrix generated from the predictions on the unseen test data is attached as Figure G.1 in Appendix G.

The AUC ROC plot is attached as Figure G.2 in Appendix G.

The Precision-Recall plot is attached as Figure G.3 in Appendix G

The training and validation epoch loss and accuracy are plotted in figures G.4 and G.5.

Chapter 6

Discussion

6.1 Results Comparison

In terms of prediction performance, both model architectures exhibited exceptional performance on the unseen test dataset. As discovered in the literature review, the CNN architecture was almost guaranteed to perform very well on this particular classification problem as demonstrated already by [30, 25], as-well as in many other problem domains in Astronomy [22]. On the other hand, the LSTM architecture has received much less attention in this field, however, recurrent architectures such as LSTM achieved remarkable results in classifying supernovae [8]. The hypothesis formulated during the literature review that LSTMs had the potential to exceed, or come close to CNN performance at classifying exoplanets was validated by the final results.

By looking at the confusion matrices (Figure F.1 and G.1) the number of correct predictions by both models well exceeds the number of false positives and negatives. Out of a total of 1200 true negatives and 350 true positives in the test dataset, the CNN model produces only **66** false positives and **33** false negatives, whereas the LSTM produces **62** false positives and **38** false negatives. The low number of false negatives is justified by the high recall scores achieved by the CNN and LSTM models, they were: 93.529% and 0.93464% respectively. This equates to the models being able to correctly predict positive TCEs as containing an exoplanet transit $\approx 93.5\%$ of the time.

The precision-recall (PR) curves generated for the models allowed the relationship between the precision and recall scores for both classes of samples to be observed. The relationship in most cases can be seen as a trade-off between the number of predictions that are relevant samples (precision) and the number of relevant samples that are predicted (recall). In this case, the relevant samples are the exoplanet TCEs (positive samples) that the models should aim to predict as frequently as possible. The goal of the PR curve is to have the largest area i.e closest to 1. The PR curves for the CNN (F.3) and LSTM (G.3) show a similar pattern in that they both display a lower PR Area score for the positive class (exoplanet) than the negative class (not exoplanet). A possibility could be that the lower number of total positive samples in the training and test sets could have caused the models to perform worse on the positive class. However, the overall performance of both models is nonetheless more than acceptable with an average PR area score of $\approx$ 0.97 for the CNN and LSTM architecture alike.

This similarity in performance is also reflected by the AUC ROC plots for the CNN (Figure F.2) where the average AUC of the model is **0.96**, and the LSTM (Figure G.2) is **0.97**. These plots show the true positive rate (sensitivity) against the false positive rate (specificity) of the both models. The sensitivity a.k.a recall, is associated to the proportion of positive TCE samples that are correctly predicted, e.g. the percentage of negative TCEs (astronomical false positives and non transiting phenomena) that get correctly predicted as not resembling an exoplanet transit [23]. The specificity is the proportion of negative samples that are correctly predicted as such, e.g. the percentage of positive TCEs that get correctly predicted as containing an exoplanet transit [23].

Monitoring the cross-entropy loss and accuracy after each epoch was also crucial to understand how well the model was being fitted to the data. The training-validation accuracy and loss plots for the CNN model are displayed in figures F.4 and F.5 in Appendix F. Likewise, the LSTM plots are displayed in figures G.4 and G.5 in Appendix G. A major concern would be if any plots showed a large deviation between the training and validation losses/accuracy as the epochs progress,

as this would indicate the models are over fitting the data, e.g. the training loss is low and the validation loss is high. The accuracy shows no major deviation as the epochs progress in either model, however, the loss plots show a slight deviation between the training and validation losses around epoch 10 where the validation loss seems to plateau and the training loss continues to decrease. This means that both models have slightly over fit the data, but not to the point where it is detrimental to the overall performance of both models on the unseen test dataset. Choosing to stop training at an earlier epoch would have prevented the slight over fitting from occurring altogether.

6.2 Architecture Comparison

While both architectures achieved remarkable performance, there are some key differences that differentiate them and make one more appropriate than the other. The CNN architecture provides excellent feature extraction capabilities through the use of convolutional layers, which allow for the representation of certain regions of the input light curve as predominant of an exoplanet transit or of a false positive such as an eclipsing binary system which produces a similar transit signal to an exoplanet. The ability to model the spatial relationships between data points is also a benefit over fully connected architectures. The max pooling layers provide another level of feature abstraction by enabling down-sampling of the feature maps extracted by the convolutional layers into higher level features. This not only augments the networks capability of representing local feature groups, but also keeps the number of trainable parameters much lower than a fully connected architecture.

In contrast, the LSTM architecture offered similar potential for modelling the data due to their inherent capability to remember states of information from previous time steps in the input, even in long sequences such as the Kepler light curves. The classification probability is determined by the final output layer (sigmoid) depending on cell states that fire on particular time steps of the input. While there are clear differences in the way each architecture models data, the final classification performance of both models are practically equal.

Convolutional layers are highly optimized for parallel computation on a GPU when compared to LSTM layers. This allows a larger CNN network configuration to be trained much faster than a significantly more simple LSTM architecture. This is in-line with the final best configurations of the models which the CNN topped $4.3m$ internal trainable parameters, compared to the LSTM which had $1.2m$. Even though the CNN architecture was more complex, the total time taken to perform 5-fold cross validation was 49 minutes compared to the LSTMs 3 hours and 30 minutes. However, both the training times and prediction performance of the LSTM could have been boosted if Truncated Back Propagation Through Time (TBPTT) was considered instead of regular BPTT [32].

6.3 Improvements

Obtaining information first hand from field experts is extremely valuable in the success of a machine learning project, this is even more important when the problem is within the context of a specialised scientific field/s such as Astronomy and Astrophysics. Given more time, directly reaching out to experts with the goal of obtaining more information to construct a more concise picture of the problem domain would be a top priority. The information obtained could have potentially sped up and improved many parts of the research and development process.

The problem of classifying exoplanets required a substantial amount of data pre-processing due to factors such as stellar variability and the diverse differences in attributes of transiting planets such as depth, duration, period, etc... These factors add a large amount of variance and noise into the raw data samples, as such, transforming the original data to make it suitable to be fed into the models was paramount to obtain the final prediction performance. The light curve binning procedure could have been improved to create multiple representations of the same sample similarly to the method proposed in [30]. Using both global and local views of each transit as inputs to the models was demonstrated to be more effective than a model using just a single representation like the approach used in this research. This would require extending the model architectures to accept multiple input representations and

generating an extra ”local” view of the transit.

Stitching quarters together could have potentially had an affect on the overall performance of the models. This is because there is a possibility that some input positive samples could have missing transits due to planets that have orbital periods that are longer than the 90 day quarters, making them fall out of scope of particular quarters. However, stitching would also increase computational time required is increased by a factor of 2 for each quarter stitched, as such, the overall affect on performance is assumed to be minimal in this research.

The evaluation of the models performance could also be improved in a few areas given more resources and time. For example, a thorough performance comparison between the proposed models and the current state of the art such as the Kepler teams' Robovetter and Autovetter systems would have added credibility to the results achieved in this study. Moreover, an elaborate analysis of the models predictions, specifically the false positives and negatives, would have shed light on particular features in the samples that the model was unable to extract, and in turn helped inform the recommended modifications to the model architecture and data processing to improve the final performance. Simply plotting a scatter plot of the attributes of the incorrect predictions such as transit information and stellar variability against the prediction accuracy of the models would have sufficed for a better overall evaluation.

The lack of sufficient computing resources also limited the extent to which the hyper-parameter optimization procedure could be performed, as a result, the hyper parameter search space was purposefully restricted to prevent very large network configurations with large numbers of trainable parameters to be evaluated. Despite the fact that the model training was dramatically sped up by harnessing the power of parallel computation through CUDA using a GTX970 GPU, certain configurations could still take many hours to train. The use of cross validation during the tuning procedure to reduce variance in results and prevent over-fitting the data did not help. Renting a cloud based GPU compute instance would have allowed a larger total number of evaluations to be performed, and testing more complex networks and a

much larger range of hyper-parameters which may have achieved a better overall performance on the data. Larger networks are presented in [30, 25] and perform very well on the Kepler data.

The total number of samples used to train, validate the hyper-parameters and test the models could have benefited from some kind of augmentation. A fair amount of research and even development experiments were conducted using methods for oversampling the minority class (positive TCE samples) in order for a balanced dataset to be created. However, as was discovered, time-series problems present a more complex challenge for oversampling techniques such as SMOTE, ADASYN etc... This is because data-points in the same sample are related to each other via a temporal component. The Kepler time series data is also very high dimensional and would require an oversampling technique that retains the covariance structure between each data point in order to create realistic synthetic samples that represent (to a close enough degree) a real Kepler light curve. Research in [6] presents INOS, a novel oversampling technique for imbalanced univariate time series classification problems such as the one in this research. Perhaps using INOS for oversampling the minority positive samples would have lead to better overall performance, however, if implemented, extensive analysis of the synthetic samples would be required to access the quality of the light curves before fitting the new data to the models.

Another suitable approach would be to train the model with a fully synthetic dataset and evaluate it on real Kepler data as was seen in [25]. The synthetic dataset in this case was generated by utilizing equations from [24] to generate noisy transit signals with quasi-periodic trends much like the real Kepler light curves.

Chapter 7

Conclusion

Big data makes the manual interpretation of exoplanet candidates a very labour intensive task, and with the launch of new exoplanet hunting telescope surveys such as TESS, the volume of data will continue to grow in the coming years. What makes the task of classifying exoplanets so difficult is the variability in the light curves caused by stellar astrophysical processes, instrumental errors, and other celestial objects or phenomena other than exoplanets. Therefore, only a human with significant domain expertise or a system capable of modelling the non-linearity in the light curves is suitable for such a problem.

In this research, two deep neural network architectures were presented as potential approaches to the problem of exoplanet candidate classification. Both the CNN and LSTM models are able to accurately distinguish the subtle differences between transiting exoplanets and false positives such as eclipsing binaries and instrumental errors. The use of the LSTM architecture was a novel contribution of this research, as currently there exists no publication that evaluates the effectiveness of this architecture at classifying exoplanets using the Kepler data.

The research questions formulated at the beginning of the study were answered throughout the various sections in the thesis, as well as the satisfaction of all the research objectives. In terms of educational objectives, a strong grasp of the current literature in the Astronomy and deep learning domain was gained, as well as extensive development experience applying pre-processing techniques to a complex univariate time series dataset, and knowledge on the best practices for deep neural

network modelling, tuning and evaluation procedures.

To conclude, it is clear from the research conducted that deep learning approaches will play a big role in the coming years in the automation of various tasks in the Astronomy domain. However, for the time being, human intervention will be necessary to validate potential candidates classified by such systems, as no machine learning model can entirely surpass the precision that real physical models offer.

Appendix A

Literature Review Criteria Table

Paper	Date	Topic	Architecture	Problem
Searching for Exoplanets using AI [25]	2017	Exoplanet Classification	CNN	Time-Series
Identifying Exoplanets with Deep Learning: A Five Planet Resonant Chain around Kepler-80 and an Eighth Planet around Kepler-90 [30]	2017	Exoplanet Classification	CNN	Time-Series
Deep Recurrent Neural Networks for Supernovae Classification [8]	2016	Supernovae Classification	LSTM, GRU	Time-Series
A Method Of Detecting Gravitational Wave Based On Time-frequency Analysis And Convolutional Neural Networks [22]	2017	Gravitational Wave Classification	CNN	Time-Series
Glitch Classification and Clustering for LIGO with Deep Transfer Learning [11]	2017	Gravitational Wave Classification	CNN	Time-Series

Appendix B

Background

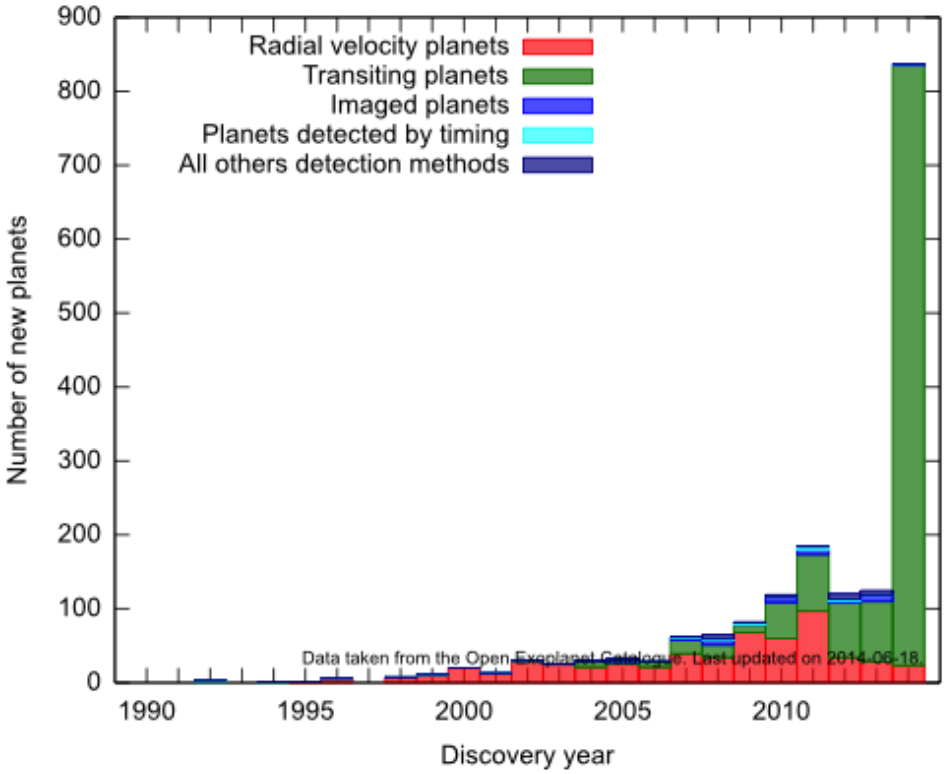

Figure B.1: Histogram retrieved from the Open Exoplanet Catalogue [1]showing the total number of exoplanet detections and the popularity of a variety of methods in the last decades. It is clear that transiting methods have been the most popular in the last decade.

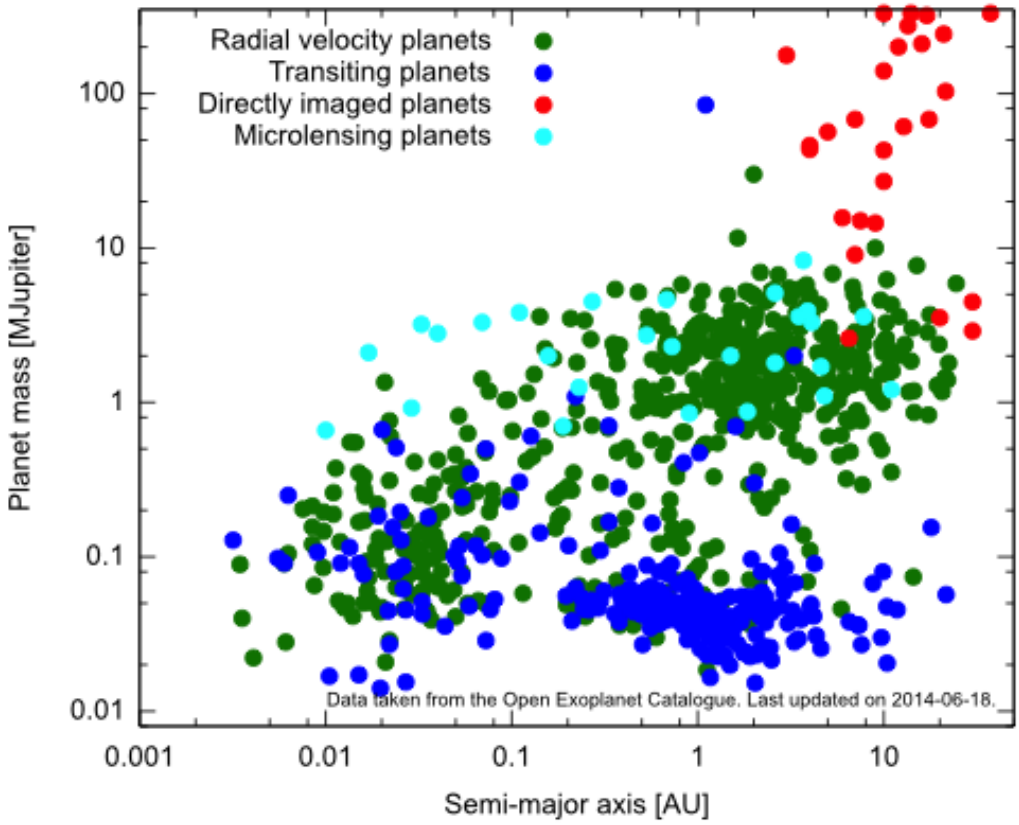

Figure B.2: Scatter plot retrieved from the Open Exoplanet Catalogue [2]showing most commonly used detection methods in relation to the mass of the planets. Transiting techniques are the most effective for detecting planets of all mass, especially with very small.

[1]https://github.com/OpenExoplanetCatalogue/open_exoplanet_catalogue

[2]https://github.com/OpenExoplanetCatalogue/open_exoplanet_catalogue

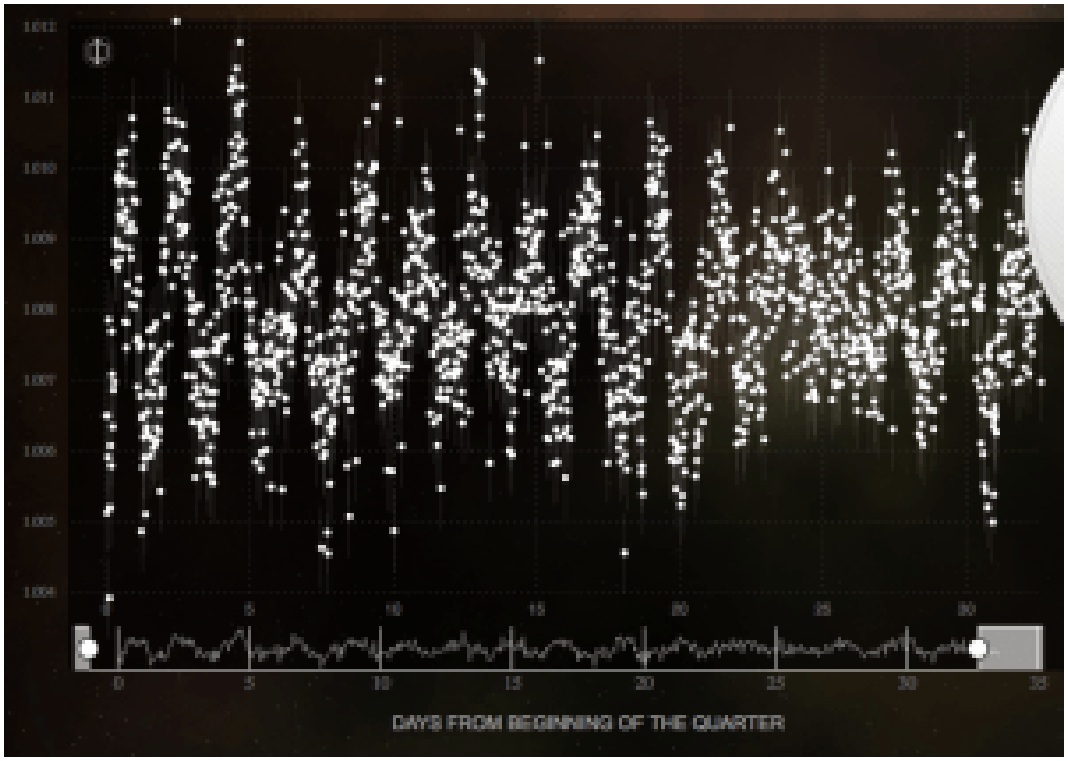

Figure B.3: A pulsating star (https://i0.wp.com/blogs.zooniverse.org/planethunters/files/2010/12/puls11-300x213.png)

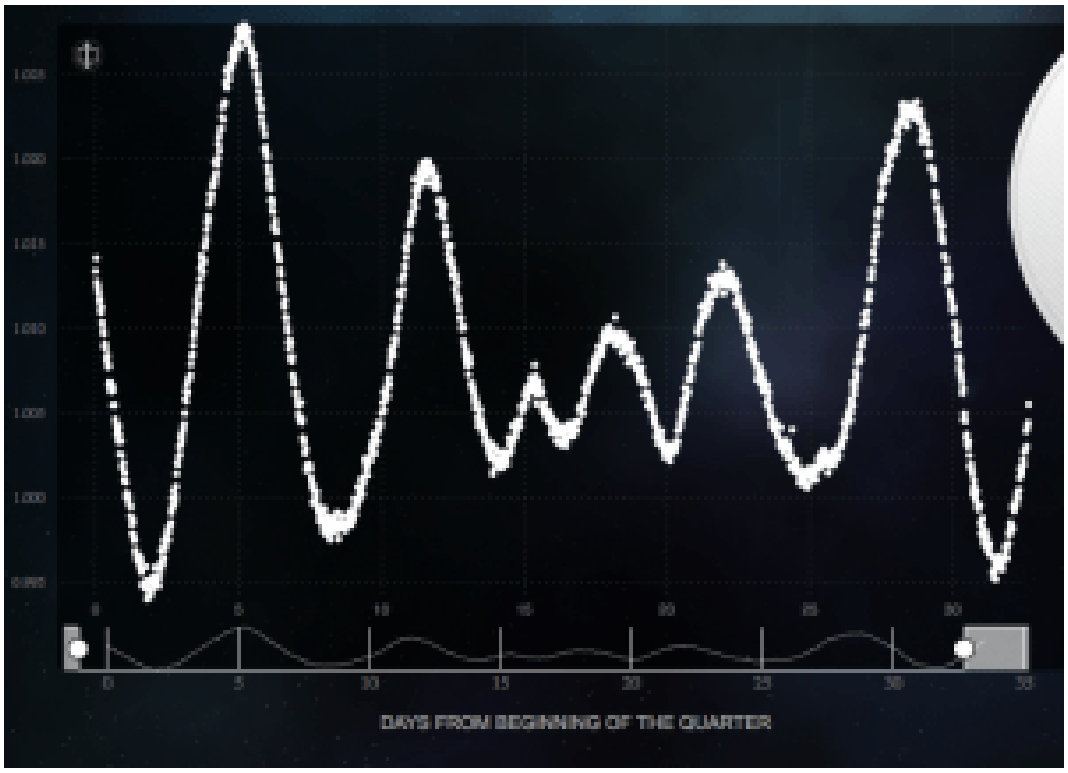

Figure B.4: Regular but variable star caused by star spots (https://i1.wp.com/blogs.zooniverse.org/planethunters/files/2010/12/irreg1-300x216.png)

Appendix C

Analysis Plots

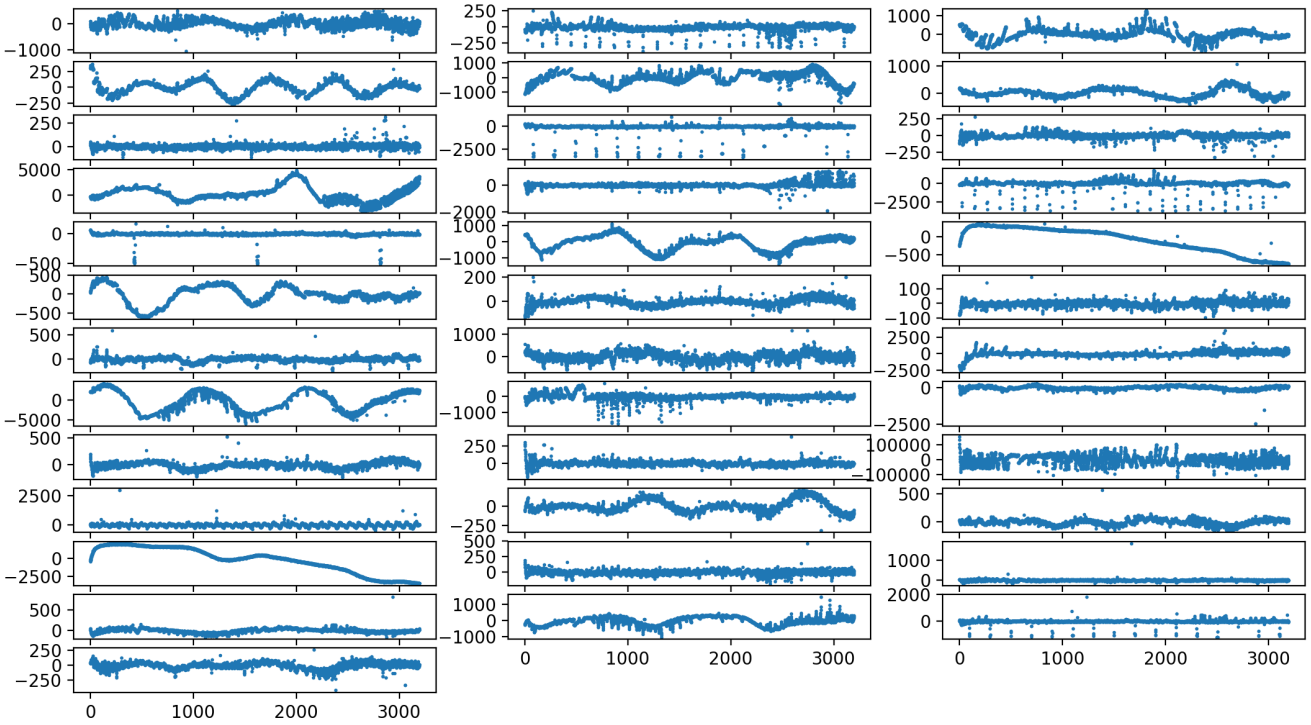

Figure C.1: PDCSAP FLUX samples as retrieved from the MAST servers. Each figure is a positive sample and therefore contains an exoplanet transit.

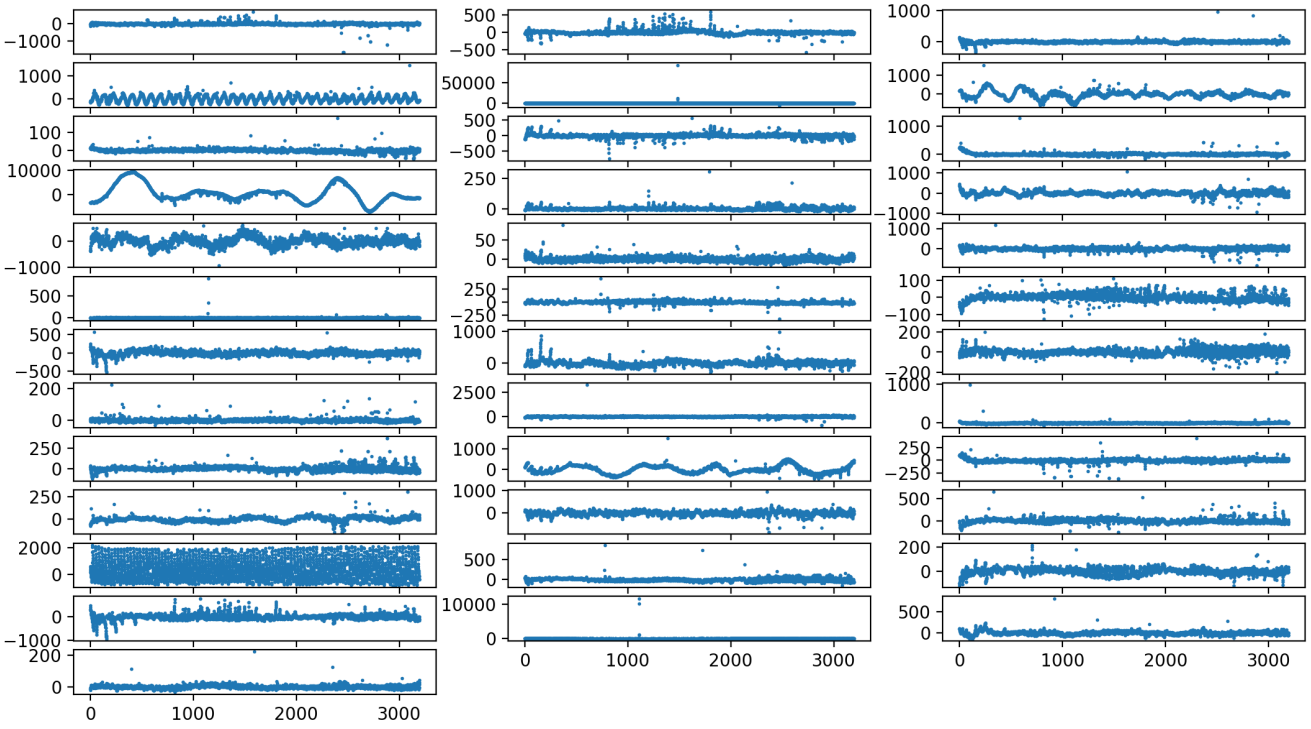

Figure C.2: PDCSAP FLUX samples as retrieved from the MAST servers. Each figure is a negative sample and therefore doesn't contain an exoplanet transit.

Appendix D

Processed Data Plots

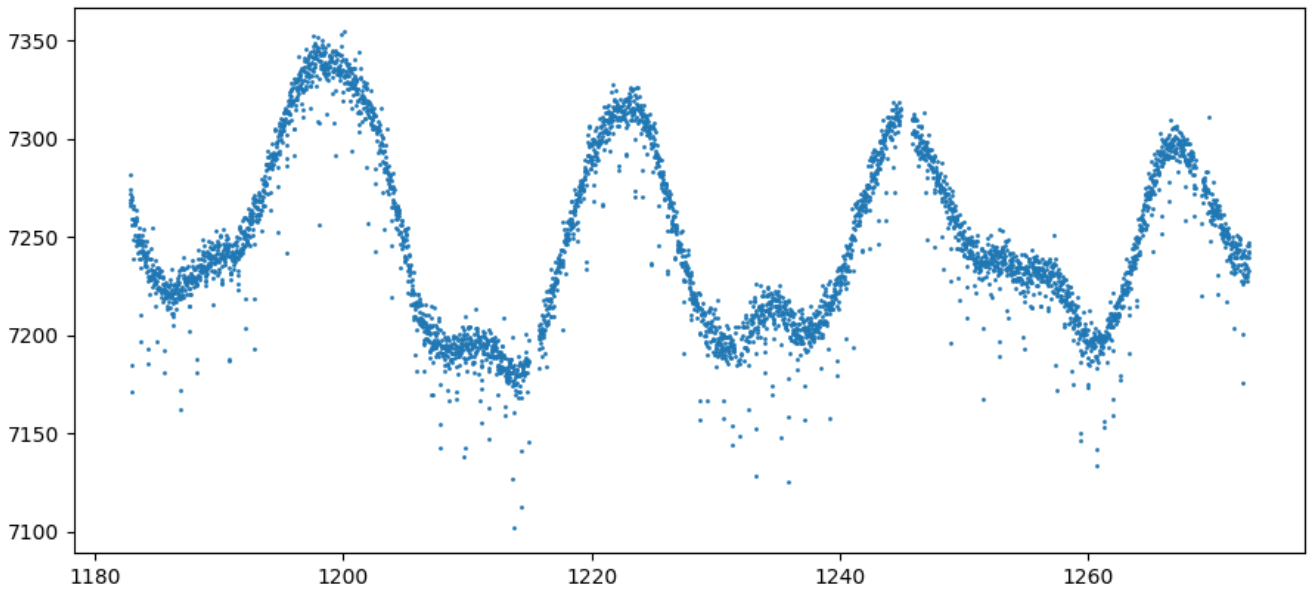

Figure D.1: Original PDCSAP Flux Light Curve (KIC 12557548)

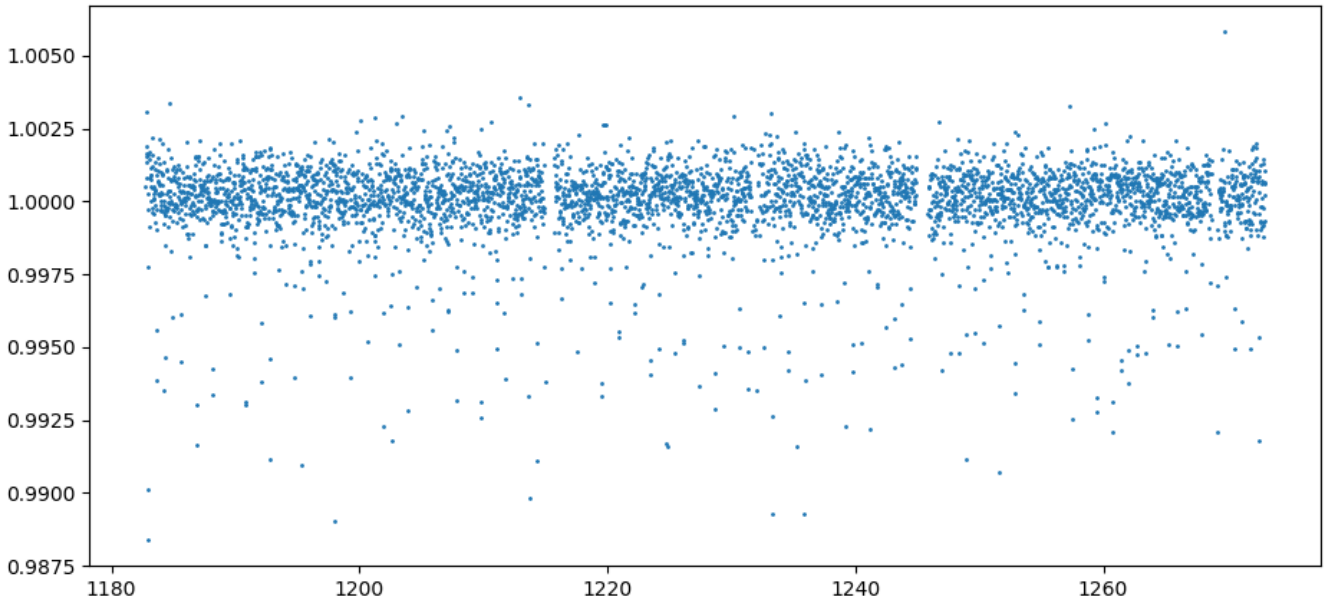

Figure D.2: Flattened Light Curve (KIC 12557548)

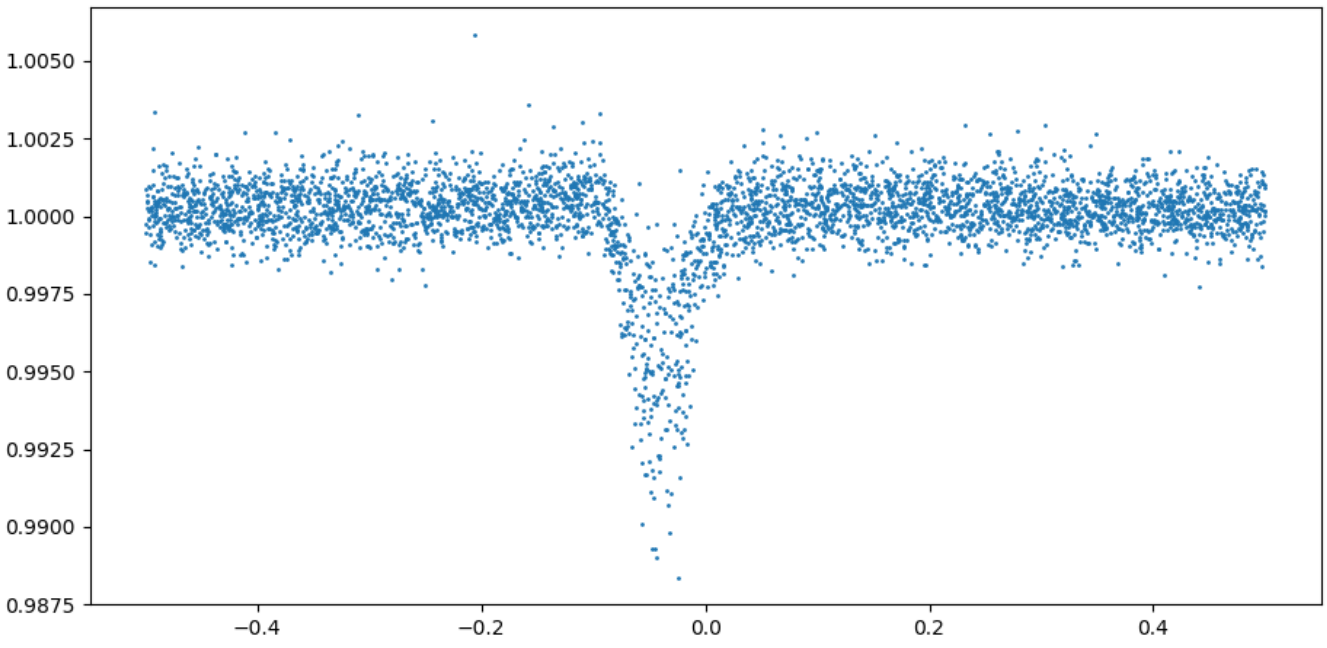

Figure D.3: Folded Light Curve (KIC 12557548)

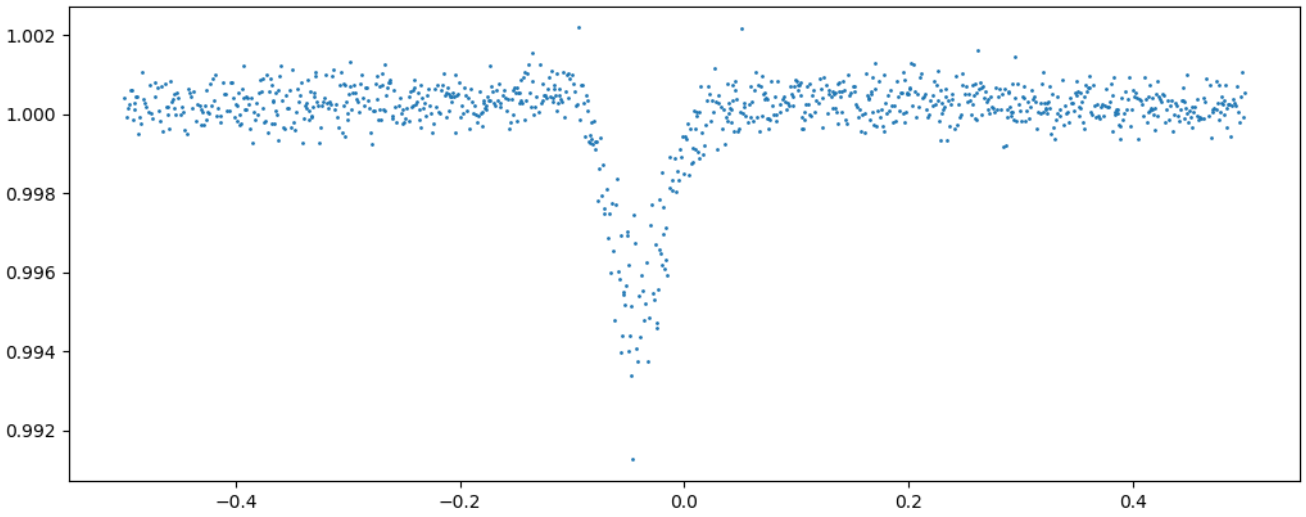

Figure D.4: Binned Light Curve (KIC 12557548)

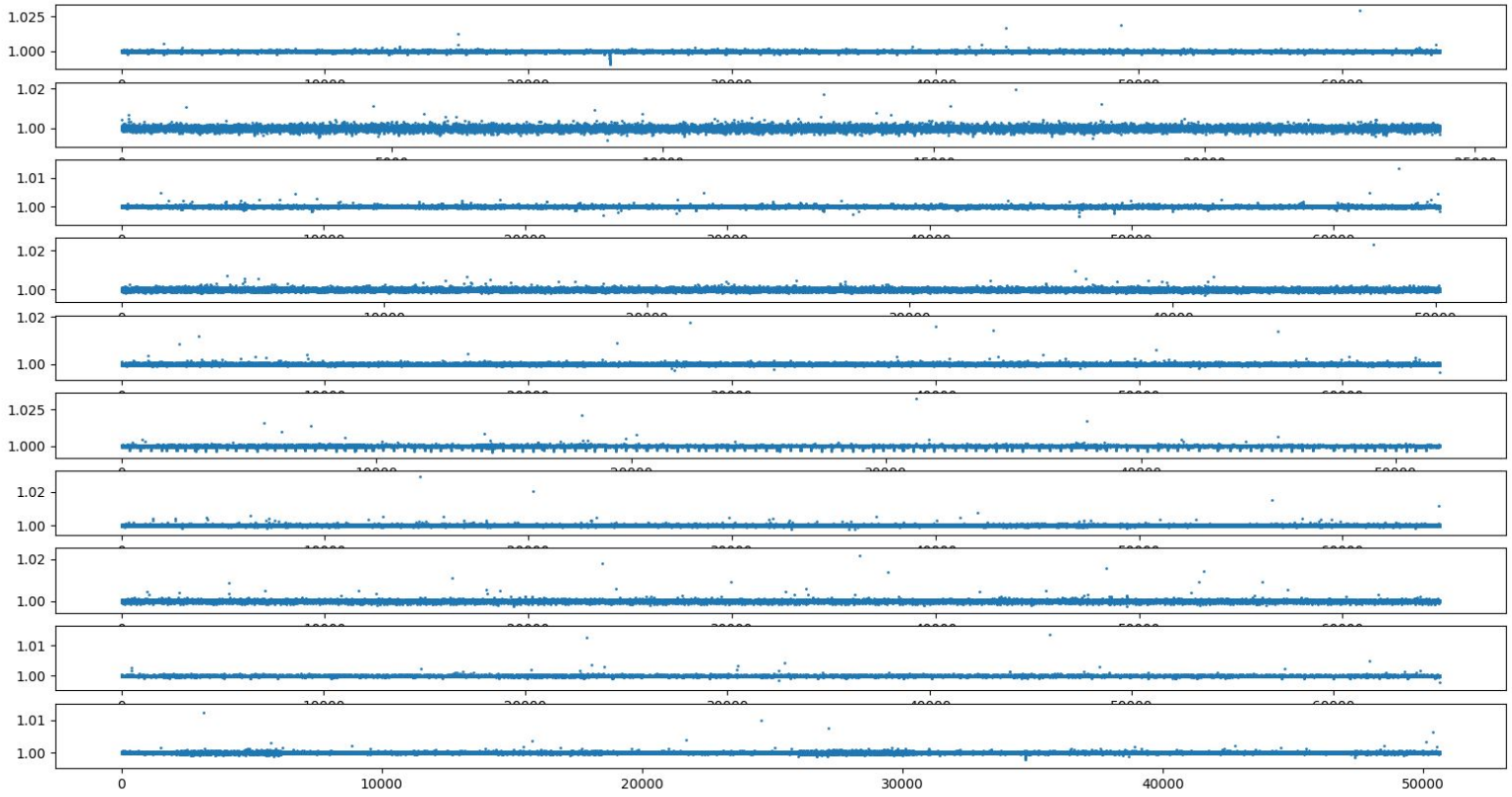

Figure D.5: Flattened Positive Sample Light Curves

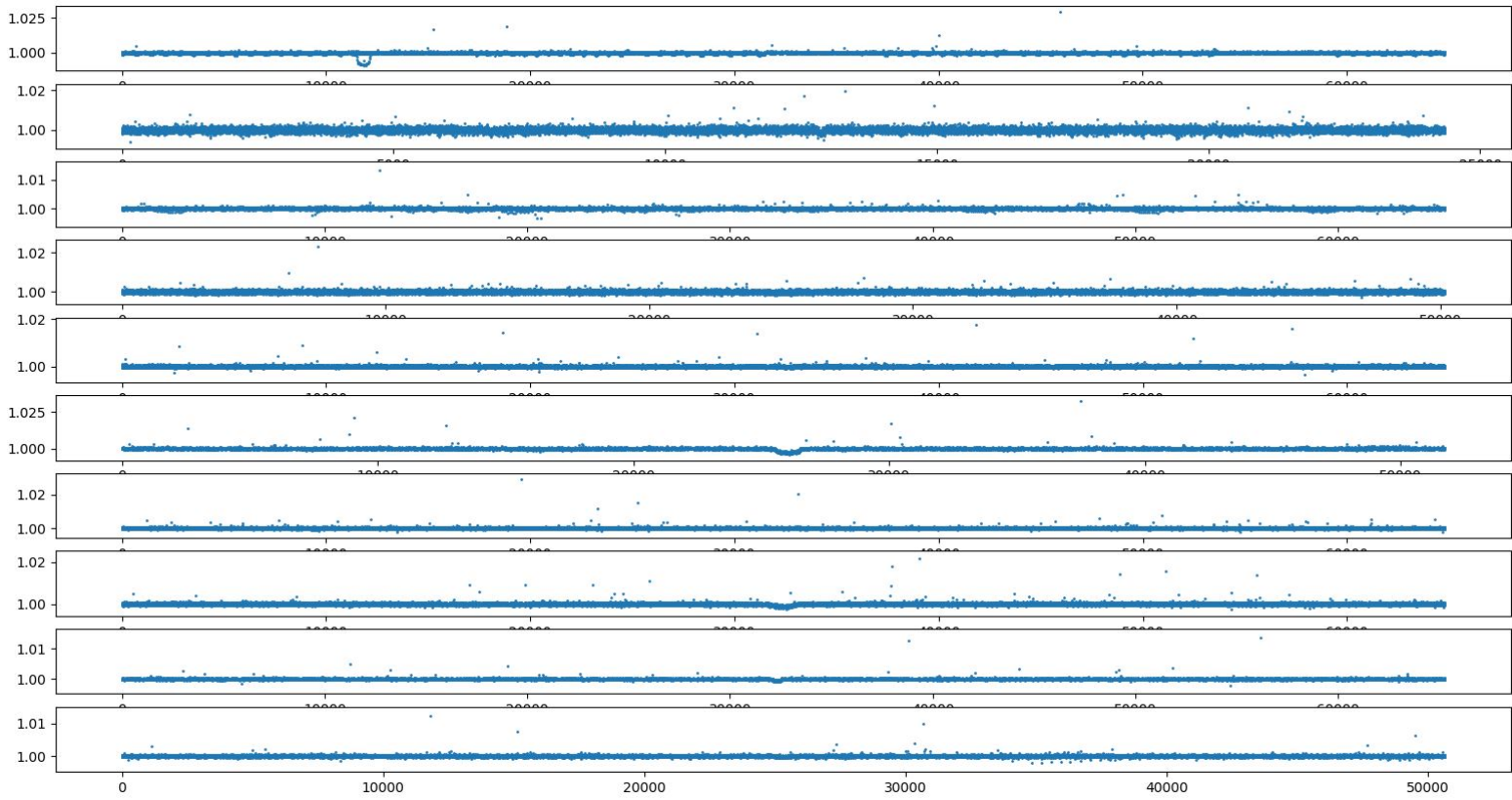

Figure D.6: Folded Positive Sample Light Curves

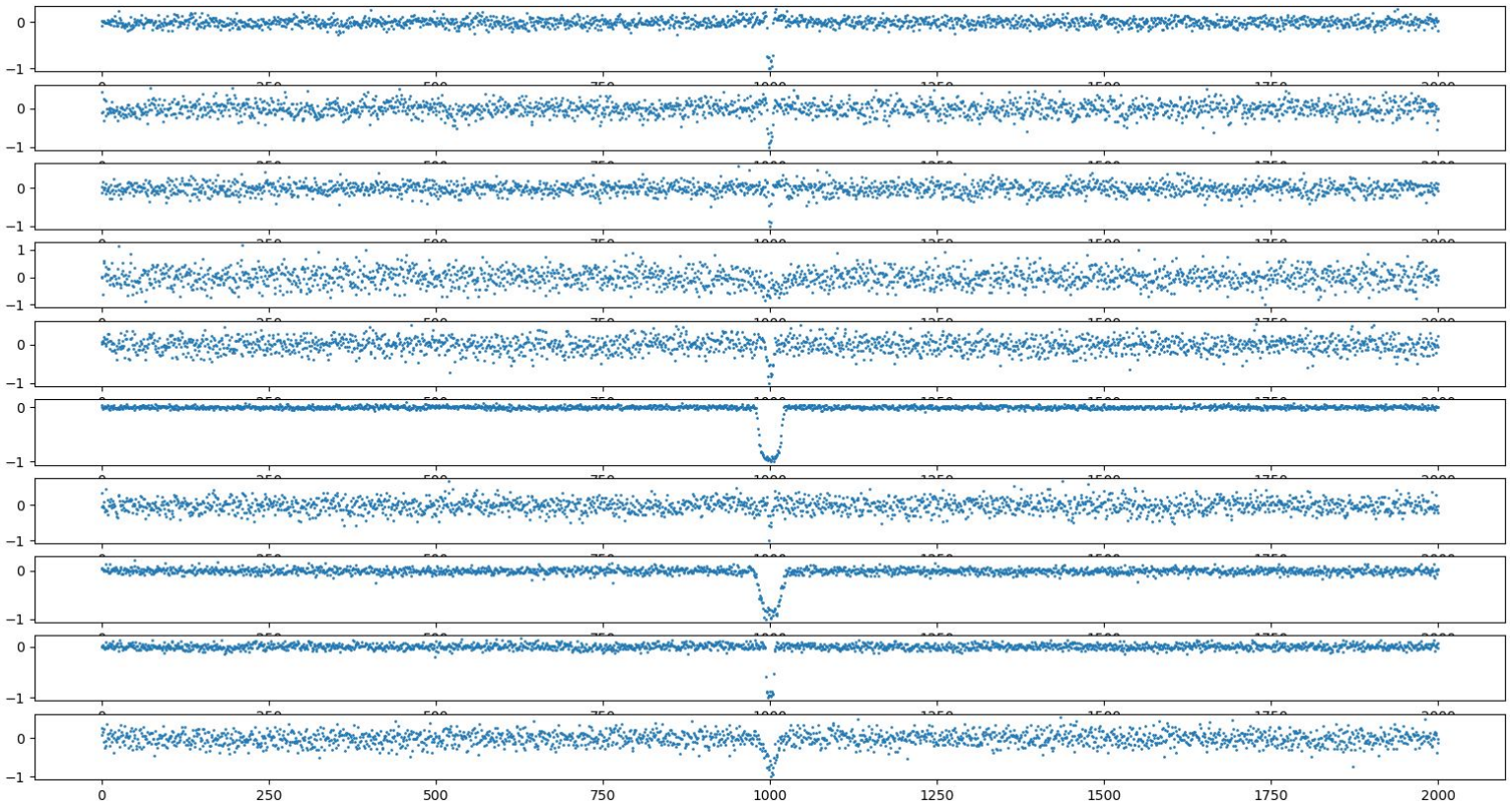

Figure D.7: Binned Positive Sample Light Curves

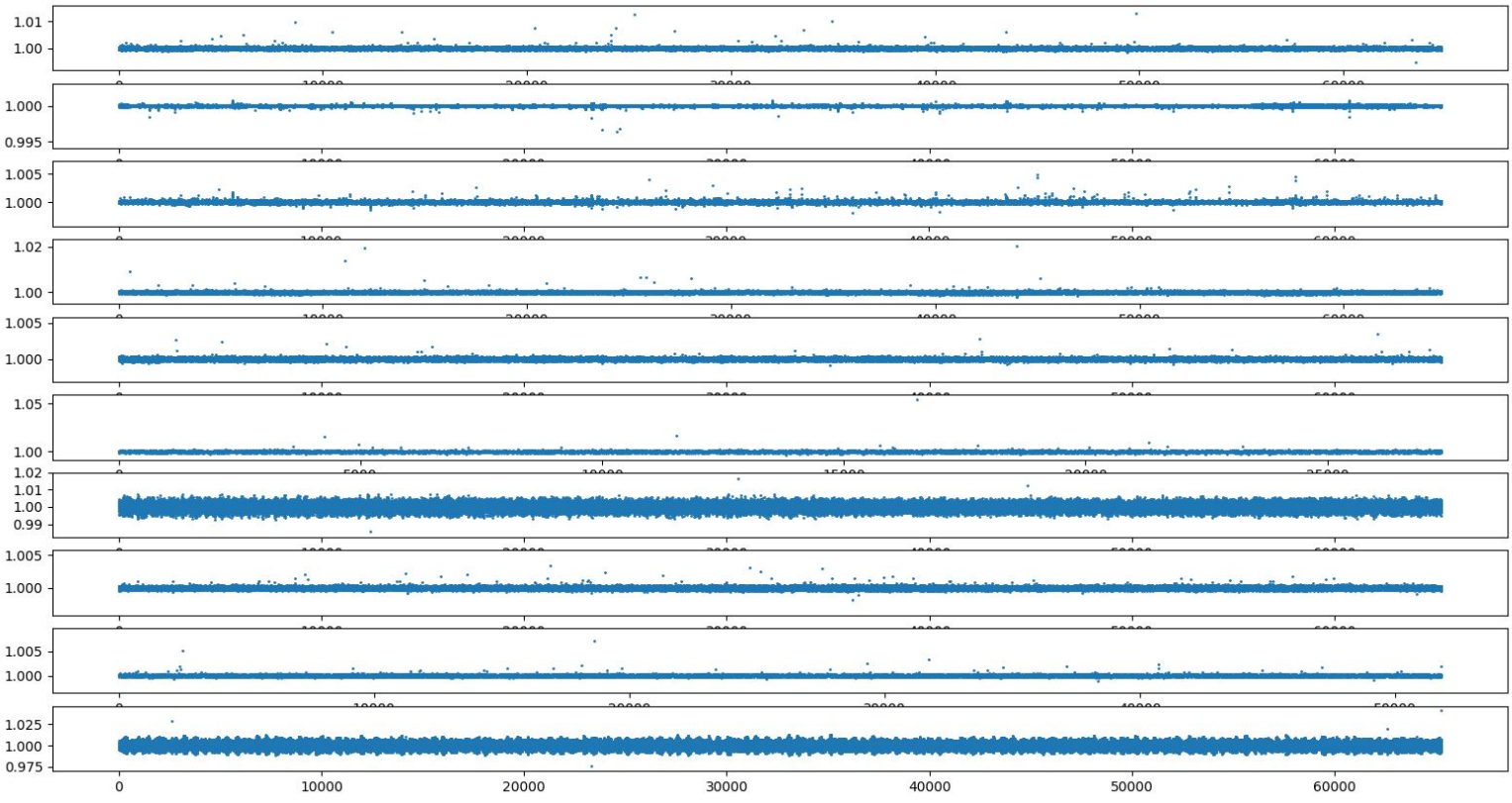

Figure D.8: Flattened False Positive Sample Light Curves

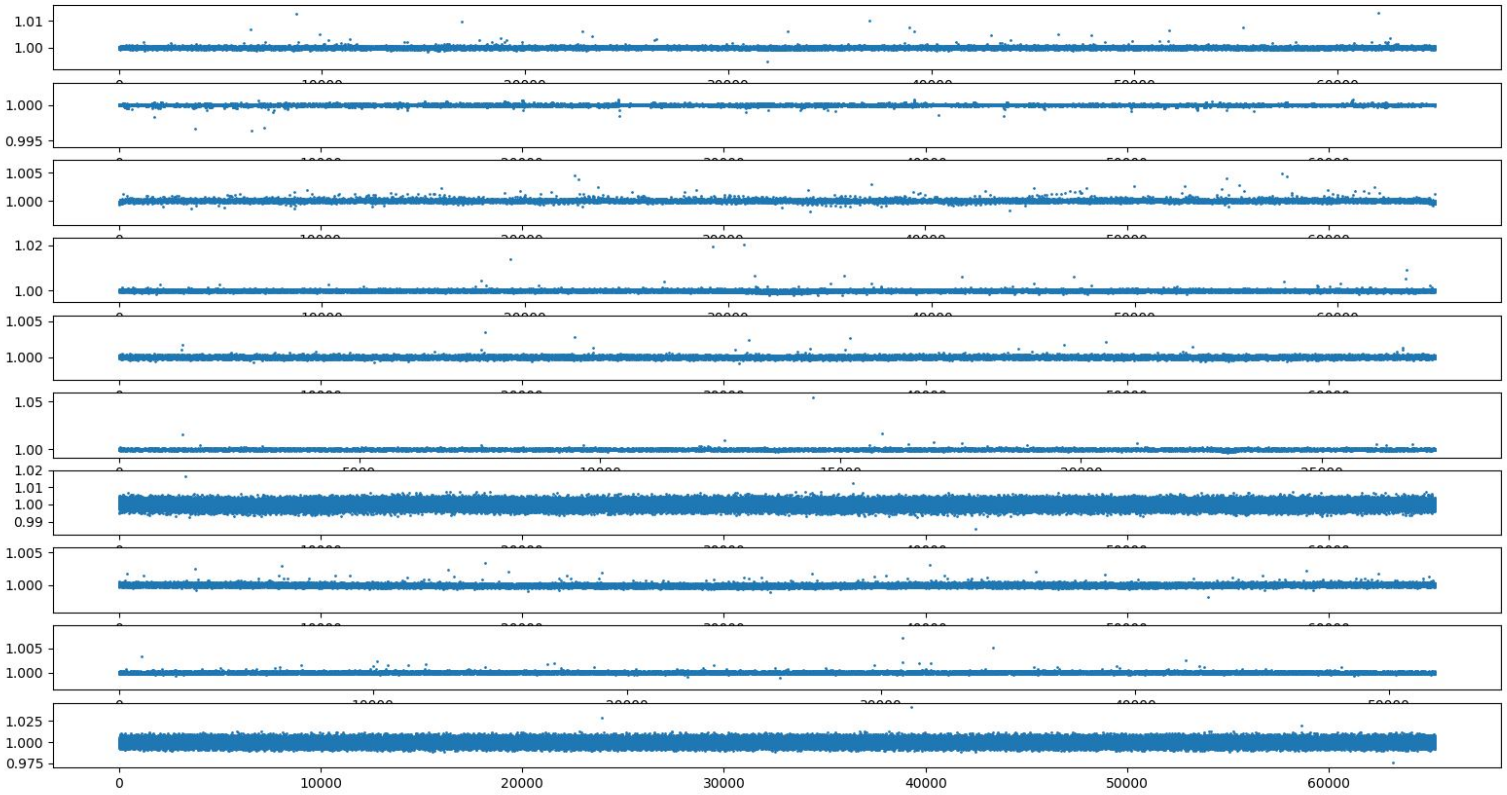

Figure D.9: Folded False Positive Sample Light Curves

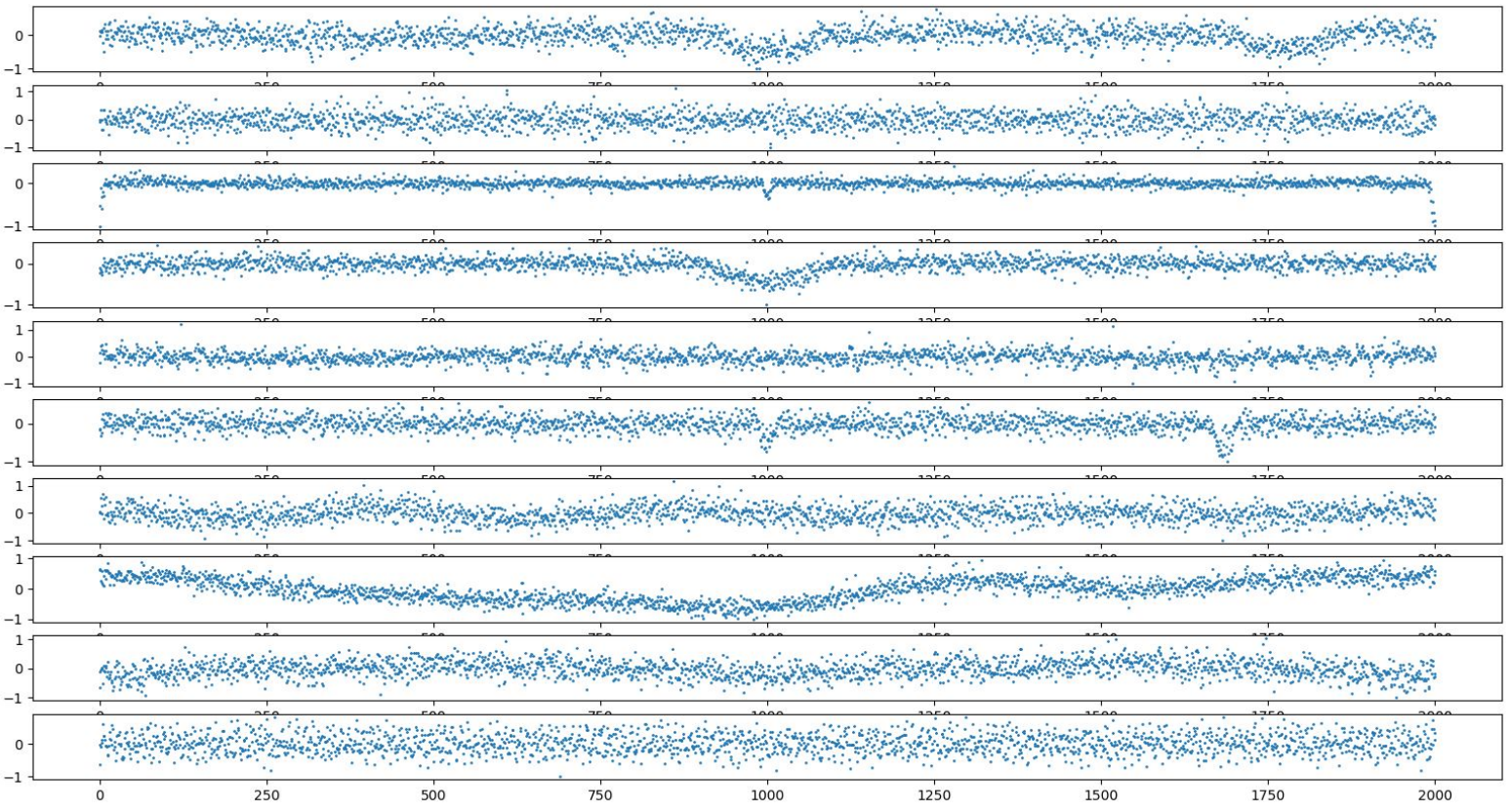

Figure D.10: Binned False Positive Sample Light Curves

Appendix E

Model Architectures

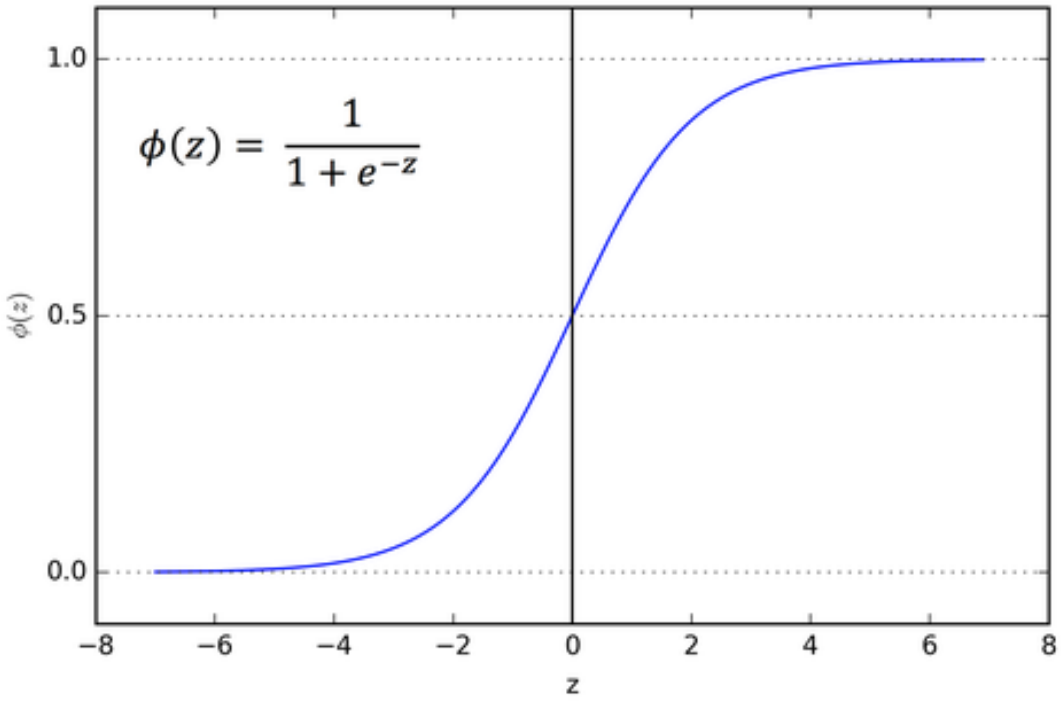

Figure E.1: Sigmoid activation function - https://cdn-images-1.medium.com/max/1600/1*Xu7B5y9gp0iL5ooBj7LtWw.png

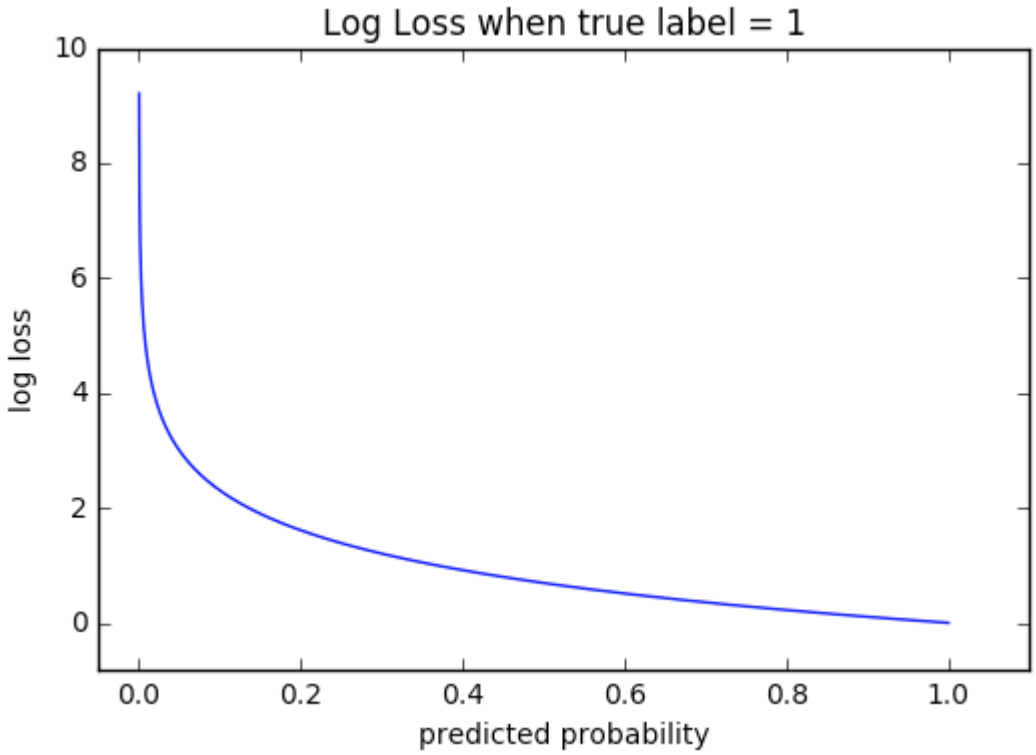

Figure E.2: Cross Entropy Loss $http://ml-cheatsheet.readthedocs.io/en/latest/images/cross_entropy.png$

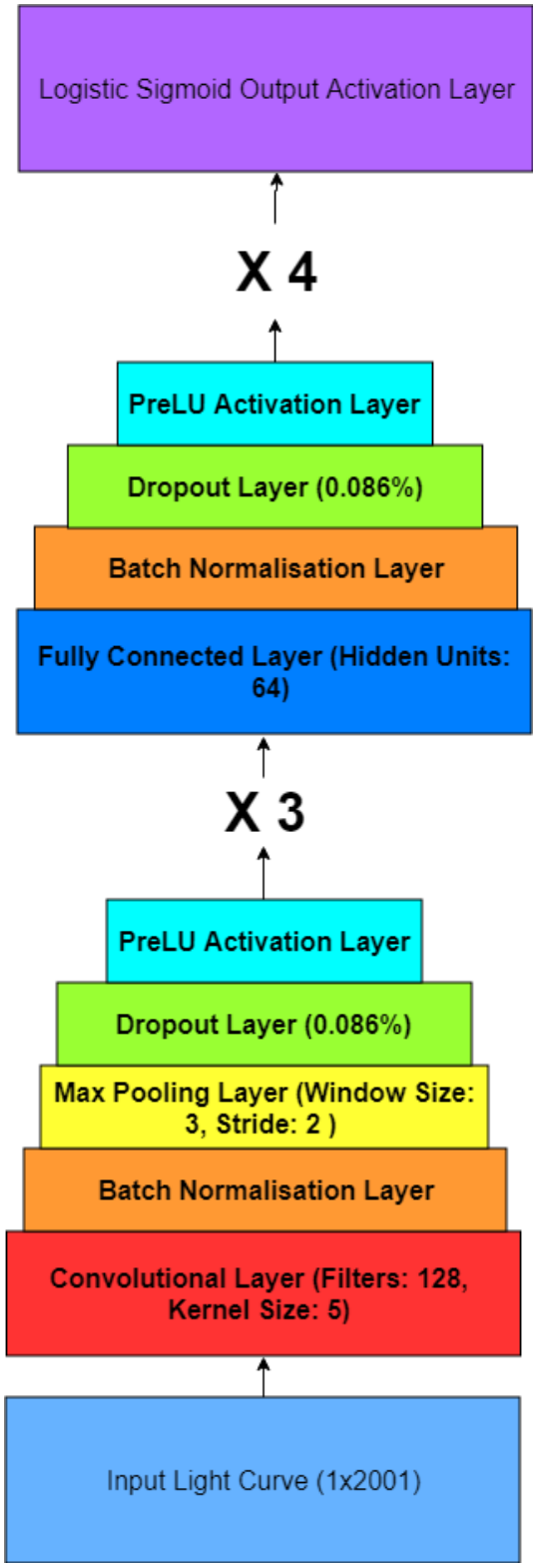

Figure E.3: Diagram showing the topology of the best CNN configuration.

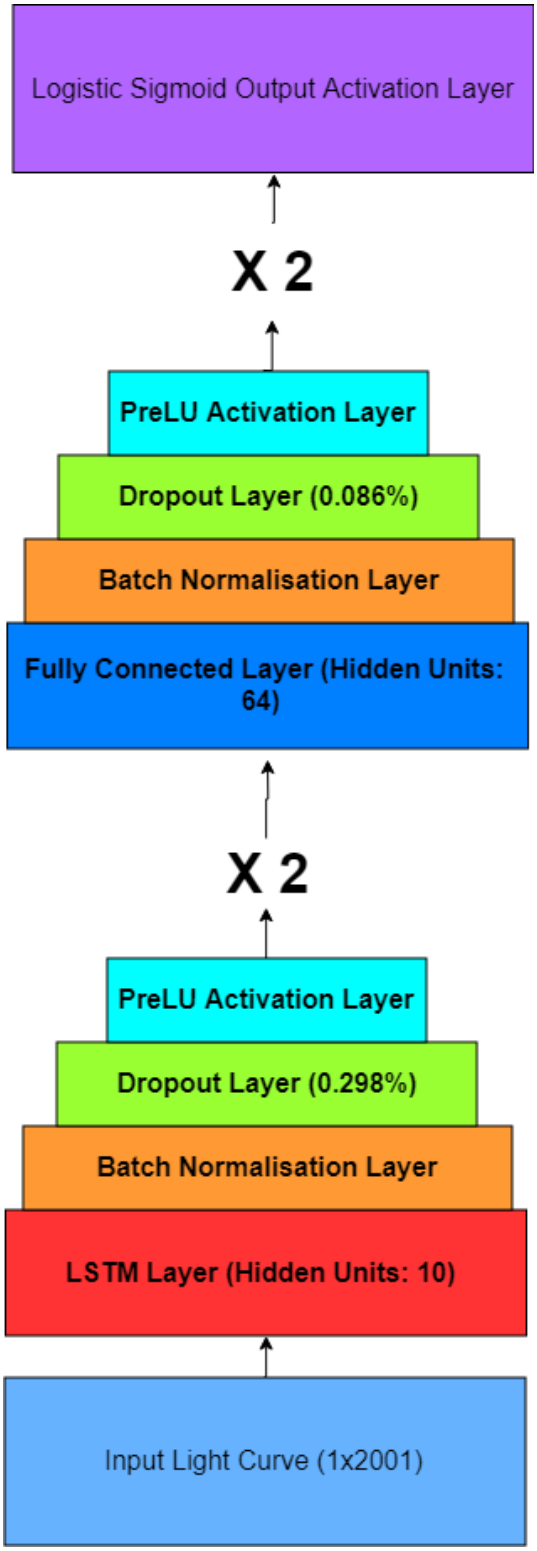

Figure E.4: Diagram showing the topology of the best LSTM configuration.

Appendix F

CNN Performance

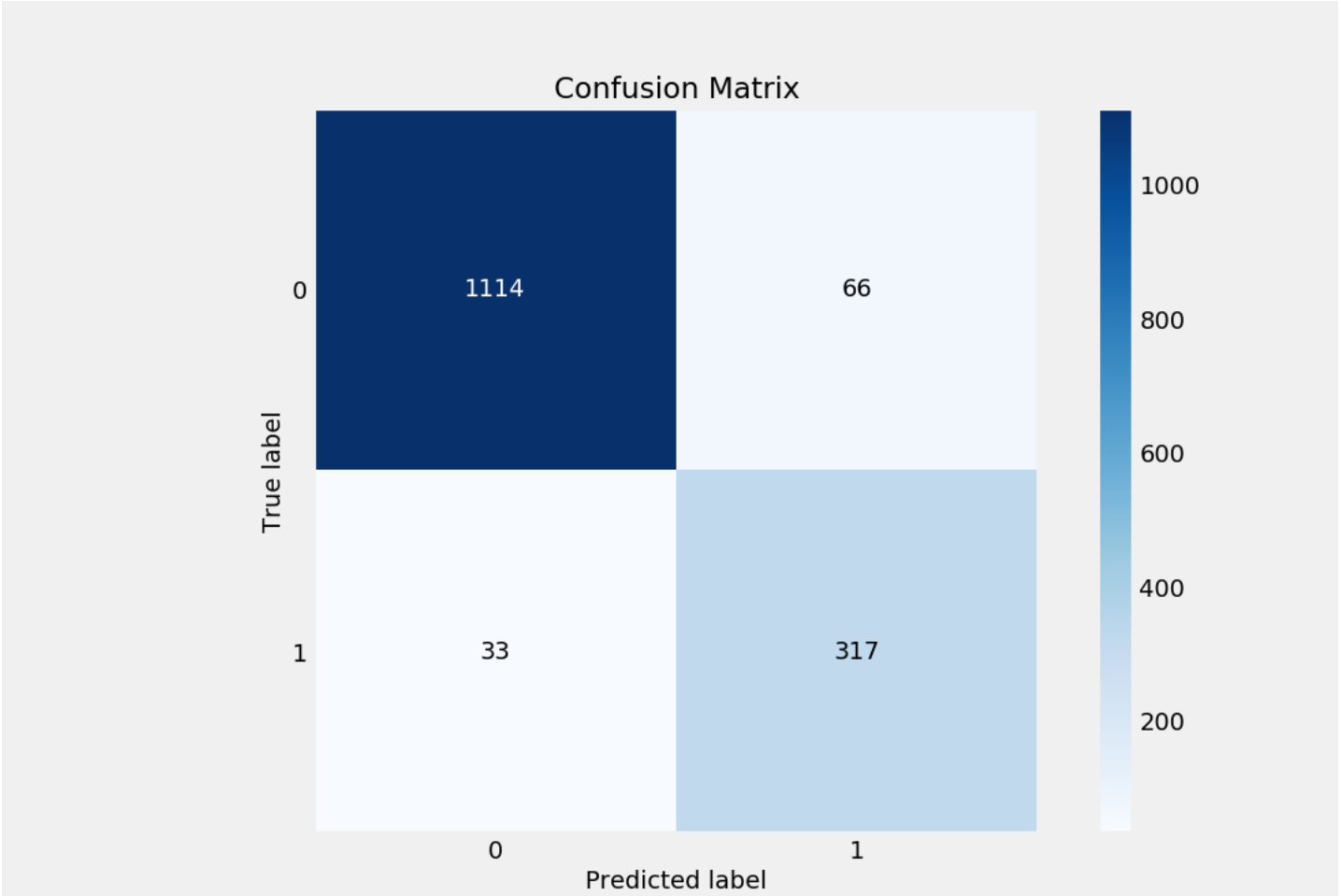

Figure F.1: CNN Confusion Matrix showing number of True Positives, True Negatives, False Positives and False Negatives

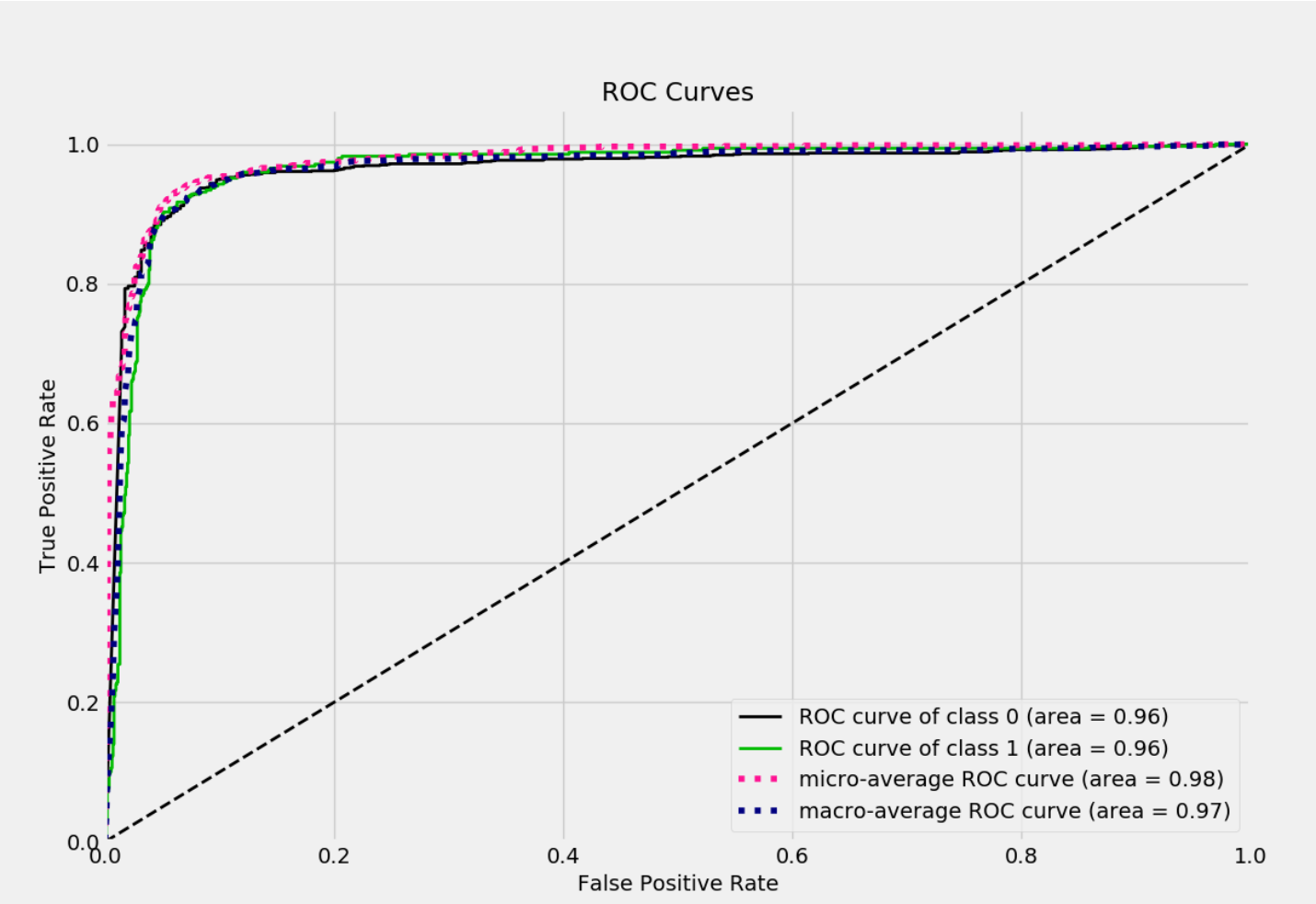

Figure F.2: CNN AUC ROC plot

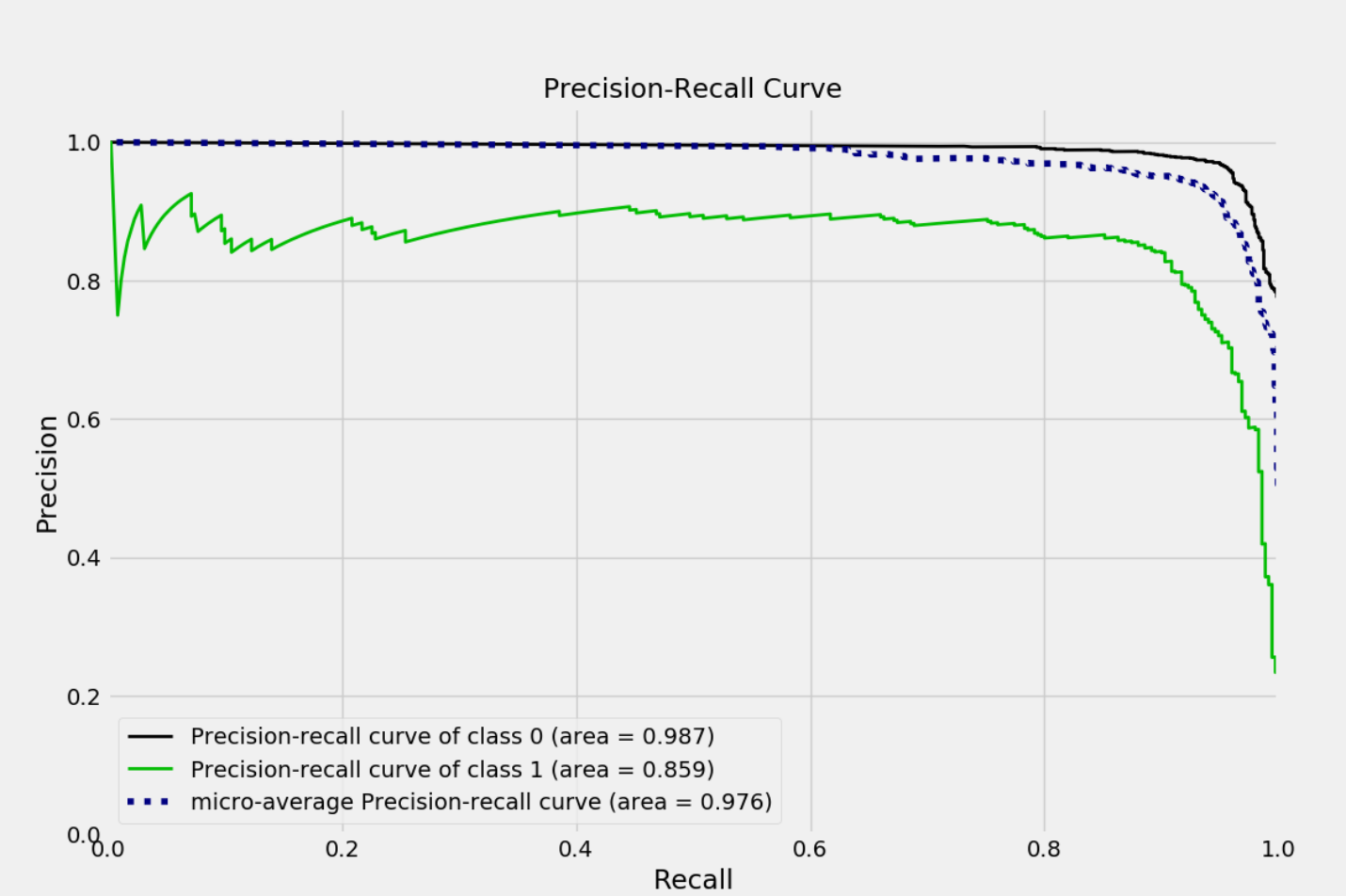

Figure F.3: CNN Precision vs Recall Curve

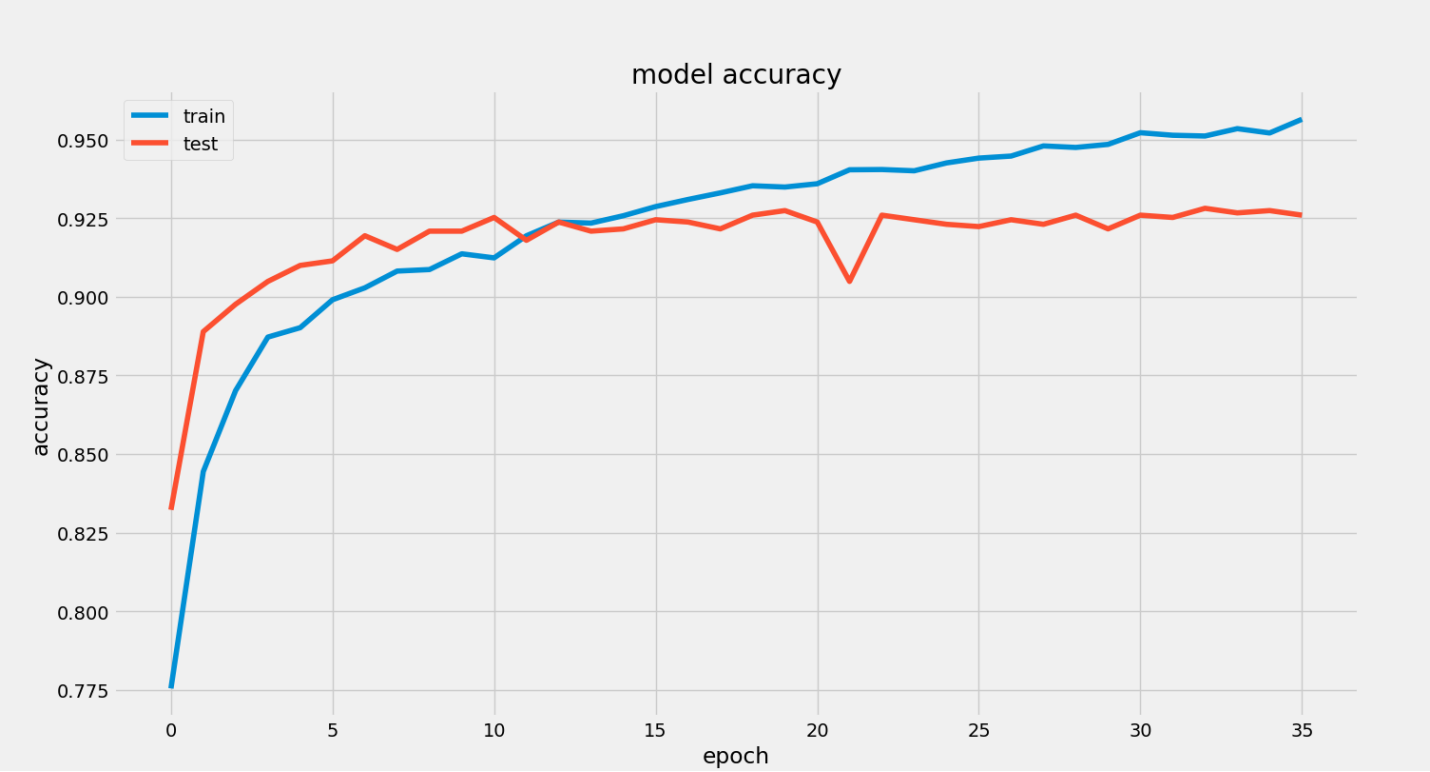

Figure F.4: CNN training and validation fold performance (Accuracy)

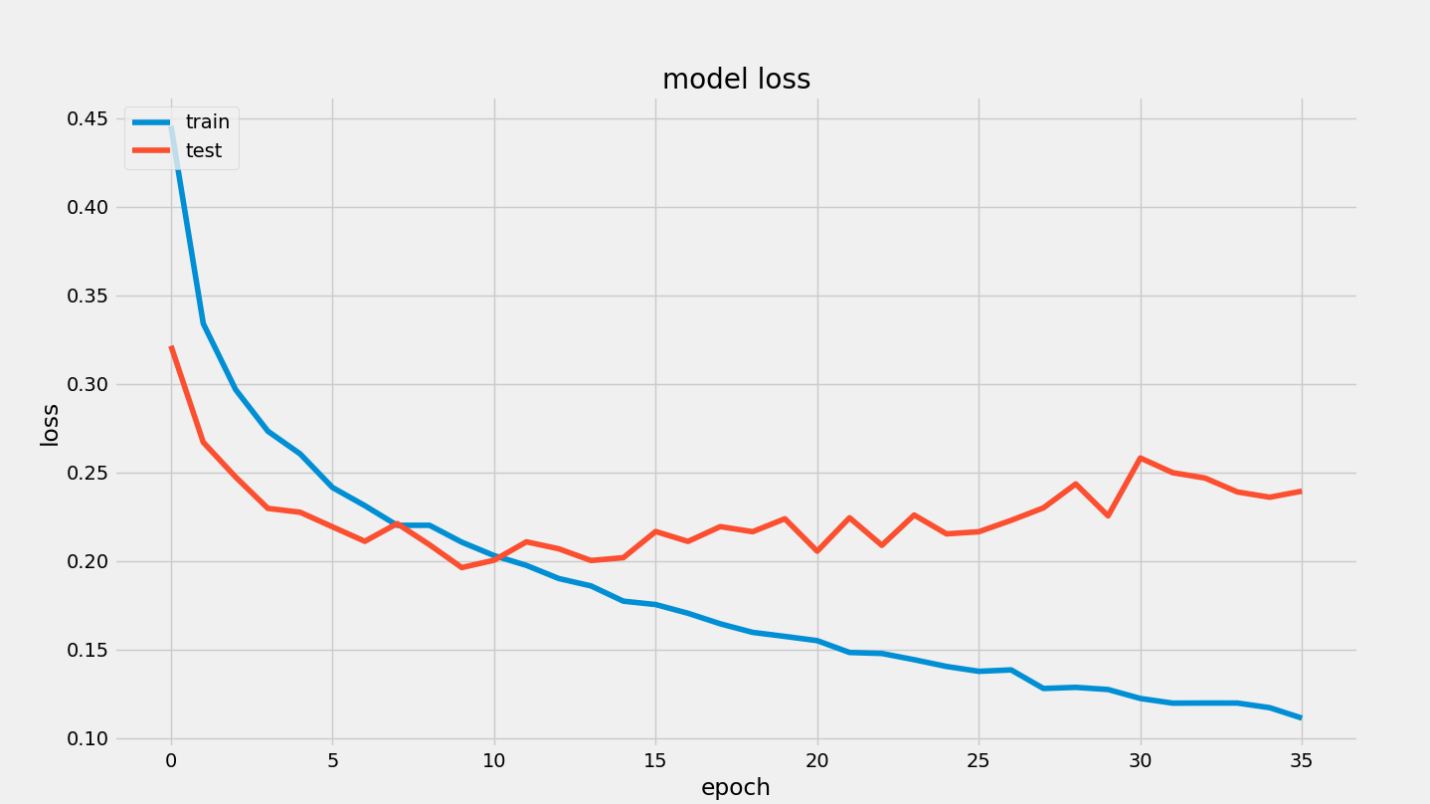

Figure F.5: CNN training and validation fold performance (Cross Entropy Loss)

Appendix G

LSTM Performance

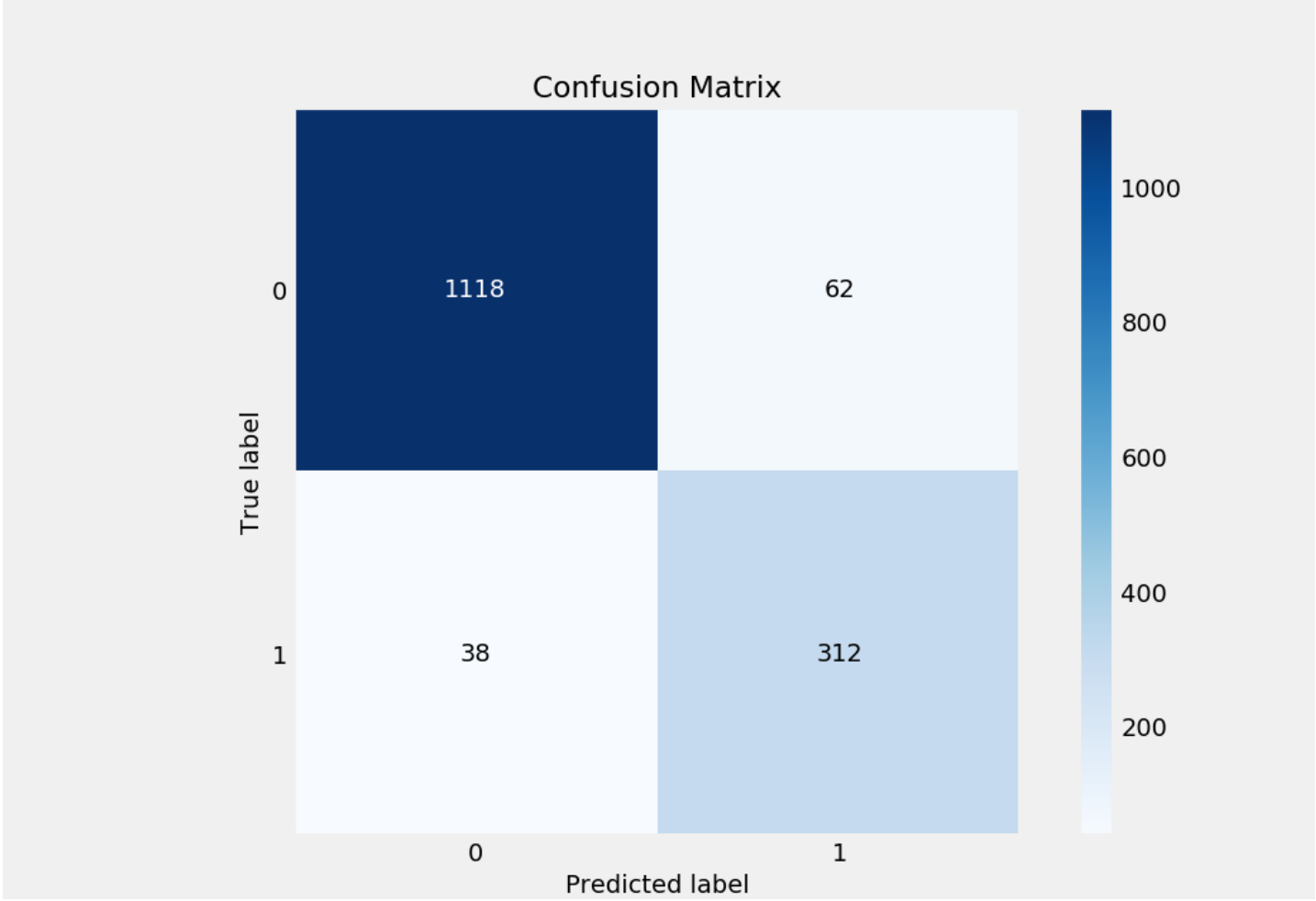

Figure G.1: LSTM Confusion Matrix showing number of True Positives, True Negatives, False Positives and False Negatives

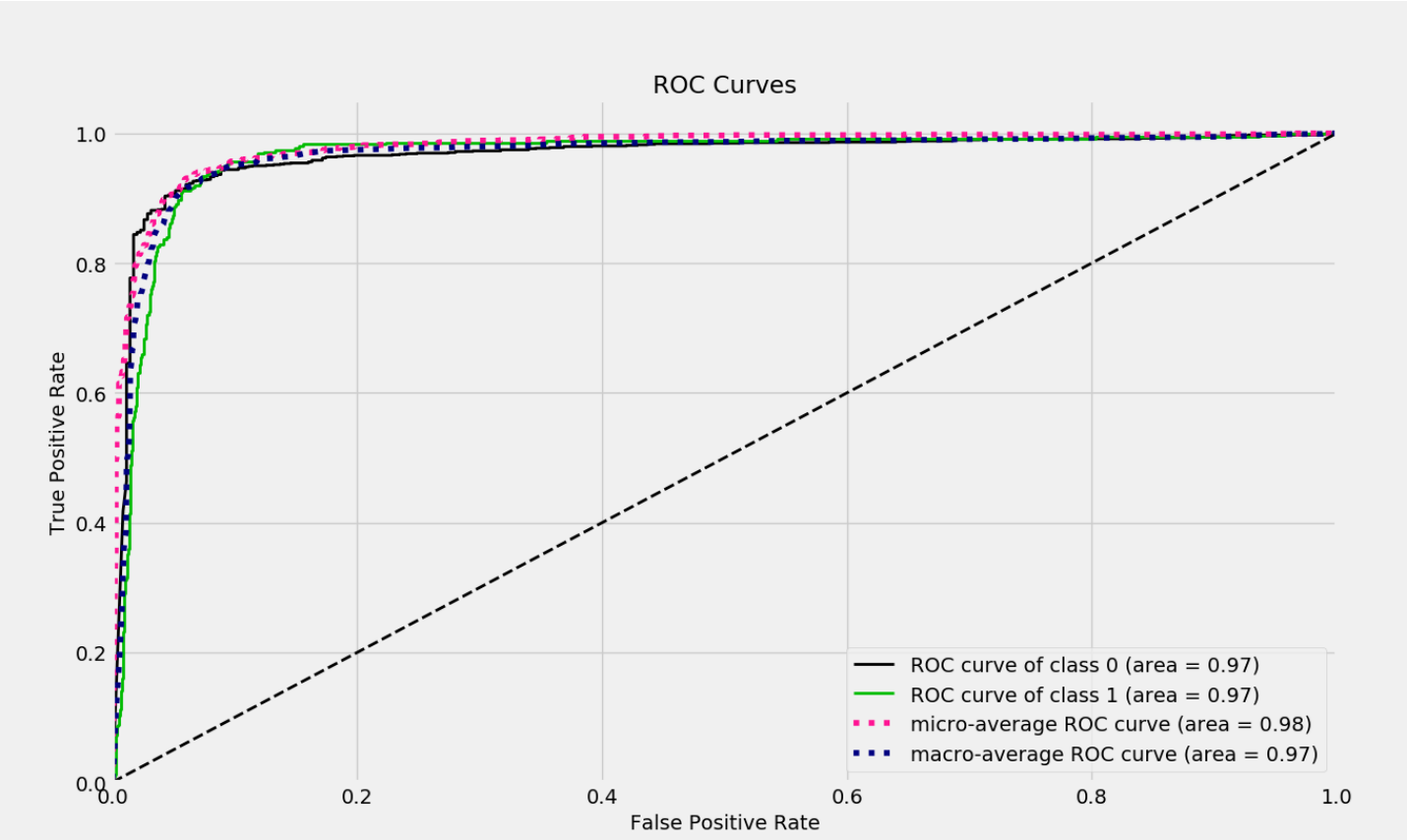

Figure G.2: LSTM AUC ROC plot

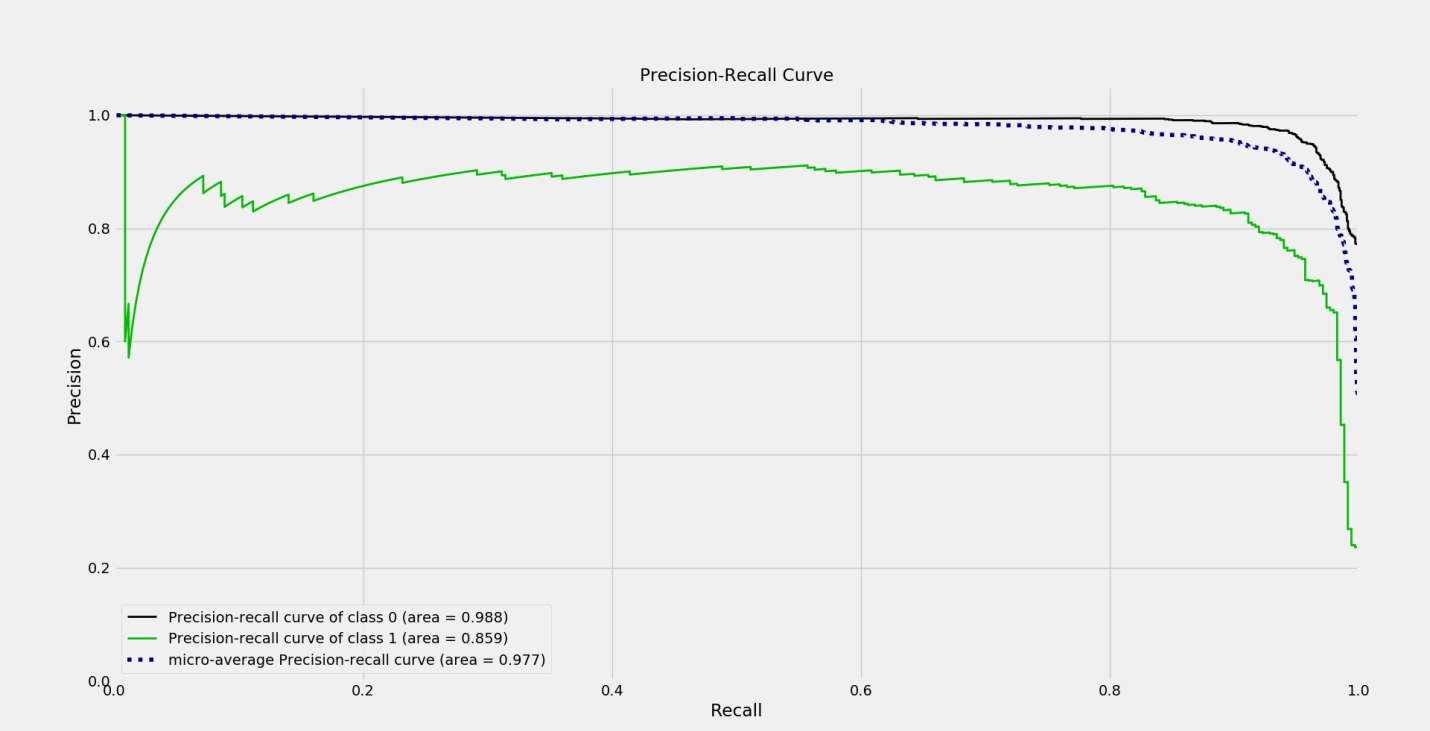

Figure G.3: LSTM Precision vs Recall Curve

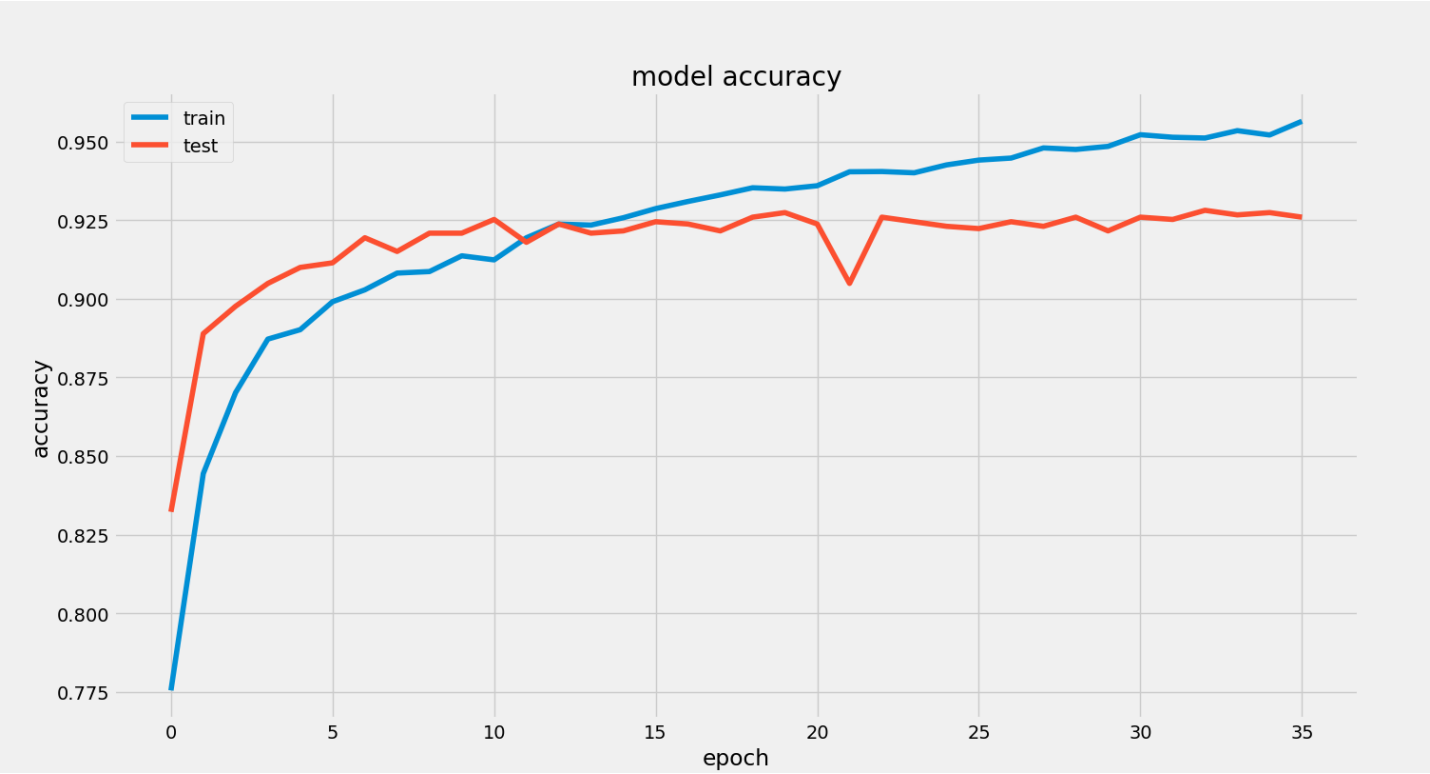

Figure G.4: LSTM training and validation fold performance (Cross Entropy Loss)

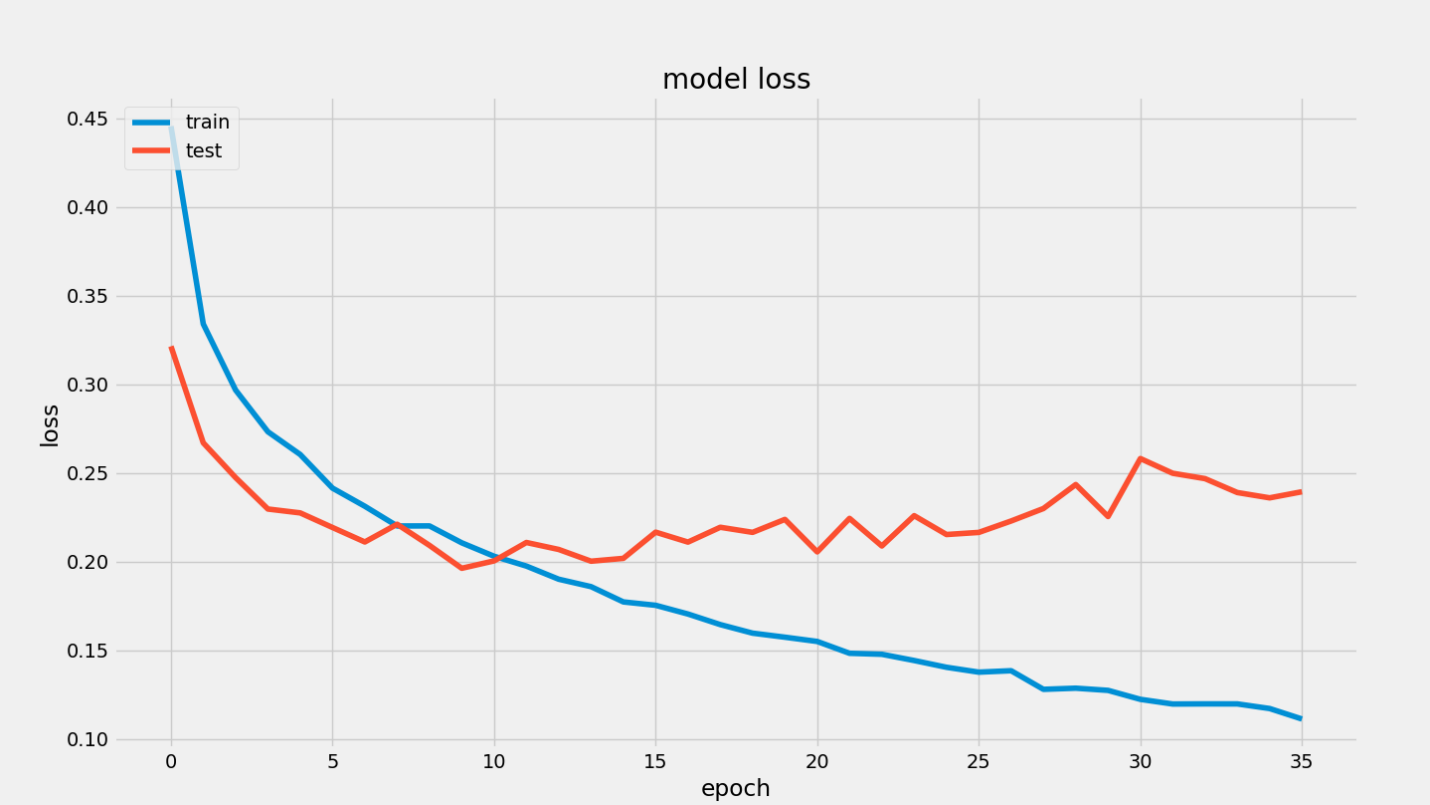

Figure G.5: LSTM training and validation fold performance (Cross Entropy Loss)

Appendix H

Project Management

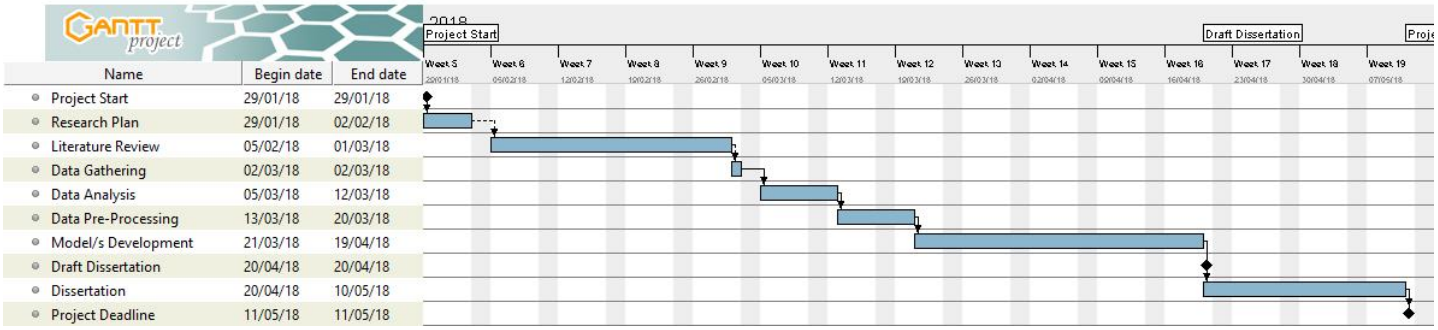

Figure H.1: Initial Project Gantt Chart

Appendix I

Tools and Frameworks

This is a list of the tools used throughout the thesis project.

- Numpy (Operations on data structures)
- Pandas (Dataframe manipulation)
- Scipy
- Matplotlib (Plotting)
- Skplot (Plotting)
- Sci-kit learn (Cross Validation)
- Keras (Deep Learning Library)
- Mendeley (Academic Paper and Reference Management)

Bibliography

[1] R. L. e. a. Akeson. The NASA Exoplanet Archive: Data and Tools for Exoplanet Research. pages 1–23, 2013.

[2] T. e. a. Barclay. A sub- Mercury- sized exoplanet.

[3] Y. Bengio, P. Simard, and P. Frasconi. Learning Long-Term Dependencies with Gradient Descent is Difficult, 1994.

[4] J. Bergstra, D. Yamins, and D. D. Cox. Making a Science of Model Search. pages 1–11, 2012.

[5] G. e. a. Cabrera-Vives. Deep-HiTS: Rotation Invariant Convolutional Neural Network for Transient Detection. *The Astrophysical Journal*, 836(1):1–7, 2017.

[6] H. Cao, X. L. Li, D. Y. K. Woon, and S. K. Ng. Integrated oversampling for imbalanced time series classification. *IEEE Transactions on Knowledge and Data Engineering*, 25(12):2809–2822, 2013.

[7] J. H. Catanzarite. Autovetter Planet Candidate Catalog for Q1-Q17 Data Release 24. *Ksci-19091-001*, (July), 2015.

[8] T. Charnock and A. Moss. Deep Recurrent Neural Networks for Supernovae Classification. 2016.

[9] J. L. e. a. Christiansen. The Derivation, Properties, and Value of Kepler's Combined Differential Photometric Precision. *Publications of the Astronomical Society of the Pacific*, 124(922):1279–1287, 2012.

[10] J. L. Coughlin. Planet Detection Metrics: Robovetter Completeness and Effectiveness for Data Release 25. pages 1–22, 2017.

[11] D. George, H. Shen, and E. A. Huerta. Glitch Classification and Clustering for LIGO with Deep Transfer Learning.

[12] J. e. a. Gu. Recent Advances in Convolutional Neural Networks. pages 1–38, 2015.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *Proceedings of the IEEE International Conference on Computer Vision*, 2015 Inter:1026–1034, 2015.

[14] S. Hochreiter and J. Urgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[15] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. 2015.

[16] J. M. Jenkins, D. A. Caldwell, H. Chandrasekaran, J. D. Twicken, S. T. Bryson, E. V. Quintana, B. D. Clarke, J. Li, C. Allen, P. Tenenbaum, H. Wu, T. C. Klaus, C. K. Middour, M. T. Cote, S. McCauliff, F. R. Girouard, J. P. Gunter, B. Wohler, J. Sommers, J. R. Hall, A. K. Uddin, M. S. Wu, P. A. Bhavsar, J. Van Cleve, D. Pletcher, J. A. Dotson, M. R. Haas, R. L. Gilliland, D. G. Koch, and W. Borucki. Overview of the kepler science processing pipeline. *Astrophysical Journal Letters*, 713(2 PART 2), 2010.

[17] J. M. e. a. Jenkins. DISCOVERY AND VALIDATION OF Kepler-452b: A 1.6 R SUPER EARTH EXOPLANET IN THE HABITABLE ZONE OF A G2 STAR. *The Astronomical Journal*, 150(2):56, jul 2015.

[18] F. Karim, S. Majumdar, H. Darabi, and S. Chen. LSTM Fully Convolutional Networks for Time Series Classification. pages 1–7, 2017.

[19] D. G. e. a. Koch. Kepler Mission Design, Realized Photometric Performance, and Early Science David. pages 1–16.

[20] J. Kremer and S.-S. et al. Big Universe, Big Data: Machine Learning and Image Analysis for Astronomy. *IEEE Intelligent Systems*, 32(2):16–22, 2017.

[21] Y. A. LeCun, L. Bottou, G. B. Orr, and K. R. Müller. Efficient backprop. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7700 LECTURE NO:9–48, 2012.

[22] X. Li, W. Yu, and X. Fan. A Method Of Detecting Gravitational Wave Based On Time-frequency Analysis And Convolutional Neural Networks. pages 1–9, 2017.

[23] T.-w. Loong. Clinical review Understanding sensitivity and specificity with the right. 327(September), 2003.

[24] K. Mandel and E. Agol. Analytic Lightcurves for Planetary Transit Searches. 2002.

[25] K. A. Pearson, L. Palafox, and C. A. Griffith. Searching for Exoplanets using Artificial Intelligence. 15(October):1–15, 2017.

[26] G. R. e. a. Ricker. The Transiting Exoplanet Survey Satellite. 2014.

[27] J. E. e. a. Rodriguez. A System of Three Super Earths Transiting the Late K-Dwarf GJ 9827 at Thirty Parsecs. pages 1–9, 2017.

[28] E. Sadeh. Societal Impacts of the Apollo Program. *Evaluation*, 1051:1–69.

[29] J. R. e. a. Schmitt. A Search for Lost Planets in the Kepler Multi-planet Systems and the Discovery of the Long-period, Neptune-sized Exoplanet Kepler-150 f. 2017.

[30] C. J. Shallue and A. Vanderburg. Identifying Exoplanets with Deep Learning: A Five Planet Resonant Chain around Kepler-80 and an Eighth Planet around Kepler-90. 2017.

[31] M. C. e. a. Stumpe. Kepler Presearch Data Conditioning IArchitecture and Algorithms for Error Correction in Kepler Light Curves. *Publications of the Astronomical Society of the Pacific*, 124(919):985–999, 2012.

[32] C. Tallec and Y. Ollivier. Unbiasing Truncated Backpropagation Through Time. pages 1–13, 2017.

[33] S. E. e. a. Thompson. Planetary Candidates Observed by Kepler. VIII. A Fully Automated Catalog With Measured Completeness and Reliability Based on Data Release 25. 2017.

[34] Yann LeCun et al. Gradient-Based Learning Applied to Document Recognition.