2021/2022

Discovery 2: An analysis of Machine Learning methods to predict exoplanet candidates

Technical Report

Jonathan Flanagan (x18143890)
X18143890@student.ncirl.ie

Contents

Executive Summary

This project examines the performance of Convolutional Neural Networks and Capsule Networks in the problem domain of classifying exoplanet candidates using light fluctuation readings from NASA telescope data. NASA telescopes record light intensity readings from observed stars, if a planet is orbiting one of these stars on the same visual plane as the telescope, an identifiable dip in light intensity is created. This type of reading is known as a transit event and can take different forms depending on the number and size of planets orbiting a particular star. This method of detection makes up 76.82% of the 5,030 exoplanets discovered.

Convolutional Neural Networks were identified as the state of the art machine learning model used for this type of classification problem, and although Capsule Networks have primarily been designed for computer vision tasks and are a relatively new concept, the project aimed to test their ability in this problem domain.

Data retrieved on confirmed exoplanets as well as false positive candidates from the Mikulski Archive for Space Telescopes was used in the analysis and the results showed the Capsule Network performed better in terms of accuracy with 0.96 compared to the Convolutional Neural Network at 0.95. The Capsule Network also performed better in precision with a 1.0 score while the Convolutional Neural Network returned 0.91. The Convolutional Neural Network scored better in recall though with 0.90 versus the Capsule Network at 0.85.

Comparing AUC scores, the Convolutional Neural Network scored best with an AUC of 0.937 while the Capsule Network scored 0.928. Conversely, comparing F1 scores of the models showed that Capsule Network scored best with 0.92 versus the Convolutional Neural Networks score of 0.90. A McNemar's statistical test returned a p-value of 0.185 concluding that there was no statistical difference between the proportion of errors between both models.

1.0 Introduction

1.1. Background

Space exploration has always been at the forefront in displaying the capabilities of the human race and has proven to be invaluable to our lives here on Earth. Some inventions used in our everyday lives started as projects to accommodate space travel. NASA Jet propulsion labs (JPL) worked on making cameras small enough to fit on space craft in the 1990's and as a result 1/3 of all cameras now contain the technology invented, enabling things like the camera phone. The JPL also worked on digital imaging technologies which in turn facilitated the invention of the CAT scan (NASA, 2016). These inventions are only to name a few.

When big data is mentioned, generally the first thing to come to mind is the internet and the massive amounts of data created every day that can be analysed to gather knowledge about human behaviour, such as risk assessment, social media post interactions and general consumer behaviour. On starting this project, I wanted to be able to explore data that didn't stem from a business perspective and was founded in personal interests. I have a wide variety of hobbies with amateur astrophotography being one of them and the mention of the upcoming James Webb telescope at the time of this project's inception excited me.

On researching more about the James Webb telescope I found that data was publicly available for other NASA missions such as Kepler and TESS , this along with NASAs recently discovered 301 new exoplanets using its ExoMiner Deep Learning model (Valizadegan, et al., 2021) gave me the idea that I wanted to put what I had learned over the past four years to the test and explore these data sets using machine learning.

These new discoveries by NASA and the launch of the James Webb telescope paired with the rise in research possibilities for machine learning applications is what eventually brought me to my project idea.

The idea set out to answer the question: When it comes to machine learning techniques used for searching for exoplanets, can newer implementations such as Capsule Networks outperform more mature established implementations such as Convolutional Neural Networks.

Although Capsule Networks have so far been used mainly for computer vision tasks could this type of neural network be adjusted to find exoplanets as effectively as Convolutional Neural Networks have been to date.

1.2. Aims

The project sets out to analyse the performance of Capsule Networks compared to Convolutional Neural Networks in the classification of exoplanet candidates. Given that Capsule Networks have displayed performance improvements over Convolutional Networks in computer vision tasks such as object classification, the aim of this project is to apply a Capsule Network to a task in a domain outside of computer vision where Convolutional Neural Networks have been used with success, testing their relative performance. The null hypothesis in this case is that Capsule Networks display a statistically significant improvement.

To accomplish this, several smaller aims need to be met, first data on confirmed exoplanets and confirmed false positives need to be sourced. This data is publicly available through the Mikulski Archive for Space Telescopes (MAST).

Using the star designations from the MAST datasets, the mission database can be queried to retrieve the light curve information for each star where it can be processed and paired with its official classification of either a confirmed planet or a false positive. Once the data is paired it can be processed and labelled to be used as train and test data for the machine learning models.

Two machine learning models will then need to be created and calibrated on the same training data, then tested on test data measuring their performance in accuracy, precision, recall, F1 score, and AUC. The same train and test data is used on each model in line with the assumptions of McNemar's test which will be used to assess the level of disagreement between the two models.

The Convolutional Neural Network is to be built with the same architecture as presented in the paper "*Identifying Exoplanets with Deep Learning: A Five-planet Resonant Chain around Kepler-80 and an Eighth Planet around Kepler-90*" (Vanderburg & Shallue, 2018) as from other papers studied this particular architecture has shown the most success with 95.4% Accuracy and a 98.5 AUC.

For the Capsule Network, it was originally discussed and presented in terms of computer vision using three dimensional input (Hinton, et al., 2017). Following the principles and the mathematics behind its original implementation the Capsule Network will need to be created and adjusted to be able to deal with one dimensional data.

With the performance information of the two models as well as the results of a McNemar's' test a conclusion will be reached on whether the null hypothesis is failed to be rejected or not. A critical review and recommendations on each model is also to be presented along with any further work and considerations.

1.3. Technology

1.3.1 Programming Language Considerations

The primary language used in this project is Python, with a mix of IDEs under the anaconda framework used throughout the project; including Juptyer notebook at the exploratory stage and Spyder for building the scripts necessary to retrieve the data and build the models for analysis.

The R programming language was considered as part of the project proposal but not used. During the beginning research stage of the project to determine the best suited technologies, the two programming languages that stood out were Python and R. After researching the available packages that would possibly be suitable to the project and attempting smaller projects from "*Comparative Approaches to Using R and Python for Statistical Data Analysis*" (Sarmento & Costa, 2017) , Python was subjectively the easier language to use along with easier to understand IDE's (Spyder IDE and Jupyter notebook). While the documentation for R is extensive the documentation for Python packages such as Pandas, Numpy, TensorFlow, Keras and Scikit Learn are more accessible and easier to follow. Books like "*Hands-On Convolutional Neural Networks with TensorFlow*" (Zafar, et al., 2018) are also readily available and a useful resource.

Other deciding factors for the choice of Python over R was the availability in each language of the Lightkurve library for data retrieval and initial processing. The LightKurve package is built on top of a Python library called AstroPy and specifically created for the initial processing of the type of data to be used. For the building of the custom Capsule Network, TensorFlow in Python also suited the project objectives better than R.

1.3.2 Data Storage Solution Considerations

A possible database solution was investigated for storing data after the retrieval process, but as all the data would be contained to one table after pre-processing it was decided that a database added un-necessary complexity to the project without adding any value to the analysis and goals of the project, CSV file format was chosen instead.

1.3.3 Data Mining Considerations

In the proposal stages of the project, it was considered that for the Data mining implementations of the project the model training and testing would be carried out both locally and using cloud processing, Saturn Cloud was chosen at this stage as the prime candidate due to the available processing power on its free student tier.

In setting up the technologies needed to conduct the testing locally, additional drivers from the cudNN library as well as software from NVIDIA's CUDA toolkit were installed on the local machine to enable the use of GPU processing when training and testing the models. After initial testing the cloud solution was deemed unnecessary due to the performance afforded by the cudNN drivers and CUDA toolkit software using the local GPU.

Model Selection

Convolutional Neural Networks

The first choice during model selection was a Convolutional Neural Network (CNN) as its implementation has demonstrated ground-breaking results in numerous fields where pattern recognition is applied and can be considered the state of the art measure. (Albawi, et al., 2017). In context of the problem domain, research in exoplanet classification, they have shown high accuracy and precision is achievable, for example "*97% average precision and 92% accuracy on planets in two-class model*" (Osborn, et al., 2019) was achievable in one research paper found. 95.4% Accuracy and a 98.5 AUC in another (Vanderburg & Shallue, 2018). NASA itself implements a Deep Learning Network called ExoMiner which recently discovered 301 additional exoplanets (Valizadegan, et al., 2021) although the details of its architecture are not publicly known. CNNs are a type of Deep Learning where the name is derived from the mathematical term of the linear operation between matrixes known as convolutional. CNNs have multiple layers. The input, convolutional, non-linearity, pooling, and fully connected layers. These, along with the output layer make up the basic architecture of a CNN.

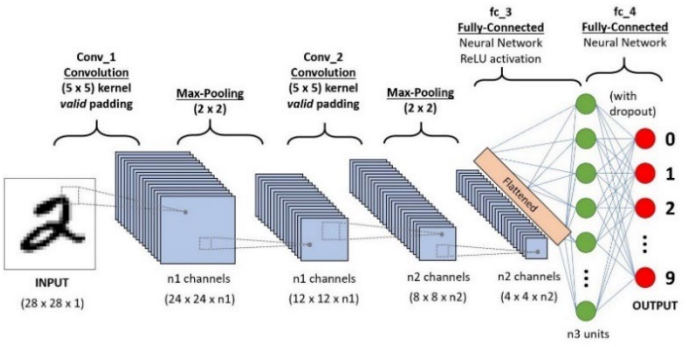


*Figure 1: Diagram of CNN (https://cdn-images-1.medium.com/max/1600/1*uAeANQIOQPqWZnnuH-VEyw.jpeg)*

CNN implementations have many configurations that differ from the diagram above. With the above visualisation as a guide, a basic description of how the CNN returns an output is, non-linearity occurs between the convolution and pooling layers which is used to adjust, through training the model over a certain number of epochs, the generated output at a given a threshold. Example output activation functions that are applied are sigmoid and ReLu.

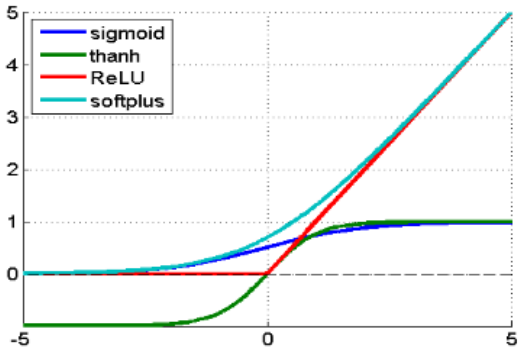


Figure 2: Examples of non-linearity functions (https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8308186)

As this project involves binary classification and the probability of such a classification, the activation method to be used for the output is sigmoid where outputs over a given threshold are classified as true.

Capsule Networks

It is the aim of this project to test the hypothesis that an implementation of a Capsule Network can display improvements on CNNs in classifying exoplanet candidates, as they have displayed improvements in other computer vision tasks involving feature recognition and classification (Hinton, et al., 2017) . Therefore, Capsule Networks were the second selection for models used.

The improvements Capsule Networks display over CNN's is in part due to the difference in how the non-linearity of the network is applied. One key difference between CNNs' and Capsule Networks is that in a CNN a single activation function is used as output, but in Capsule Networks the activation function is based on comparisons between multiple incoming predictions by where a Capsule Network, using convolutional capsules can share knowledge by tying the weights of feature detectors together. This is known as routing by agreement and enables a Capsule Network to subdivide the features of an image into independent structures, regardless of their orientation, and make probabilistic determinations of the whole based on the sum of the features.

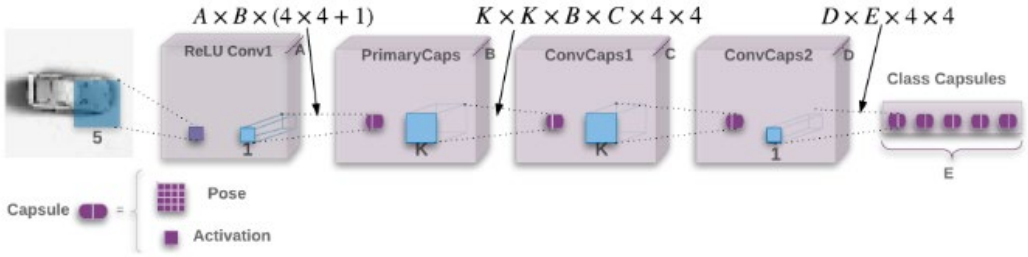


Figure 3 Representation of CapsNet (https://openreview.net/pdf?id=HJWLfGWRb)

1.3.4 Graphical Output Considerations

In the initial stages of the proposal, considerations for graphical output was between Tableau and Power-Bi but as the project moved from proposal to implementation these technologies provided no extra benefit in the analysis to be carried out between two Machine Learning models, and it was decided that any graphical output could be done inline using the Python libraries Matplotlib and Seaborn.

1.3.5 Other Considerations

For other ancillary tasks such as reporting, and documentation Microsoft Word and Excel are used. CSV files for testing are stored locally and can be viewed with excel, and reports are written with Microsoft Word. For version control GitHub is used.

1.3.6 Technology Details

Below is a table of technologies, software packages, Python libraries and system requirements used in the project.

Software Technologies Used	
IDE	Usage
Jupyter notebook	Data Exploration
Spyder	Data Retrieval, Data Processing, Model Building

Non IDE	Usage
cudNN	Drivers for GPU usage of TensorFlow
CUDA toolkit	GPU usage of TensorFlow

Python Libraries & Packages	
Package	Usage
Pandas	Data Manipulation, API connection
Numpy	Data manipulation, feature creation.
TQDM	Progress outputs to console
Time	Sleep timer on http requests
Glob	Directory and file navigation.
OS	Memory Management
Shutil	Memory Management
LightKurve	Data collection from MAST, light curve processing
Matplotlib	Data visualisation and file export
Seaborn	Data visualisation and file export
SciPy	Gaussian filtering
TensorFlow	Model creation and custom layer creation in Capsule Network
Keras	Framework for TensorFlow model building
Scikit-Learn	Data stratification, processing, and model metrics

System Details	
Type	Details
CPU	AMD Ryzen 5 3600x
RAM	128GB @ 3200 MHz
GPU	RTX 2060 Super 16GB (modified)
Operating System	Windows 10 64bit
Connectivity	Up: 123.20 Mbps – Down: 48.58 Mbps

1.4. Structure

Providing a brief overview of the structure of the project document and a description of what is discussed in each section of the report.

Data	Details of the data set used. How it was sourced and retrieved. Exploratory analysis on the initial dataset as well as the basic pre-processing steps involved and how the labelled dataset for classification was created.
Methodology	What methodology was used for the project. How each step of the methodology chosen applied to the project, including a summary of each step such as data transformation and feature selection processes. Describing how the project went from the raw imported data to the labelled classification data to model creation, implementation, output, and evaluation.
Implementation	Describing the models, their implementation and algorithms used in the data mining process along with how they were applied in each model, including any additional processing involved of the data before running each model.
Results	Presenting the results of the model analysis and performance metrics including statistical tests performed, prediction accuracy, recall and precision scores. F1 score, AUC for each model and the results of McNemar's tests.
Conclusion	A summary of the main hypothesis laid out in the project and the results of the analysis along with insights gathered throughout the process and a critical evaluation of the project, its outcomes, and the processes to achieve those outcomes.
Further development or research	Future research that could be carried out that would add substance to the research already carried out in the project if time and resources where available.

2.0 Data

The following section provides an overview of the datasets, the sources, the process of data collection, a description of each dataset, their features, and initial exploratory analysis of the data.

2.1.1 Initial Data Sources and Collection

The initial data sources for this project are from the Kepler telescope mission and are accessed as part of several data products the Mikulski Archive for Space Telescopes (MAST) maintains. Confirmed exoplanets and Kepler objects of interest are provided by MAST and is collected through API requests. The API requests used are predesigned queries supplied in their documentation (Mikilski Archove for Space Telescopes, 2021).

Current exoplanets are retrieved through the table access protocol. (https://exoplanetarchive.ipac.caltech.edu/docs/TAP/usingTAP.html).

While the Kepler Objects of interest are retrieved through the standard API (https://exoplanetarchive.ipac.caltech.edu/docs/program_interfaces.html#koi)

2.1 Initial Data

Details of the initial MAST data download needed for querying the mission information for each star. This data was explored and analysed for possible features that could provide any additional value to the exoplanet classification as well as extracting features for use in downloading mission data through the LightKurve API and labelling of training and test data.

Dataset	Source	Format	Details	Information
Confirmed Exoplanet Listings	MAST API Query	CSV	4884 Rows 373 Columns	Details on the exoplanet such as the star name it orbits, the star mass, brightness, and the orbital period of the confirmed exoplanet
Kepler Objects of Interest	MAST API Query	CSV	9564 Rows 50 Columns	Details on the objects of interest such as the star, the orbital period of the object, disposition, and Kepler ID

2.1.2 Initial Data Details

Confirmed Exoplanet Listings

This dataset contains 373 columns of information, a descriptor from MAST on the contents of each column can be viewed in XML format here. From the 373 initial columns 22 columns are used as part of this project to gather some insights, domain knowledge and possibly use features as part of the dataset for the machine learning models.

Full description of the used columns:

Name	Type	Description
pl_name	Char	The planet name
pl_letter	Char	The planet lettering system (single alphabetical character)
hostname	Char	The host (Star) name for the planet discovered
tic_id	Char	The target identification number, for use downloading the specific Pixel Image data
disc_pubdate	Char	When the discovered planet was published
disc_year	int	The year the planet was discovered
discoverymethod	Char	The method used to discover the planet, Transit Microlensing etc
disc_locale	Char	From where the discovery was made, from the ground or from space
disc_facility	Char	The facility that was in charge of the instrument that made the discovery
disc_instrument	Char	The type of instrument used, For Kepler it is the Kepler CCD Array
disc_telescope	Char	The name of the telescope, or camera that made the discovery
pl_orbper	double	Is the orbital period of the planet discovered
pl_eqt	double	The equilibrium temperature (Kelvin) of the planet discovered
pl_dens	double	The density (g/cm3) of the planet discovered
pl_trandur	double	The transit duration in days
pl_radj	double	The discovered planets Jupiter Radius
pl_rade	double	The discovered planets Earth Radius
pl_bmasse	double	The discovered planets Earth mass (Planet Mass*sin(i)/sin(i))
st_age	double	The stellar age in Gigayear (Gyr)
st_mass	double	The Stellar Mass (Solar Mass)
tran_flag	int	Detected by Transits Flag
sy_dist	double	The system distance in (pc) Parsec approx. 3.26 light-years

Kepler Objects of Interest

This dataset contains 9654 rows and 50 columns, after reading the full documentation it was determined that five columns provided information possibly relevant to the project and were selected for further exploration.

These columns of interest are used to check for data in-balance for the machine learning model as well as basic statistics in regard to what will be outputted for the model classifications. These columns are paired with the confirmed planet data by the Kepler name and Kepler ID to identify targets from the particular Kepler mission and transit detection method used in this project.

Full description of the used columns:

Name	Type	Description
kepid	Int	Target identification number, as listed in the Kepler Input Catalogue (KIC). The KIC was derived from a ground-based imaging survey of the Kepler field conducted prior to launch.
kepler_name	Char	Kepler number name in the form "Kepler-N," plus a lower-case letter,
koi_score	Double	A value between 0 and 1 that indicates the confidence in the KOI disposition. For CANDIDATEs, a higher value indicates more confidence in its disposition, while for FALSE POSITIVEs, a higher value indicates less confidence in that disposition.
koi_disposition	Char	The category of this KOI from the Exoplanet Archive. Current values are CANDIDATE, FALSE POSITIVE, NOT DISPOSITIONED or CONFIRMED. All KOIs marked as CONFIRMED are also listed in the Exoplanet Archive Confirmed Planet table
koi_period	double	The interval between consecutive planetary transits (days)

An initial exploration was then carried out on the two data sets to both gain domain knowledge and determine if any additional features could be used as part of the classification implementation.

2.1.3 Initial Exploratory Analysis

Kepler Confirmed Exoplanets

The Kepler confirmed exoplanets file was downloaded through the MAST API, it provides details on the exoplanet such as the star name it orbits, the star mass, brightness, and the orbital period of the confirmed exoplanet.

The analysis looks briefly at:

- The Confirmed Planets dataset being used.
- The main features of the dataset
- If any of the features could be included in the data for classification.

From descriptions in the data documentation 22 columns were chosen from the 373 available columns to determine if they may be useful in the classification models.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4884 entries, 0 to 4883
Data columns (total 22 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   pl_name          4884 non-null   object
 1   pl_letter        4884 non-null   object
 2   hostname         4884 non-null   object
 3   tic_id           4751 non-null   object
 4   disc_pubdate     4884 non-null   object
 5   disc_year        4884 non-null   int64
 6   discoverymethod  4884 non-null   object
 7   disc_locale      4884 non-null   object
 8   disc_facility    4884 non-null   object
 9   disc_instrument  4884 non-null   object
 10  disc_telescope   4884 non-null   object
 11  pl_orbper        4726 non-null   float64
 12  pl_eqt           3710 non-null   float64
 13  pl_dens          4779 non-null   float64
 14  pl_trandur       3638 non-null   float64
 15  pl_radj          4868 non-null   float64
 16  pl_rade          4870 non-null   float64
 17  pl_bmasse        4862 non-null   float64
 18  st_age           3971 non-null   float64
 19  st_mass          4880 non-null   float64
 20  tran_flag        4884 non-null   int64
 21  sy_dist          4869 non-null   float64
dtypes: float64(10), int64(2), object(10)
memory usage: 839.6+ KB
```

Figure 4: datatypes for the 22 chosen columns

Discovery Methods & Discovery Telescopes

These two attributes are checked to see the percentages of each discovery type and what type of telescope (instrument) was used to make the detection.

telescope & method %

disc_telescope	discoverymethod	count	%
0.95 m Kepler Telescope	Transit	3150	64.4963144963145
Canon 200mm f/1.8L	Transit	220	4.504504504504505
3.6 m ESO Telescope	Radial Velocity	200	4.095004095004095
Multiple Telescopes	Radial Velocity	174	3.562653562653563
0.1 m TESS Telescope	Transit	172	3.5217035217035217

Figure 5: Telescope and detection method %

The Kepler telescope discovered 64.5% of the confirmed planets in the dataset with a discovery method of Transit. Missions involving this telescope will be the focus of the machine learning aspect of the project.

Transit discovery method itself makes up 76.82% of all discovered exoplanets and is shown to be the most successful method of detection so far. The transit method paired with the Kepler telescope is the main data source for this projects machine learning implementation.

	discoverymethod	count	%
0	Astrometry	1	0.020475
1	Disk Kinematics	1	0.020475
2	Eclipse Timing Variations	16	0.327600
3	Imaging	55	1.126126
4	Microlensing	120	2.457002
5	Orbital Brightness Modulation	9	0.184275
6	Pulsar Timing	7	0.143325
7	Pulsation Timing Variations	2	0.040950
8	Radial Velocity	899	18.407043
9	Transit	3752	76.822277
10	Transit Timing Variations	22	0.450450

Figure 6 Detection Method %

When looking for just planets that were detected as part of the Kepler mission (not K2, the Kepler Mission and Telescope combination has discovered 54.73% of all exoplanets in the MAST archives and will be the focus of the rest of the exploratory analysis.

	disc_facility	disc_telescope	discoverymethod	count	%
0	Kepler	0.95 m Kepler Telescope	Transit	2673	54.729730
1	K2	0.95 m Kepler Telescope	Transit	477	9.766585
2	La Silla Observatory	3.6 m ESO Telescope	Radial Velocity	200	4.095004
3	Transiting Exoplanet Survey Satellite (TESS)	0.1 m TESS Telescope	Transit	172	3.521704
4	W. M. Keck Observatory	10 m Keck I Telescope	Radial Velocity	171	3.501229
...	...	...	...	...	...
116	Infrared Survey Facility	1.4 m IRSF Telescope	Imaging	1	0.020475
117	Las Campanas Observatory	6.5 m Magellan I Baade Telescope	Imaging	1	0.020475
118	Paranal Observatory	8.2 m ESO VLT UT1 Antu Telescope	Astrometry	1	0.020475
119	Paranal Observatory	8.2 m ESO VLT UT1 Antu Telescope	Transit	1	0.020475
120	Acton Sky Portal Observatory	279mm RASA-11 wide-field telescope	Transit	1	0.020475

Figure 7: Mission detection %

Discovery Years

Kepler launched on March 7, 2009, and ran until October 30, 2018, the years planets where detected are plotted below. With 2014 to 2016 containing the highest number of detections with 2020 and 2021 next. Discoveries were made after the mission finished through exploring the data that had previously been collected.

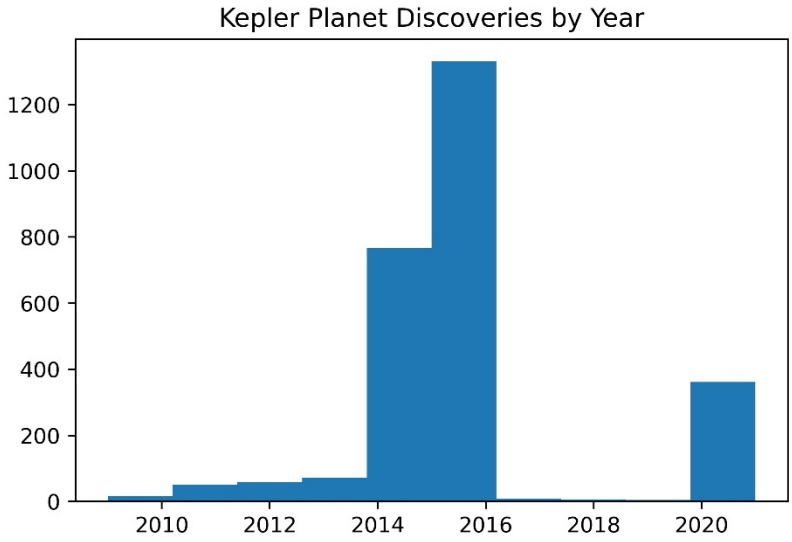


Figure 8: Discoveries by year

Exoplanet Orbital Period

The orbital periods of confirmed exoplanets were examined to see if the distribution might be useful for further analysis. The orbital period can give insight into the distance from a host star a planet is based on their mass, as well as give a possible insight into what generalised orbital period could be used when folding light curves to detect an exoplanet.

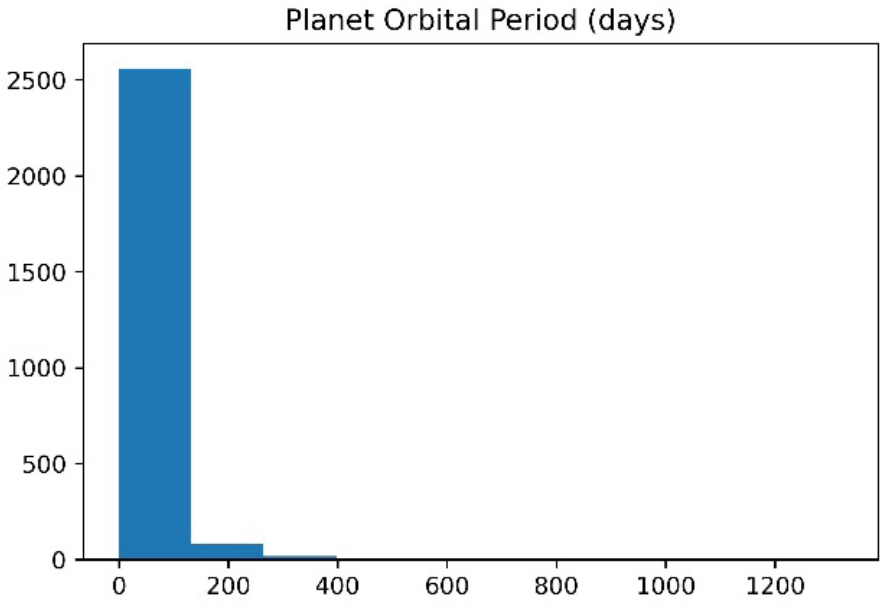


Figure 9: Orbital Period distribution

Orbital Period Stats

Stat	Value
count	2673.0
mean	30.753326332424272
std	75.88205397827873
min	0.35500744
25%	5.19492202
50%	11.55530021
75%	27.082511
max	1322.3

Figure 10: Orbital Period descriptive statistics

The distribution visualisation didn't show any useful information in classification or a generalised orbital period, with a large cluster in the 0 to 100 day period, the mean orbital period was 30.75 days with a standard deviation of 75 days. A minimum of 0.35 days and a maximum of 1322.3 days. With the 75% close to the mean at 27.08 days.

Exoplanet Equilibrium temperature

Measured in Kelvin degrees, the equilibrium temperature was examined to see if it could be paired with other data that could compliment the flux readings later in the project and possibly create a dependant variable. The distribution is centred between the 500 and 1000 degrees mark. Plotted against the orbital period gave no discernible pattern.

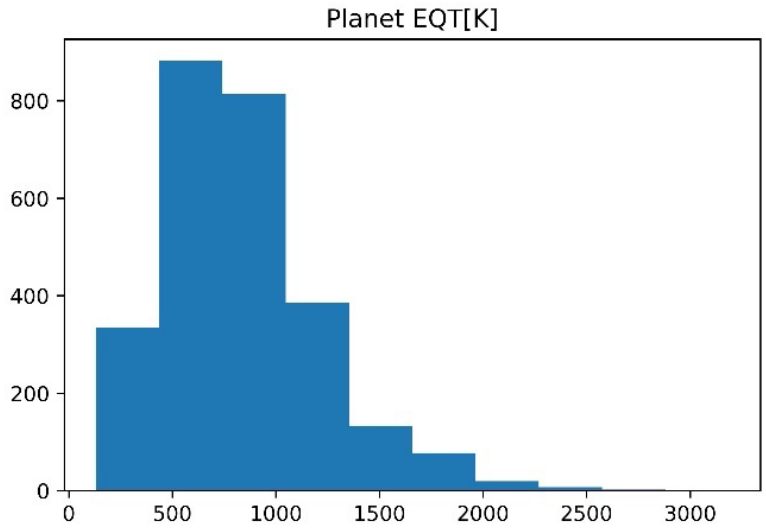


Figure 11: Planet EQ Temp in Kelvin

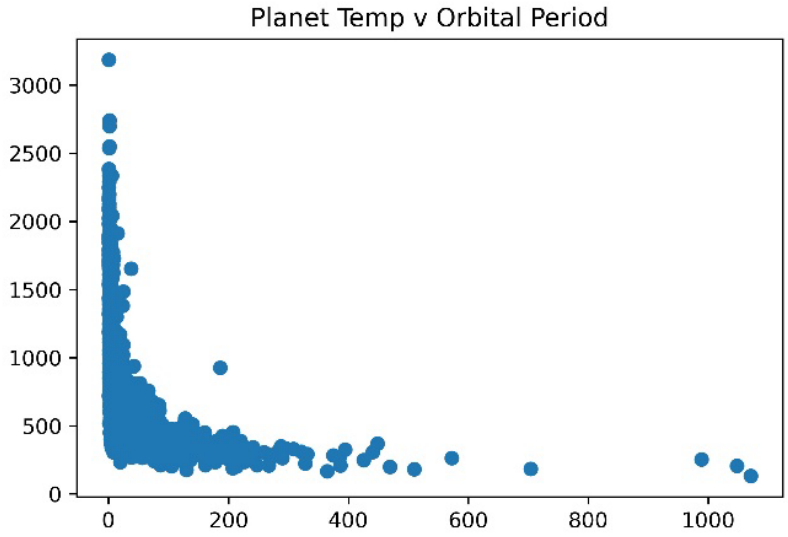


Figure 12: Temp [K] and Orbital Period

Exoplanet Density

The density of exoplanets is measured in g/cm**3 and again when plotting the distribution of the densities, all the readings were stacked around one central point. In the dataset the max density is 1290.0 the min is 0.03, the mode is 5.71 and the median is 3.065. No relationship appears between orbital period and density or equilibrium temperature.

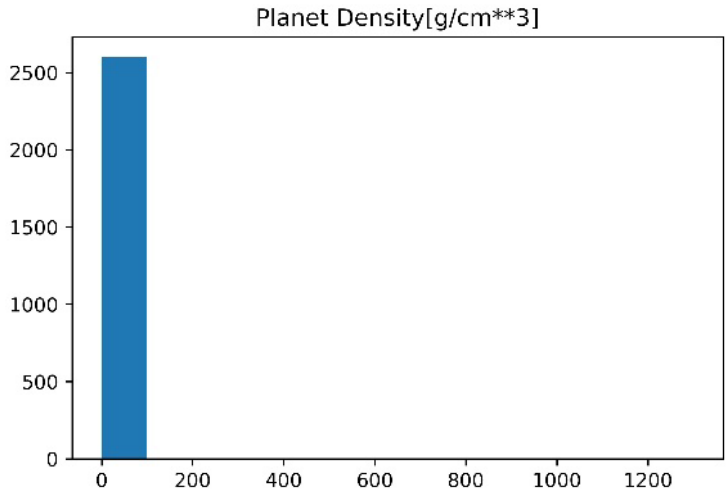


Figure 13: Density distribution

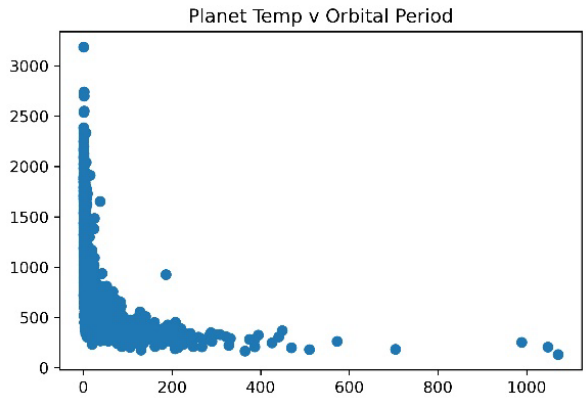


Figure 14: Density and Orbital Period

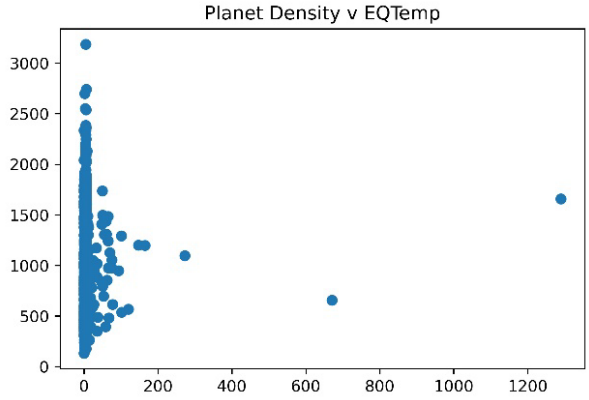


Figure 15 Density and Temperature

Planet Density Stats

Stat	Value
count	2606.0
mean	5.0926749808134995
std	29.954707152509364
min	0.03
25%	2.05
50%	3.065
75%	4.77
max	1290.0

Figure 16: Density descriptive statistics

Exoplanet Density

Transit durations are timed in days and are the duration for the planet to transit across the orbital plane of the star facing towards earth. The hope in examining these measurements would be to find a pattern helpful in applying classification later whereby a generalised dip in flux could be noted as a feature. Although the distribution is centred around the 2 to 3 day period the variance is too great to generalise a folding period for light curve analysis later in the analysis.

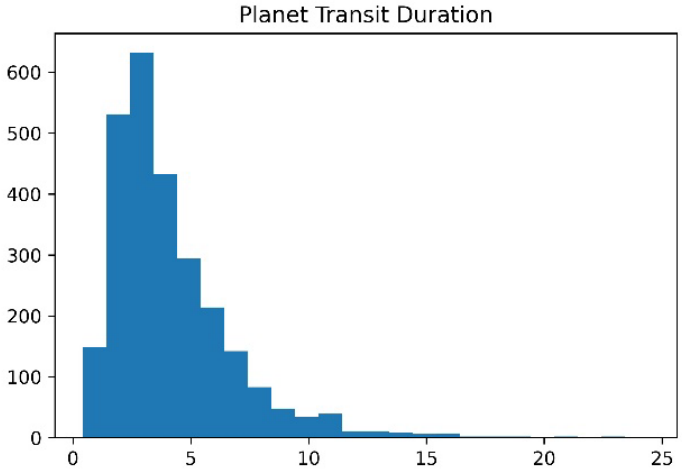


Figure 17: Transit durations [days]

Stellar Mass and System Distance

These attributes were explored next to determine if a possible pattern emerged in distances or stellar masses to concentrate on if a candidate's likely hood could be affected by either of these. Interestingly the furthest planet detected by the Kepler mission is 3460.51 Parsecs or roughly 11,281.2626 light years away. But a useful feature for filtering possible stars when applying to later models was not found.

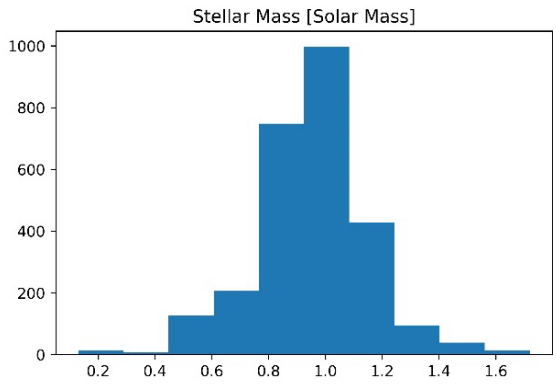


Figure 18: Solar Mass Distribution

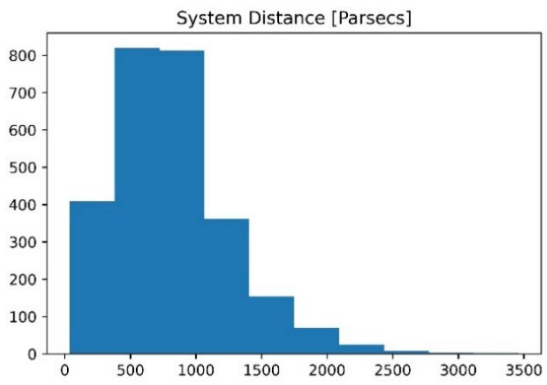


Figure 19: Distance in Parsecs

Stellar Mass [Solar Mass] Stats

Stat	Value
count	2673.0
mean	0.9511036288814071
std	0.19697358813228316
min	0.13
25%	0.84
50%	0.96
75%	1.07
max	1.72

Figure 20: Solar Mass Statistics

System Distance [Parsecs] Stats

Stat	Value
count	2663.0
mean	818.4248652612065
std	449.332458762824
min	36.4396
25%	486.36
50%	769.096
75%	1047.03
max	3460.51

Figure 21: Distance Statistics

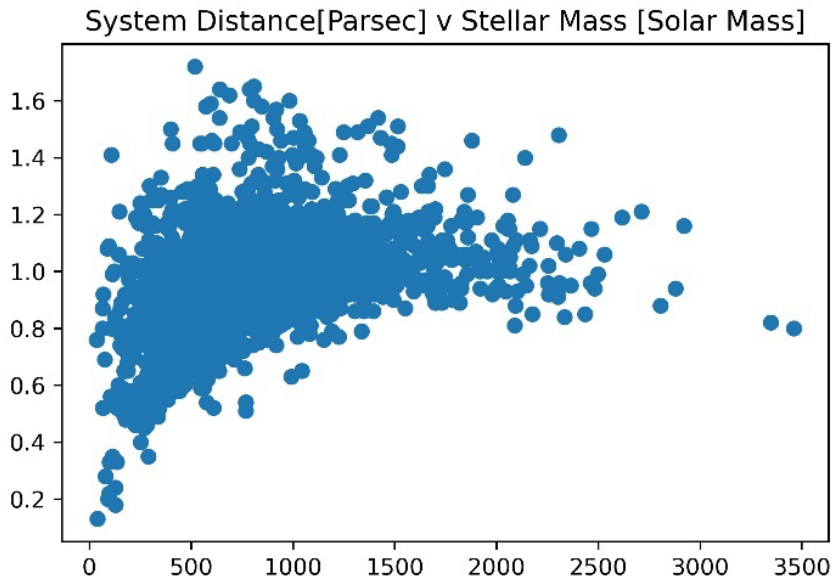


Figure 22: Distance and Solar Mass

Candidate Dataset

Features from the candidate dataset are used to retrieve the light curves from the mission data API. This information is used for pairing the light curve information with the correct Kepler object and labelling the light curve data with its official designation to be used in the machine learning models.

The analysis will look briefly at:

- The Candidates dataset being used.
- The main features of the dataset
- If any of the features will be needed for the Classification Machine Learning

From the 50 columns present in the dataset, there are only 5 columns of interest, these columns will help retrieve the light curves and label each light curve as confirmed exoplanet or a false positive. The columns to be used are the ***kepid***, the ***kepler_name***, ***koi_score***, ***koi_disposition*** and the ***koi_period***. The Kepler name will be used as part of the exploratory analysis on the dataset when counting entries.

First the count and percentage of candidate dispositions are calculated.

Candidate Disposition %

koi_disposition	count	%
FALSE POSITIVE	4840	64.49893390191897
CONFIRMED	2664	35.501066098081026

Figure 23: % of Confirmed versus Negative

The calculation shows that 64.49% of the dataset are False positives and 35.50% of the dataset are confirmed exoplanets, with an imbalance towards the false positive this will need to be addressed and tested when applying the models. Random sampling as well as data generation are available approaches.

Next the koi score, or probability of a correct classification is explored. Where a candidate is a confirmed planet a score closer to one is the preferred value and if a candidate is a false positive, a koi score closer to zero is the preferred value. The mean from the confirmed is 0.96 where the mean from the false positives is 0.038.

Koi Score Stats (Confirmed)

Stat	Value
count	2650.0
mean	0.9641192452830191
std	0.13734826532066208
min	0.0
25%	0.992
50%	1.0
75%	1.0
max	1.0

Figure 24: Koi score stats - confirmed

Koi Score Stats (False Positive)

Stat	Value
count	3946.0
mean	0.03810466294982267
std	0.15879922180278527
min	0.0
25%	0.0
50%	0.0
75%	0.0
max	1.0

Figure 25: Koi score stats - false positive

2.2 Investigating Sample Light Curve Data

This part of the exploratory analysis looks at a sample of a known exoplanet, how the data is retrieved through the LightKurve API, and the generalised steps needed to take place during the retrieval process to transform the data that will make up the dataset. It also displays the characteristics of what the output of a known exoplanet looks like when graphed.

The main steps involved are:

- Downloading Target Pixel Images
- Apply Aperture Masks
- Converting to Light curves
- Removing instrument noise
- Folding Light curves
- Visualising Exoplanet Light curves

Using a combination of the ***kepid*** and the ***koi_disposition*** from the candidates the API is used to search and select the target pixel files of a confirmed exoplanet. Visualizations of the known exoplanet are produced along with the process of folding (using the ***koi_period***) and cleaning the light curve to display how the data will be represented in the final labelled dataset. *kepid_6922244*, Kepler name *Kepler-8b* is a known exoplanet in the candidate dataset, it has a KOI score of 0.998 and a transit period of 3.522498 days. Search results through the API return 49 results.

```
search_result = lk.search_targetpixelfile('KIC 6922244', au
search_result
```

SearchResult containing 49 data products.

#	mission	year	author	exptime	target_name	distance
				s		arcsec
0	Kepler Quarter 00	2009	Kepler	1800	kplr006922244	0.0
1	Kepler Quarter 01	2009	Kepler	1800	kplr006922244	0.0
2	Kepler Quarter 02	2009	Kepler	60	kplr006922244	0.0
3	Kepler Quarter 02	2009	Kepler	60	kplr006922244	0.0
4	Kepler Quarter 02	2009	Kepler	60	kplr006922244	0.0
5	Kepler Quarter 02	2009	Kepler	1800	kplr006922244	0.0
6	Kepler Quarter 03	2009	Kepler	60	kplr006922244	0.0
7	Kepler Quarter 03	2009	Kepler	60	kplr006922244	0.0
8	Kepler Quarter 03	2009	Kepler	60	kplr006922244	0.0
9	Kepler Quarter 03	2009	Kepler	1800	kplr006922244	0.0
...	...	...	...	...	...	...
39	Kepler Quarter 12	2012	Kepler	60	kplr006922244	0.0
40	Kepler Quarter 12	2012	Kepler	1800	kplr006922244	0.0
41	Kepler Quarter 13	2012	Kepler	60	kplr006922244	0.0
42	Kepler Quarter 13	2012	Kepler	60	kplr006922244	0.0
43	Kepler Quarter 13	2012	Kepler	60	kplr006922244	0.0
44	Kepler Quarter 13	2012	Kepler	1800	kplr006922244	0.0
45	Kepler Quarter 14	2012	Kepler	1800	kplr006922244	0.0
46	Kepler Quarter 15	2013	Kepler	1800	kplr006922244	0.0
47	Kepler Quarter 16	2013	Kepler	1800	kplr006922244	0.0
48	Kepler Quarter 17	2013	Kepler	1800	kplr006922244	0.0

Length = 49 rows

Figure 26: Example of search results for Target Pixel Files

Using the last result which is quarter 17 the target pixel files collection for that mission are downloaded and a 6x6 pixels image of one of the files contained can be constructed to visualise the star.

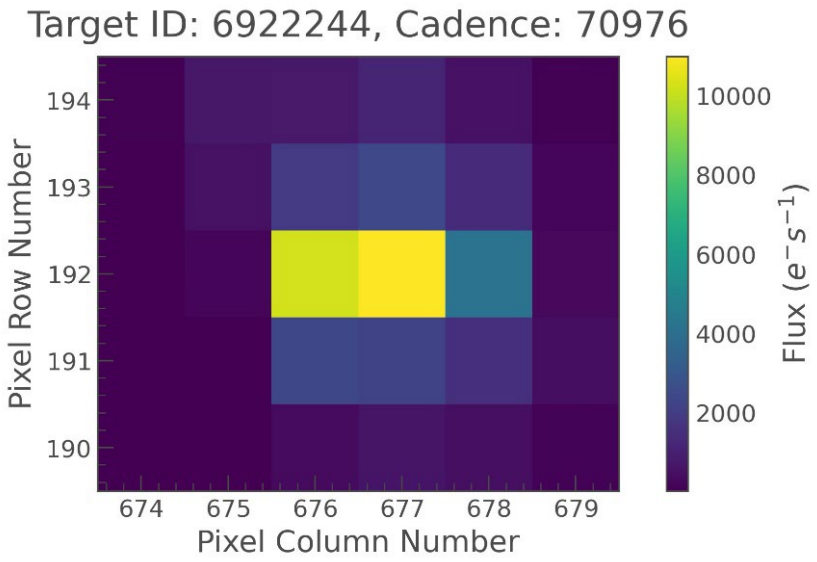


Figure 27: Target Pixel File

Pixel images in each collection contain the flux data (colour scale of the pixel) needed to create the flux readings used to build the dataset for the machine learning models.

An aperture mask is applied to the image to visualise the Flux sources of the image.

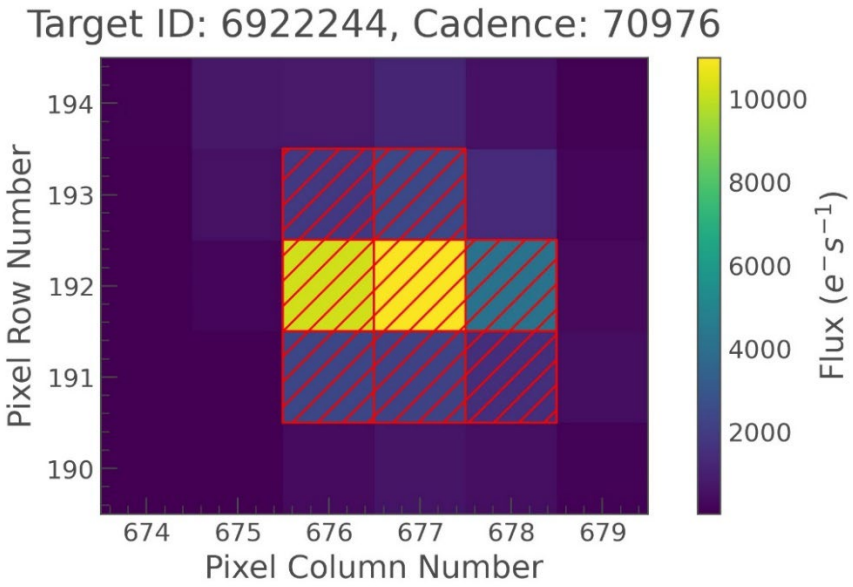


Figure 28: Aperture Mask Application

Each pixel file in the collection can be examined using the Lightkurve python library and used to create a light curve data frame for analysis, where each row contains one observation of the star. The column of concern in the data is the Flux.

```
# convert the collection of targte pixel files to a lightcurve
lc =  tpf.to_lightcurve(aperture_mask='all')
lc
```

KeplerLightCurve length=1286 LABEL="KIC 6922244" QUARTER=17 CAMPAIGN=None

time	flux	flux_err	centroid_col	centroid_row	cadenceno	quality
	electron / s	electron / s	pix	pix		
object	float32	float32	float64	float64	int32	int32
1559.2266409377262	45592.14453125	10.228622436523438	676.8112972450118	192.07152290905165	70976	0
1559.2470752079025	45620.26171875	10.229371070861816	676.8119858259928	192.07197517008095	70977	0
1559.267509578196	45591.25390625	10.22878646850586	676.810995831429	192.07181150973543	70978	0
1559.2879440486286	45591.44921875	10.22887897491455	676.8105453842174	192.07154026954547	70979	0
1559.3083783191832	45591.8125	10.22874927520752	676.8112823546404	192.0720303438852	70980	0
1559.328812689855	45609.59765625	10.229267120361328	676.8105852046192	192.07231446325775	70981	0
1559.3492471606369	45596.43359375	10.228636741638184	676.8114198720602	192.07225536266174	70982	0
1559.3696814315408	45591.1796875	10.22861385345459	676.8110931580934	192.07222875247868	70983	0
1559.390115902621	45598.77734375	10.22860050201416	676.8109345072082	192.0730532169704	70984	0

Figure 29: converted light curve file

Flux (***flux***) values from the data frame above are then be used to produce a graph over time of the flux readings.

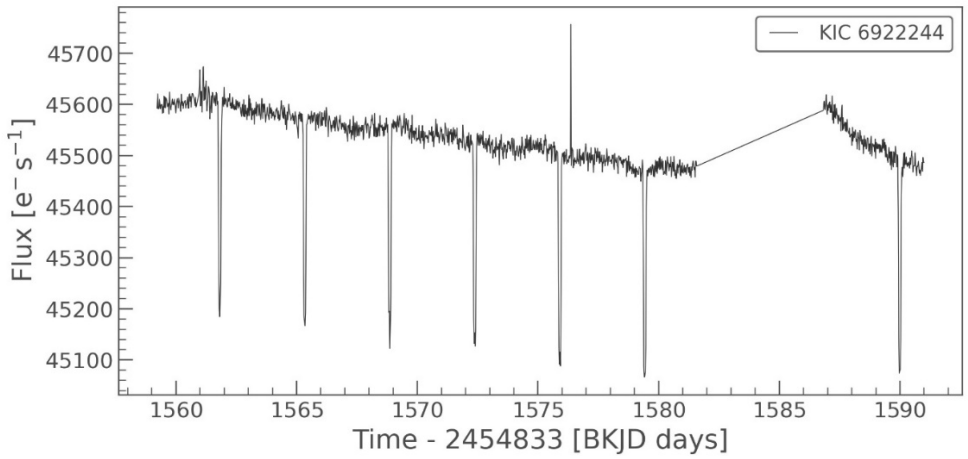


Figure 30: Light curve plotted overtime

A noticeable dip occurs in the graph approx. every 3 days. Checking the ***koi_period*** information from the candidates file confirms an orbital period of every 3.522498 days.

Before folding the graph on this section, it is necessary to flatten the graph to make sure when folding occurs it is done on the same plane.

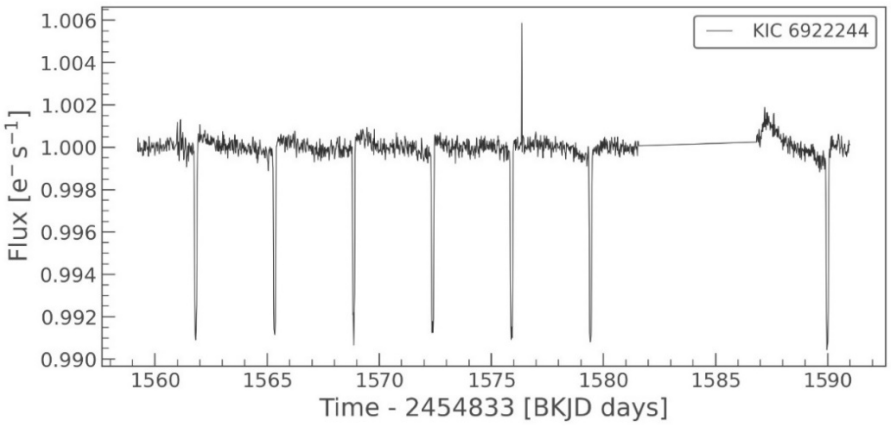


Figure 31: Flattened light curve

Once the graph is flattened it can be folded on itself using the ***koi_period*** measurement to produce a cleaner reading of a single transit event.

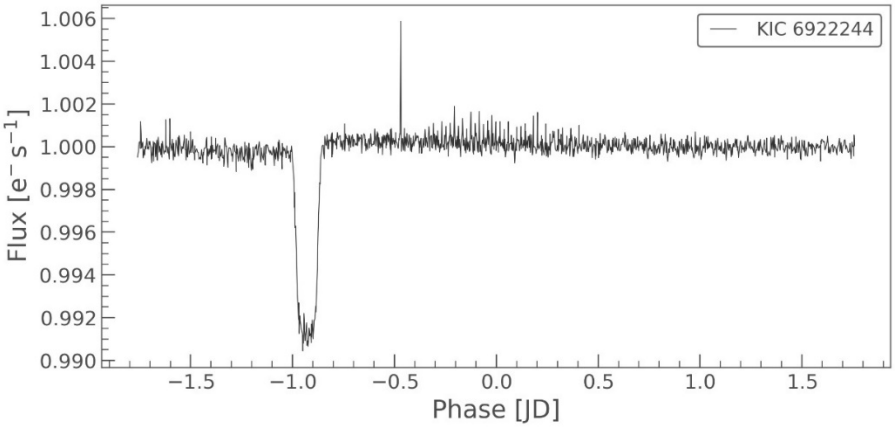


Figure 32: Folded light curve

There is still a lot of noise/distortion in the light curve graph, a gaussian filter is applied to produce cleaner results.

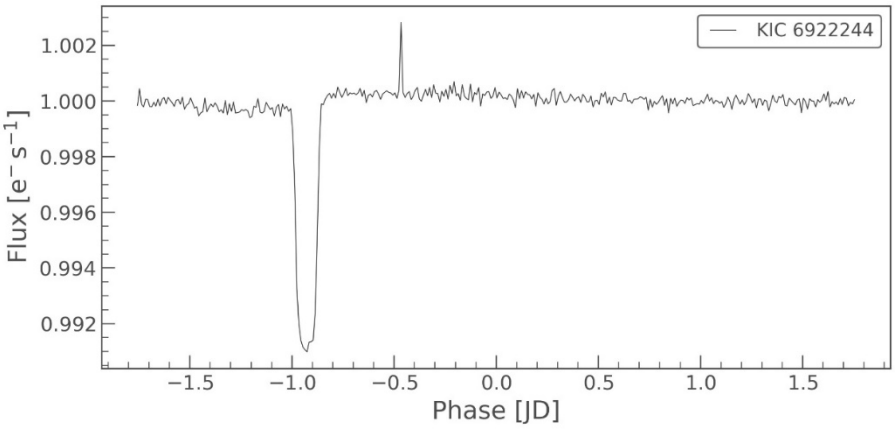


Figure 33: normalised light curve

The exoplanets presence can be seen clearly now as the curved dip in the graph. This process is scripted to automate the retrieval and initial processing of all candidates.

2.3 Initial Flux Data Retrieval

Two methods of creating the flux dataset were created for this project, the first method was to download the raw target pixel files (TPF) programmatically using the Kepler IDs (***kepid***) in the objects of interest file and then using the orbital period ***(koi_period***) from the candidates file to fold the flux reading. Accessing the target pixel file is done with the Lightkurve python library which uses the Astroquery API, also connecting to MAST.

The Kepler data is released in quarterly batches and there are 17 quarter releases. The target pixel files where searched and using the most recent quarter for any particular star the pixel files where downloaded.

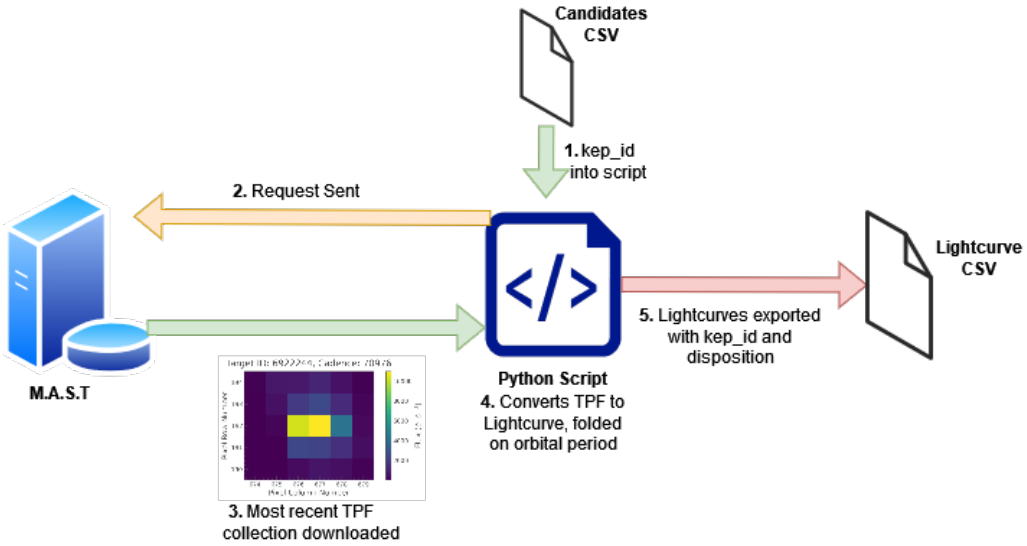


Figure 34: Basic TPF conversion architecture

Files and data generated from the TPF download:

Dataset	Source	Format	Details	Information
Target Pixel Data	Astroquery API through Lightkurve interface	TPF/FITS	9564 Collections, each collection has approx. 1286 pixel files. 12,299,304 pixel files in total	Pixel files converted to light curve objects and deconstructed to csv format
Light curves	Extracted from Target Pixel Data	CSV	9564 Rows, 1294 Columns	1286 flux measurements per row, Kepler ID, name, disposition, P disposition, transit period, period err1, period err2 and Kepler name
Labelled Dataset	Compiled	CSV	7504 Rows 1287 Columns	1286 flux measurements per row including its classification label, 1 = CONFIRMED, 0 = FALSE POSITIVE

2.4 Final Flux Data Retrieval

The issue arose that for each star listed in the objects of interest file, it may have been imaged during different quarters of the Kepler mission and a planet may have been detected in any one of them. The initial assumption when beginning the TPF retrieval was these detections would be present at the correct orbital period listed in the candidates file.

The TPF method only allowed the downloading of one collection of pixel files relating to a specific quarter of release. To run this method over all 17 quarters would have taken too long and the memory capacity needed to large, to correct this error a second method for retrieving the flux readings was created. This method involved directly downloading the complete collection of flux readings instead of the TPF files from every quarter for each star. 7504 stars where present in the candidates file, removing duplicate stars resulted in 6610 stars to have flux readings, which were directly downloaded using a different query in the LightKurve API to the TPF download. In this case, every instance can be downloaded in the form of its flux readings in FITS file format. Each instance had an average collection size of 50,000 over the mission period resulting in approximately 300 million collections of flux readings downloaded locally to the FITS folder created in the project directory (31.3GB).

Each instance was then stitched together using a second script to create one continuous flux file for each star. The results were centred and binned into 2000 flux readings for each star. Where data was missing due to the binning process a linear interpolation was applied (Vanderburg & Shallue, 2018). The results where then transposed to row instead of column where each row contained one observation of a star, its ID, name, disposition, p disposition and 2000 flux readings. These rows where then concatenated together, normalised, and exported to a single csv file (***candidates_with_flux.csv***). The disposition and p disposition where then used to create the labelled dataset containing either 1 for confirmed planet or 0 for false positive and 2000 flux readings (***labeled_data.csv***).

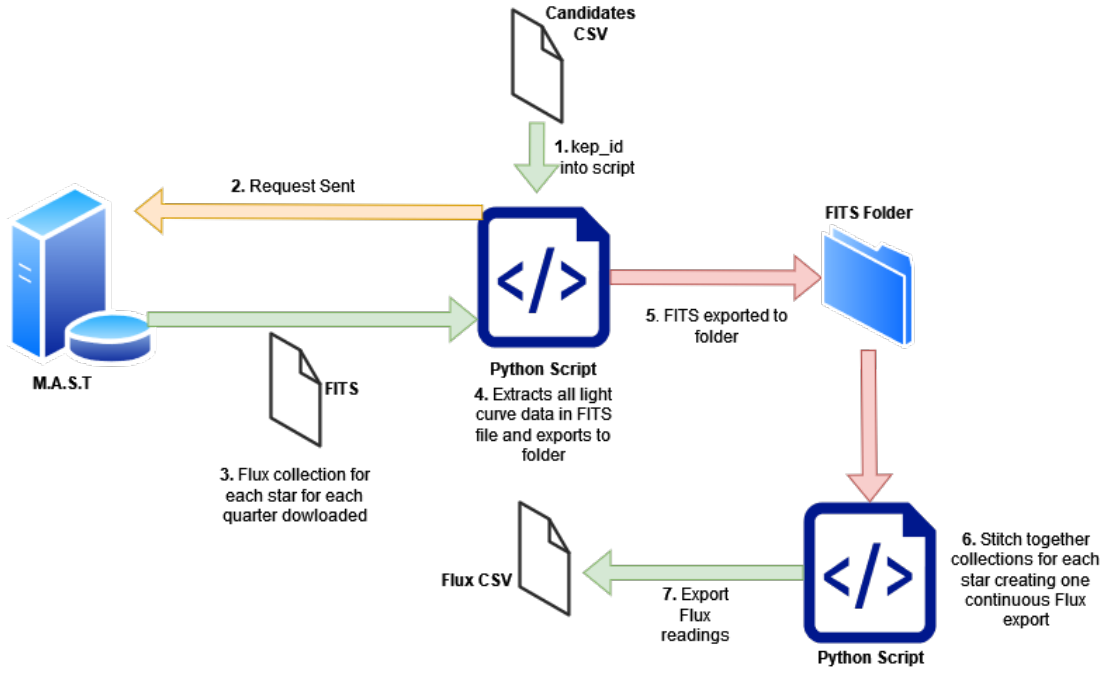


Figure 35: Basic Light curve download (FITS) architecture

Dataset	Source	Format	Details	Information
Flux Collection	Astroquery API through Lightkurve interface	FITS	6,610 Collections, each collection has approx. 50k instances. 300million flux readings in total	Pixel files converted to light curve objects and deconstructed to csv format
Candidates with flux	Extracted from Flux collection	CSV	6167 Rows, 2004 Columns	2000 flux measurements per row, Kepler ID, Kepler name, disposition, P disposition.
Labelled Data	Compiled	CSV	6167 Rows 2001 Columns	2000 flux measurements per row including its classification label, 1 = CONFIRMED, 0 = FALSE POSITIVE

3.0 Methodology

The methodology chosen to be used throughout this project was KDD. KDD (Knowledge, Discovery of Databases) has five distinct steps (Selection, Pre-processing, Transformation, Data Mining, and Interpretation/Evaluation) that are often applied iteratively and is used to extract useful structured patterns from data. It is a core data mining methodology, and it was determined that this project would benefit from the application of this methodology by breaking the process into smaller problem sets and providing structure to each step.

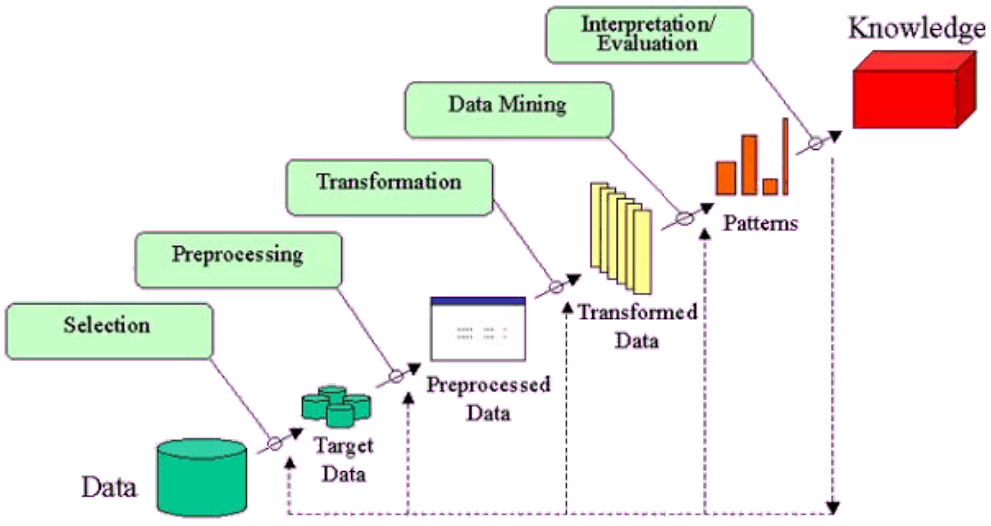


Figure 36: KDD Steps (Fayyad, et al., 1996)

3.1 Selection

The data selection process involved identifying the necessary provider where the flux data could be accessed programmatically in its raw state without any pre-processing haven been applied already. As the data is freely available for public use and is highly publicised in relevant work relating to this project, the data source contributor was easy to identify as the Mikulski Archive for Space Telescopes (M.A.S.T).

The input shape and data format used in the machine learning models was known from previous literary reviews and needed to be numeric with the relevant classification labels applied. The challenge for the initial data selection was how to programmatically collect the data, apply labelling and transform it to the structure needed.

To correctly identify which Kepler IDs to search and download through the LightKurve API other data was needed. Determining which datasets from the provider contained the additional information needed was also a challenge as there are hundreds of different variations of datasets and different access points for downloading them; bulk downloading, API, and direct weblink download.

On reading the documentation available and using the MAST in built viewer to view tables two data sets where identified that provided the correct information to be able to search the LightKurve API and apply labelling, the Kepler Objects of Interest table, and the confirmed exoplanets table. Considering the scale of the project, the API endpoints through python scripts was the most suitable way to download this data and keep in line with the programmability requirements needed for the project.

After downloading and initial exploration of the identified datasets it was determined that from the Kepler Objects of Interest table the data needed was the ***kepid*** for identifying stars, the ***kepler_name*** for identifying the exoplanets if any, the ***koi_score*** for identifying the probability of an exoplanet, the ***koi_period*** for folding purposes and the ***koi_disposition*** for labelling. The Kepler list of confirmed exoplanets contained other data needed, ***disc_instrument***, ***disc_telescope***, ***disc_facility*** and ***discoverymethod*** for identifying Kepler Mission objects, and ***pl_name*** for confirming items in the objects of interest table.

Once downloaded and the correct exoplanet information was selected and organised using the columns mentioned, the new data contained a list of known exoplanets and known false positives. The ***kepid*** from this data was then used to retrieve the relevant target pixel files / flux readings through the LightKurve API for each star.

3.2 Pre-processing

The pre-processing stage of the methodology in relation to this project involves using the data obtained from the Kepler Objects of Interest table paired with the Kepler list of confirmed exoplanets and retrieving the flux readings for each star in raw format, applying pre-processing techniques during the download phase before exporting to a csv file ready for transformation.

During the pre-processing stage it was important to apply the principals of tidy data (Wickham, 2014), making sure the output file to be analysed met the following requirements:

1. "*Each feature measured should be in one column*". In terms of this project meaning each Label and Flux reading
2. "*Each observation should be in a different row*". Meaning each star in this project
3. "*There should be one table for topic of interest*", for this project there would be only one table at the end of pre-processing and transformation.
4. "*If there are multiple table, they should include a column that allows them to be linked*". In the case of this project this only applied in the initial stages of downloading and pre-processing.

As mentioned previously an initial data download using Target pixels was used in conjunction with data pre-processing to create a data set ready for labelling. After initial errors in this data a second download and pre-processing method was implemented, both methods are discussed at this stage.

3.2.1 Method 1: Target Pixel Files

Using the target pixel file method, the Kepler ID was used to search mission data using the Lightkurve API for all the TPF collections and the most recent collection downloaded. Once downloaded the TPF collection was converted to light curve information with aperture mask applied, flattened, and then the orbital period (***koi_period)*** was used to apply an automatic fold to the light curve reducing the data size. A gaussian filter was then applied to the folded light curve which was then converted into a panda's data frame. The data was then transposed from column to row and the information relating to the ID, disposition and transit period where added.

Steps included in the Target Pixel File method:

1. API searched using Kepler ID
2. Mission information retrieved
3. Most recent quarter of imaging identified
4. TPF collection for identified quarter download
5. Aperture mask applied and data flattened
6. Fold applied on given orbital period and centred
7. Gaussian filter applied to reduce noise
8. NANs searched for and removed
9. Data converted to pandas' data frame
10. Column transposed to row
11. Kepler name, ID, and disposition added to data frame
12. Data frame added to list and process repeated for each Kepler ID
13. List of data frames concatenated for export to csv.

This pre-processing part of the project processed over 12 million TPF's and took approx. nine hours of continuous runtime to complete. The data was then exported to a csv file and saved locally. When inspected though not all stars contained the same number of flux readings, ranging from 1,286 to over 3,000.

The process was applied again but a window size of 401 and a bin size of 0.01 were set as parameters while folding the TPF's, this produced a flux reading size of 1,604 for each star. When null values where removed the smallest number of flux readings for a star was again 1,286 with the max number of readings at 1,604. As the centre point of the data was column 643 and was where any exoplanet should be located, the null values at the end of each row were removed for a completed shape of 1,286 Flux readings and 6,167 instances.

Checking the data integrity after this process through manual inspection and graphing flux readings of known exoplanets showed that not all exoplanets where present in the data set due to the nature of their orbital time around their host star and the quarter in which it was imaged, a new solution was needed.

3.2.2 Method 2: FITs collections & stitching

To overcome the data integrity issues in the Target Pixel File method, this second method using the LightKurve API, accessed the complete mission catalogue for each star spanning the entire mission period. Each collection for each star was downloaded, stitched together using a feature available in the LightKurve API to create one single observation and saved into its own folder inside a newly created FITS folder in the data directory of the project.

Memory management was put in place as this method of downloading complete collections created a cached version in the LightKurve system directory. After every download the cached version was deleted leaving just the stitched version.

After duplicate stars were removed from the kepid list (some stars had multiple exoplanets) the original 7504 stars was reduced to 6610. With each star having approx. 50 thousand flux readings registered the total number of readings download was approx. 300 million. To be able to complete this process a function in the Python script was created to allow the process to be pause and then resumed from the previous spot. In total these files took approx. 40 hours to download. Although time consuming, this method of download ensured that each exoplanet observation would be present, and the data integrity maintained.

To reduce the data to 2,000 flux readings per observation and maintain any presence of exoplanets different methods of folding needed to be applied than in the target pixel file method.

First a temporary folding based on the official orbital duration was set using the API fold method. Then a fractional duration calculation was made based on the transit duration and the orbital period.

$$fractional\ duration\ = \frac{(duration\ in\ hours\ /\ 24)}{orbital\ period}$$

Once the fractional duration was calculated and the temporary folding created, the phase period of the temporary folding was used to create a phase mask.

$$phase\ mask\ = |phase < (fractional\ duration\ \times\ 1.5)|$$

The phase mask was then used to create a transit mask which could be applied to the entire flux reading during the flattening process. The transit mask uses the time interval of the original light curve and the phase mask time interval value to return a transit masking.

After the transit mask had been created it was applied during the flattening and folding process where the folding could now take place on the orbital period.

The newly flattened and folded light curve was binned into sizes of 2,000 to fit inline with the planned architecture of the CNN model. (Vanderburg & Shallue, 2018) . the data was then normalized. After normalization nan values where masked and the global light curve was created using the calculation provided through the LightKurve API.

$$global\ light\ curve = \left(\frac{binned\ light\ curve}{|nan\ mask\ flux\ min|}\right) \times 2.0 + 1$$

The global light curve was then converted into a panda's data frame. On examination of the data at different stages of the process it was noticed that due to the action of binning the light curves small portions of missing flux readings were created, to impute these missing values a linear interpolation was applied to the flux reading and did not affect the overall integrity of the data (Vanderburg & Shallue, 2018).

The newly created global light curve was then transposed from one column of flux reading to one row of flux readings. Each row now containing 2000 normalized flux readings. For each star this method was applied to, the ***koi_disposition*** was added based on the ***kepid*** as a labelling column for a confirmed or false positive exoplanet.

The single row data frame was then added to ***candidates_with_flux.csv*** file ready for further transformation and labelling before the data mining process could begin.

3.3 Transformation

Transforming the flux and candidate data as part of the methodology used in the project involved correctly labelling the data in the context of the machine learning models to be applied. The data mining implementation involves binary classification therefore the ***koi_disposition*** column which was used to label the flux readings was converted from "confirmed" to 1 and "false positive" to 0.

For the flux readings themselves there are 2,000 readings per star. Resulting in untidy fluctuations that could impact any models used.

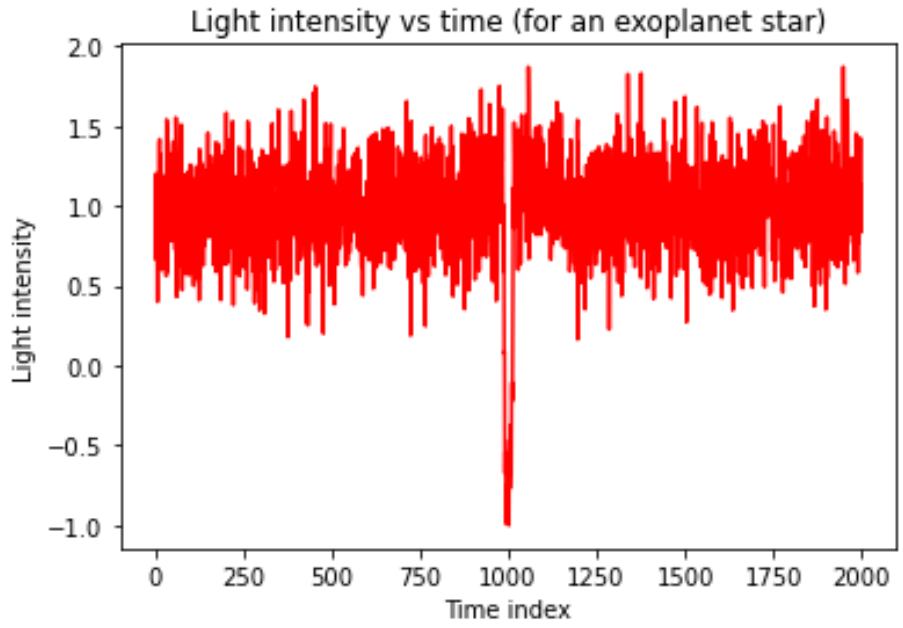


Figure 37: Flux for known exoplanet plotted

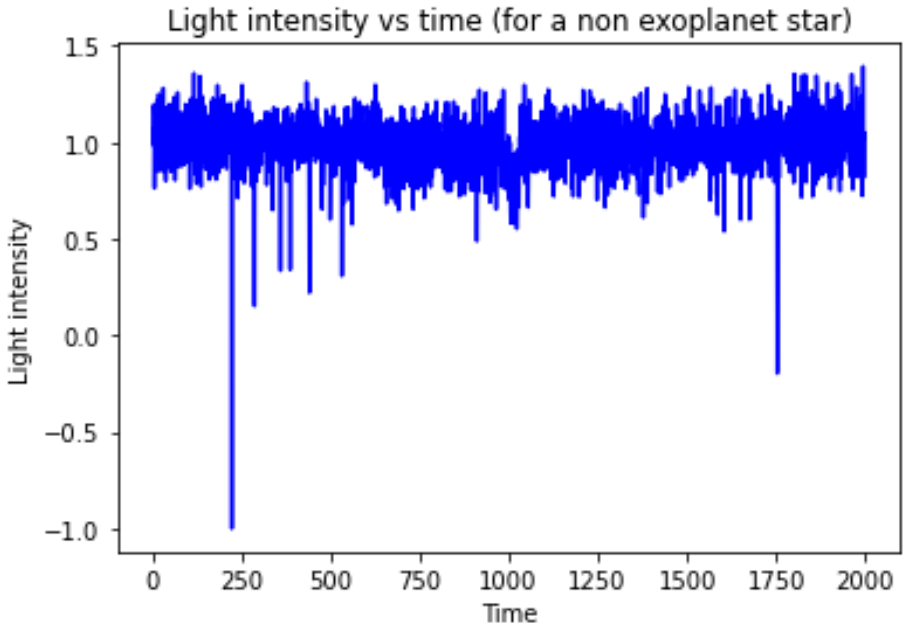


Figure 38: Flux for known false positive plotted

To smooth the data out and remove any noise present in the flux readings a gaussian filter with a sigma of 50 is applied at this stage similar to the initial target pixel method.

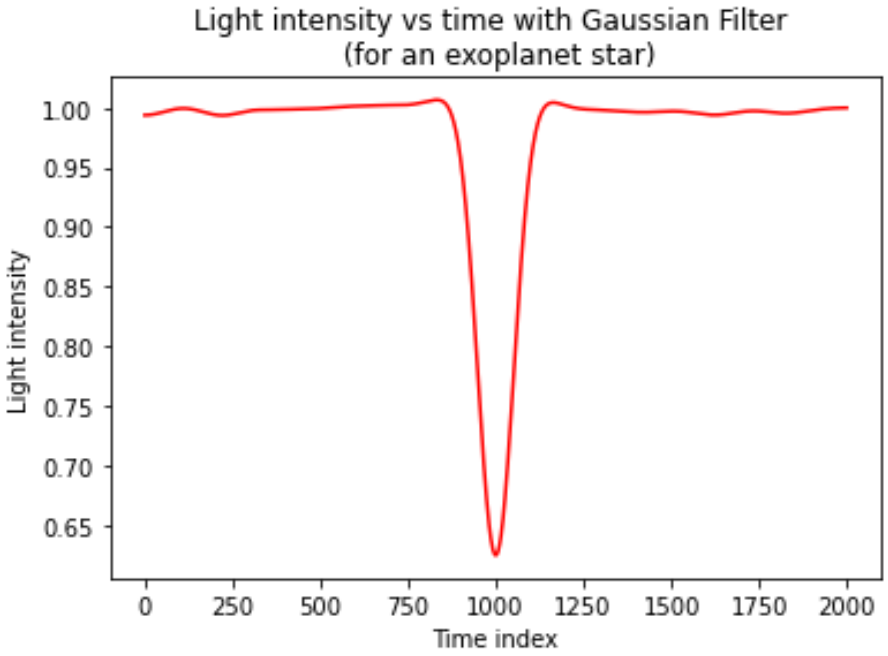


Figure 39: Flux for known exoplanet with gaussian filter

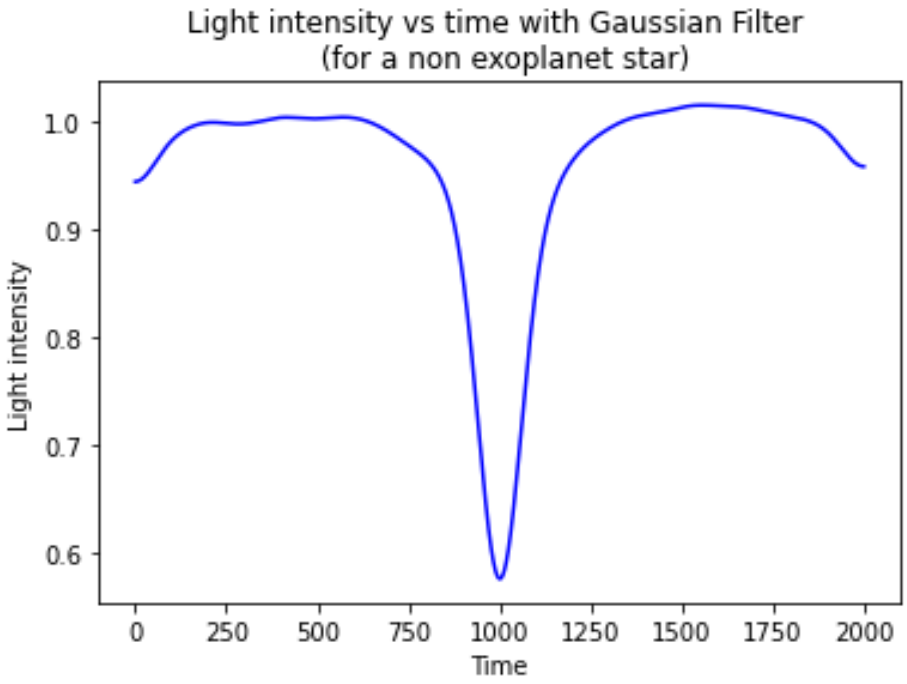


Figure 40: Flux for known false positive with gaussian filter

Further making the data suitable for the machine learning models scaling is applied to the flux readings between 0 and 1.

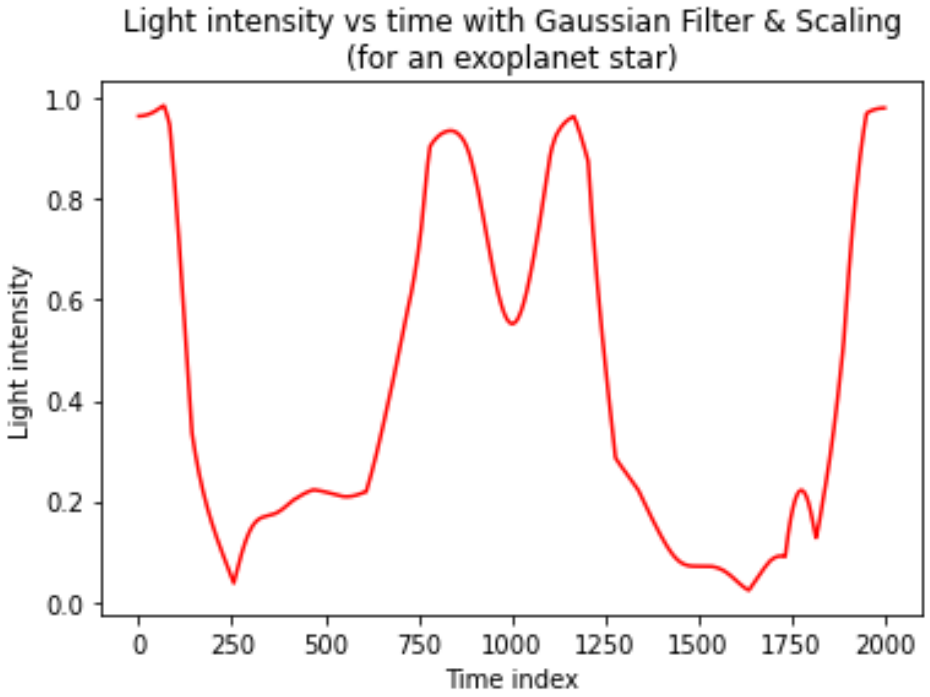


Figure 41: Flux for known exoplanet with gaussian filter and scaling

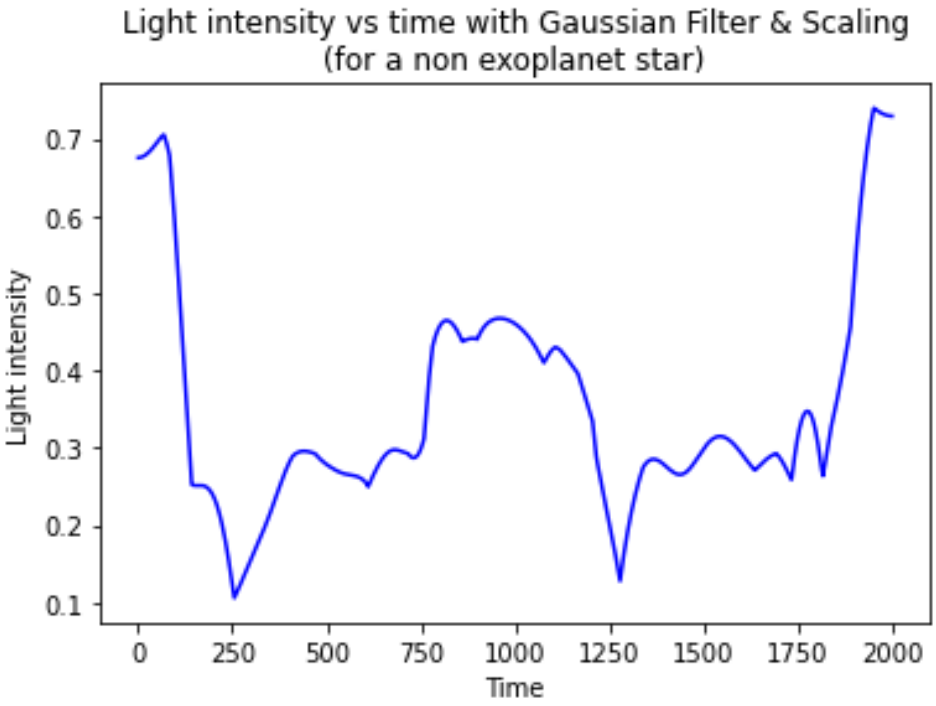


Figure 42: Flux for known false positive with gaussian filter and scaling

To prepare the labelled flux readings for the data mining models, training, test and validation sets are created from the labelled data set. The initial shape of the labelled data is 6167 rows with 2001 columns. Of the 6167 rows 1,654 are confirmed exoplanets leading to a 73% imbalance in favour of false positives. A desired split of the data for this project was 75% training, 12.5% testing and 12.5% validation. To achieve this split while keeping the same proportion of confirmed versus false positives in each batch of data, a stratified k-fold approach was used. The resulting data sets where:

Dataset	Confirmed	False Positive	Total	Percent Split
Test	207	564	771	27% - 73%
Train	1240	3382	4622	27% - 73%
Validation	207	564	771	27% - 73%
Totals	**1654**	**4510**	**6164**	**27% - 73%**

Due to the imbalanced nature of the data sets two additional approaches were considered, data generation and sampling. The LightKurve API provided documentation on generating artificial light curves, but it was decided that at this point in the project it would be more conductive to continue the research with the data as it was and if issues arose from this level of imbalance, to come back to this point iteratively and make any needed adjustments. The same consideration was taken for the sampling method.

To be prepared for testing of the models and knowing that initial weights could be applied to the classes as part of the CNN modelling, weights for each class were calculated. Using the training set the weights for each class were calculated using the formula:

$$class\ weight = \frac{\left(\frac{1}{count\ of\ class}\right) * total\ observations}{2}$$

The results of the class weightings were:

	Confirmed	False Positive
Weighting	1.863	0.683

The next step in the transformation process was the shuffling of each data set. A random seed was set for reproducibility and each data set shuffled. A McNemar's test is applied to the prediction results of each model at a later stage, so the assumption of the test that each model makes predictions on the same data set was to be upheld here with the order of the shuffle upheld through multiple tests. Once the data sets were shuffled the target variables (labels) where separated from the independent variables (Flux readings) creating an X and Y for each data set. The resulting transformation concluded in X-Train (Labels), Y-Train (Flux readings), X-test (Labels), Y-test (flux readings), X-val (Labels) and Y-val (Flux readings) to be used in the data mining methods.

3.4 Data Mining

In the data mining portion of the project the data sets created in the transformation process were used with the main objective of classification, two approaches were used, Convolutional Neural Networks (CNN) and Capsule Networks.

3.4.1 CNN

The basic architecture of the CNN was taken from the paper "*Identifying Exoplanets with Deep Learning: A Five-planet Resonant Chain around Kepler-80 and an Eighth Planet around Kepler-90*", from literary reviews of CNN usage this architecture proved to be the state of the art version for comparison with 95.4% Accuracy and a 98.5 AUC on classification using the global light curves (Vanderburg & Shallue, 2018). In the paper a combination of two CNN's were created using the global light curves and a local (cropped version) of the same light curve. When these CNNs are combined the accuracy scores where Global 95.4%, Local 92.4% and combined 96.0%. The AUC results for these were Global 98.5, Local 97.3 and combined 98.8.

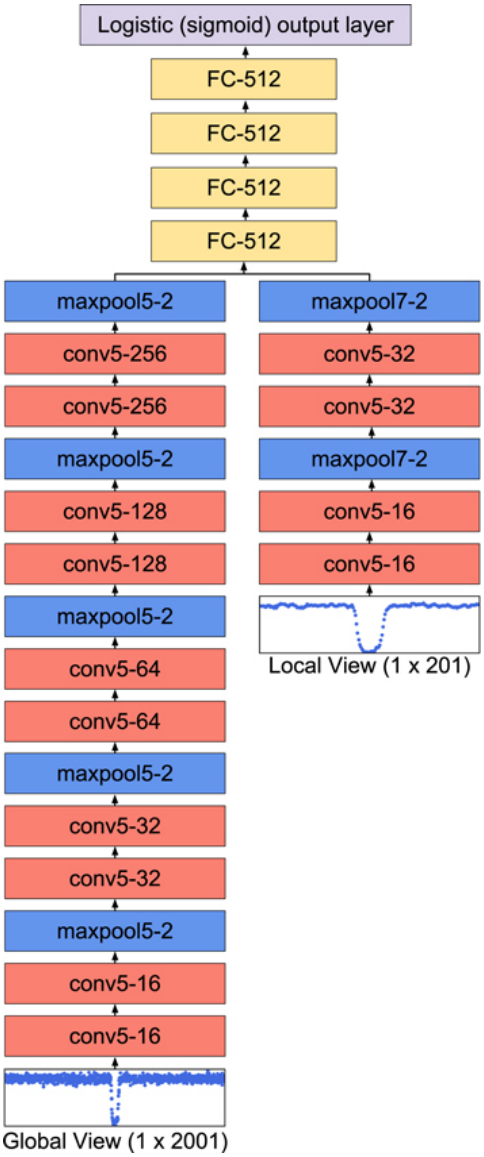


Figure 43: CNN Architecture (Vanderburg & Shallue, 2018)

For this project the global view architecture was followed in attempt to reproduce the results of the paper, the global view architecture was followed as it showed the best single performance out of the joined model.

The architecture of the CNN has 15 layers total, 5 one dimensional convolutional layers, 5 pooling layers, 4 fully connected dense layers and an output layer.

The convolutional layers take certain parameters for number of filters, kernel size, activation function, and padding. A filter number is set for the number of filters per layer, each filter identifies features in the data set at that layer, the filters at that layer are used to give an indication of how strongly a feature appears. The location of where that feature occurs in the data does not matter, therefore with CNNs this method reduces the number of weights the network needs to learn. The weights for these filters are then updated throughout the training process. The kernel size relates to the size of each of the filters in relation to shape of the data.

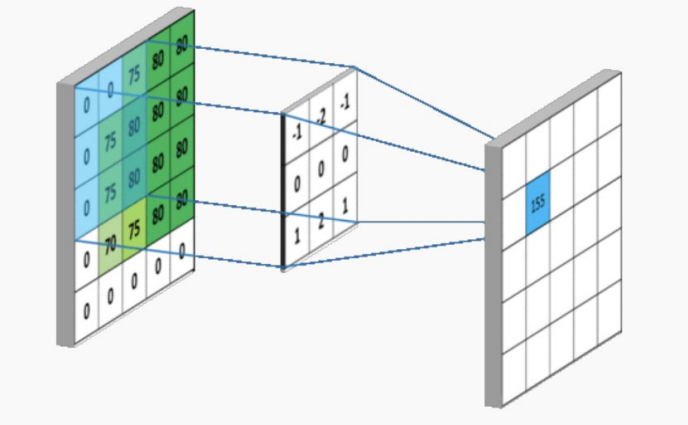


Figure 44: Example of convolutions and kernels in a CNN for image processing (Stuart, 2019)

The padding argument for the convolutional layer is used primarily for the edges of the data structure, using padding can remove the need for any down sampling needed in the data when passing through layers, for example in image classification it would deal with edges of the image. When the kernel specified is applied to the image it makes sure the output from the convolutional layer is the desired shape. (Stuart, 2019)

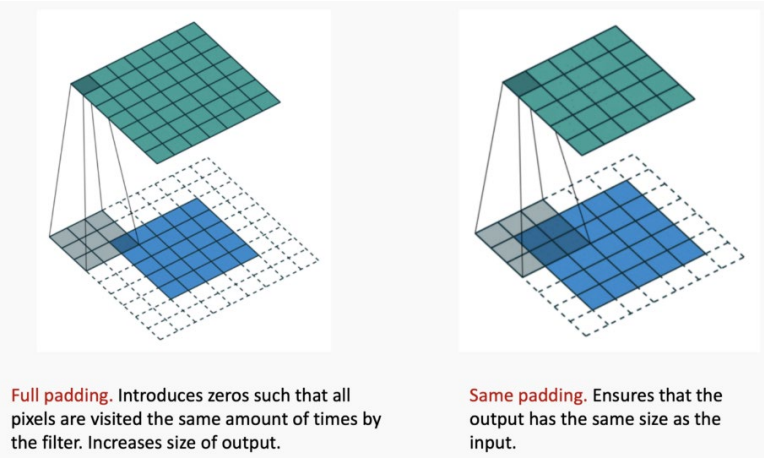


Figure 45: Illustration of padding being applied in a CNN (Stuart, 2019)

The activation function of the convolutional layer is where the non-linearity occurs, as this layer is not the final output layer a rectified non-linear unit (ReLU) function is applied.

The pooling layer is a form of dimension reduction. It uses, for example, the average, or in the case of the architecture presented, the max value of a filter. Then using a sliding window approach the max pooling is calculated across the data set. Parameters set for this layer are the size of the window and the stride (size of the movement steps across the data set).

The dense or fully connected layers in the network are used to flatten the output from the convolutional layers and generally appear in the model architecture just before the final output. These layers aggregate the information from the feature map created in the previous layers and are used to produce the final output. The parameters this layer takes are the number of nodes in the layer and the type of activation function required depending on the task of the network. In the context of this project the fully connected layers use a ReLU function again until the final output layer which is a single sigmoid output.

Summary of the architecture to be used in the CNN:

Layer Type:	Parameters	Parameter Values
First Convolutional Layer	Filters, Kernel Size, Activation	16, 5, ReLU
Max Pooling Layer	Window size, Stride	5, 2
Second Convolutional Layer	Filters, Kernel Size, Activation	32, 5, ReLU
Max Pooling Layer	Window size, Stride	5, 2
Third Convolutional Layer	Filters, Kernel Size, Activation	64, 5, ReLU
Max Pooling	Window size, Stride	5, 2
Fourth Convolutional Layer	Filters, Kernel Size, Activation	128, 5, ReLU
Max Pooling	Window size, Stride	5, 2
Fifth Convolutional Layer	Filters, Kernel Size, Activation	256, 5, ReLU
Max Pooling	Window size, Stride	5, 2
Flatten	None	
Fully Connected Layer 1	Nodes, Activation	512, ReLU
Fully Connected Layer 2	Nodes, Activation	512, ReLU
Fully Connected Layer 3	Nodes, Activation	512, ReLU
Fully Connected Layer 4	Nodes, Activation	512, ReLU
Output Layer	Nodes, Activation	1, Sigmoid

Before compiling the CNN model, an optimizer and a loss function are added. The optimizer in the architecture presented is the Adaptive Moment Estimation (ADAM optimizer). This type of optimizer uses a combination of two gradient descent calculations, the momentum and the root mean square propagation, to efficiently control the rate of gradient decent and reach the global minimum with minimum oscillation in the gradient. The Adam optimizer can be termed as industry standard in classification networks.

For the loss function of the model different functions where investigated. Categorical cross entropy was looked at but as this type of loss function is more useful in multi class classification it was decided that binary cross entropy would best suit the project.

3.4.2 Hypermodel CNN

To further test the hypothesis that the CNN architecture discussed in the previous section was the one most suited for the problem domain, a second version of the CNN was created. This version was built using the Keras Hypermodel library and involved inputting basic architecture details with options to cycle through for building and testing different models. Then using Hypermodel tuning, each combination of options was used, testing each model, and returning the best combination of layers / parameters the Hypermodel could find.

The options structure for the Hypermodel parameters was:

- Up to 5 one dimension convolutional layers with ReLU activation.
- Each convolutional layer cycled through:
    - A range of filters from 16 to 256 in increments of 16.
    - Kernel sizes 3, 5, and 11.
- A max pooling layer added after each convolutional layer with:
    - Window size of 5 or 10
    - Stride of 2 or 4
- A flatten layer
- Up to 4 fully connected layers with ReLU activation
- Each fully connected layer cycled through:
    - Node numbers of min 32 and max 512 in increments of 32
- A single node output layer with sigmoid activation
- Adam optimizer
- Binary cross entropy loss function
- A learning rate options of 0.01, 0.001, 0.0001

As this created a large number of models to be tested, some constraints were set on the tuning. The original model from the previous architecture was set as default. Hyperband tuning from the Keras API was used. Each model was tested at 50 epochs. Tuning factor was set to 3, which is the reduction factor for the number of epochs and number of models for each bracket (Li & Jamieson, 2018), and the hyperband iterations was set to 2 which is the number of iterations over the full algorithm. It was kept at 2 due to time constraints but a higher value is usually recommended depending on resources and time available.

3.4.3 Capsule Network

Capsule Networks were introduced in 2017 by Geoffrey E. Hinton in the paper "*Dynamic Routing Between Capsules*" (Hinton, et al., 2017), in attempt to overcome some of the problems associated with CNNs. The problems being CNNs don't store the relative spatial relationship between features in a data set, they require relatively large amounts of training data to be able to achieve high accuracy, and the use of pooling layers for reduction increases the possibility of losing some useful features through the network.

Capsule Networks as described in the paper were designed to deal with computer vision problems and as such take 3 dimensional input, the height, width, and colour channels of the image being classified. The basic architecture behind a capsule network is similar to a parse tree, where each layer is divided into small groups of neurons called capsules and each node corresponds to an active capsule. The routing process between layers involves each capsule iteratively choosing a capsule from the previous layer as the parent. This iterative process addresses one of the CNNs issues by storing the spatial relationship of features and assigning these features as part of the whole.

A unique property of the capsule network is a separate SoftMax activation function that gives a probabilistic output on whether a particular entity exists. The output of each capsule is a vector and using dynamic routing the output is sent to the appropriate parent node. This type of routing, known as "*routing-by-agreement*" proves to more effective than the pooling strategy applied in CNNs. The agreement part of the routing is the scalar product of the agreement between the output (Vj) from each capsule (j) and the prediction from the current capsule ($\widehat{U}_{j|i}$). The agreement formula ($a_{ij} = V_j \,.\, \widehat{U}_{j|i}$) is then treated as log likelihood (Hinton, et al., 2017).

Pseudo code for the routing algorithm taken from Hinton's 2017 paper (Hinton, et al., 2017):

Define Routing ($\widehat{U}_{j|i}$, r, l)

- **For** all capsules i in layer l and capsule j in layer $(l + 1)$: $b_{ij} \leftarrow 0$
    - **For** r iterations
        - **For** all capsule i in layer l: $c_i \leftarrow softmax(b_i)$
        - **For** all capsule j in layer $(l + 1)$: $s_j \leftarrow \sum_i c_{ij}\, \hat{u}_{j|i}$
        - **For** all capsule j in layer $(l + 1)$: $v_j \leftarrow squash(s_j)$
        - **For** all capsule i in layer l and capsule j in layer $(l + 1)$: $b_{ij} \leftarrow b_{ij} + \hat{u}_{j|i}.\, v_j$
    - **Return** Vj

A custom margin loss function is used in the paper presented to handle multiple classifications in an image. As the mathematics was not entirely understood and able to be transposed to a one dimensional architecture with only two classifications, a custom margin loss was not used, in place of the custom margin loss a binary cross entropy loss function was used. This loss function is the same as used in the CNN and provides a standard loss function available in the Keras library for binary classification problems.

A general Capsule Network is split into different layers, initially convolutional layers are used to create feature maps, these feature maps passed to the primary capsule layer. As the output from the convolutional layer is one dimensional and no agreement is to be calculated here, the routing does not occur at this layer.

The output from the primary capsules are multi-dimensional (8D in the paper presented) and each 8D capsule shares its weights with the other primary capsules. For the secondary capsule layer, each class to be identified has one 16D capsule associated with it and each one receives input from the previous primary capsules. This is where the routing process occurs.

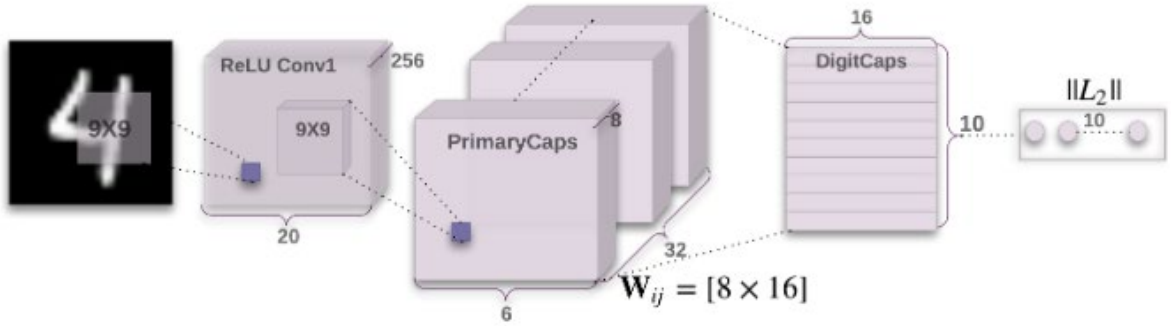


Figure 46: Example of Capsule Network Architecture (Hinton, et al., 2017)

After the secondary capsule layer, a decoder network is utilized. In the case of this project the decoder network is a fully connected dense layer using ReLU activation functions. After the decoder network a single dense layer for output is used with a sigmoid output for classification.

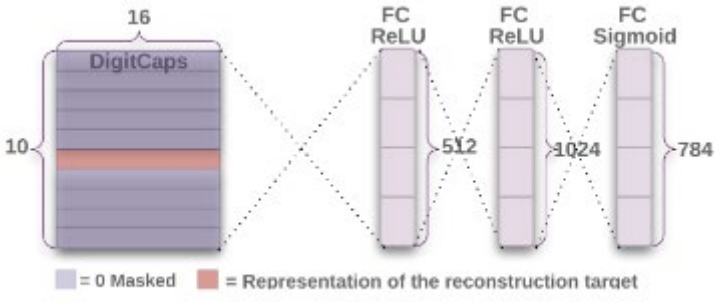


Figure 47: Example of decoder network in Capsule Network (Hinton, et al., 2017)

A major difficulty in the project was changing the application of a Capsule Network from a computer vision type problem to in essence a wave form problem. To do this the network had to be converted from taking 3 dimensional input to 1 dimensional input while keeping the integrity of the routing algorithm. To achieve this, convolutional layers were reduced to 1D layers, the secondary capsules were reduced to 4D instead of 16D and the number of secondary capsules corresponded with the two classes to be identified.

The primary capsule layer consisted of a 1D convolutional layer with an output reshaped by the custom squash function giving 2 outputs, one for each class.

Custom TensorFlow layers were created for the primary capsule, secondary capsule, and squash functions. The squash function was used to correct the output of the primary capsule so it can be passed to the secondary capsule layer, using the squared norm multiplied by scale of the vectors produced by the primary layer. This process drives the length of large vectors to under 1 and the length of small vectors to 0.

For the secondary capsule layer, inbuilt TensorFlow tiling along with matrix multiplication and SoftMax functions were used. The resulting output is then passed through a TensorFlow squeeze function to drop any unneeded axis.

The number of routings applied (layers of secondary capsules), was iterated over to find the best number from 1 to 5 routings. The decoder layer consisted of a dense layer with 512 nodes (similar to the CNN) using a ReLU activation and the final output layer was a single fully connected layer with 1 sigmoid output node.

A masking layer was also created for use during the training process of the network as per Hinton's' paper. This masked layer applies a mask to all the outputs except for the correct output and is used while training for the reconstruction of the input. This reconstruction is then used to minimize the sum of squared differences between the predicted outputs and correct output.

Summary of the architecture to be used in the Capsule Network:

Layer Type:	Parameters	Parameter Values
First Convolutional Layer	Filters, Kernel Size, Activation	112, 5, ReLU
Second Convolutional Layer	Filters, Kernel Size, Activation	256, 5, ReLU
Primary Capsule	Capsules, Channels, Kernel Size, Stride	2, 20, 5, 2
Squash	Vectors, axis	N, -1
Secondary Capsule Layer	Capsules, routings	2, [1-5]
Masking	Used to Mask Y during training	N/A
Decoder Layer	Nodes, Activation	512, ReLU
Output Layer	Nodes, Activation	1, Sigmoid

3.5 Interpretation / Evaluation

Evaluation of the machine learning models is done across multiple metrics; prediction accuracy, recall, precision, F1 score and AUC. A confusion matrix, True Positive Rates (TPR) and False Positive Rates (FPR) are also calculated and produced for each model. These methods of comparison are standard practice for measuring performance metrics of a machine learning model.

		True Class	
		Positive	**Negative**
Predicted Class	**Positive**	True Positive Count (TP)	False Positive Count (FP)
	Negative	False Negative Count (FN)	True Negative Count (TN)

The confusion matrix is a visualization used to display the results of the model based on the correct labels and what the model predicted. Using the values in the confusion matrix produced the discussed metrics are calculated.

Accuracy is the proportional measure of how well the model predicted the true positives and the true negatives combined. The formula for which is:

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

Recall (also known as true positive rate) is used to measure the proportion of the actual true positives correctly classified. The formula for recall is:

$$Recall = \frac{TP}{(TP + FN)}$$

Precision is the proportion of correct true positive classifications. The formula for precision:

$$Precision = \frac{TP}{(TP + FP)}$$

The F1 score is derived from the precision and recall values and is defined as the harmonic mean of precision and recall. A pitfall with the F1 score is it assumes equal importance between precision and recall, therefore a high F1 score is informative to the extent both high precision and recall are likely on a large portion of the classifications, whereas a low F1 score does not give any indication whether precision is low, or recall is low. The formula for the F1 score is:

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

The false positive rate (FPR) is the proportion of false positive calculations. The formula for FPR is:

$$FPR = \frac{FP}{FP + TN}$$

Along with these metrics the area under curve (AUC) is calculated for each model, the AUC makes use of the TPR and the FPR. For the AUC a ROC curve is created where both metrics are computed with multiple thresholds of a logistic regression and plotted as a single graph. The area under the curve in this graph is considered the AUC. A simple AUC score using two points can be calculated using the below formula:

$$AUC = \left(\frac{1}{2}\right) - \left(\frac{FPR}{2}\right) + \left(\frac{TPR}{2}\right)$$

Locally because of a restriction on computing power, McNemar's statistical test is used to measure if there is any significant disagreement differences between the models. Other statistical tests would be possible but with the restriction in the number of times tests can be performed, McNemar's test shows to be the best option (Dietterich, 1998).

McNemar's test is a paired nonparametric statistical hypothesis test and uses a 2x2 contingency table based on the correct and incorrect predictions by each model.

	Model 2 Correct	**Model 2 Incorrect**
Model 1 Correct	Yes / Yes (**a**)	Yes / No (**b**)
Model 1 Incorrect	No / Yes (**c**)	No / No (**d**)

Each cell is the sum of each combination of the Yes/No values in the table above. The McNemar's test specifically tests the marginal homogeneity between the models and is referred to as checking if the disagreements between the two cases match.

Two assumptions of the test are that the two models make predictions on the same data set and there is a count of at least 25 in each cell of the contingency table used for the calculation. The test itself has a chi-squared distribution and 1 degree of freedom.

The null hypotheses (H_0) of the test is that the level of disagreement between is the same. The alternative hypotheses (H_1) is that the level of disagreement between the models is skewed, and they disagree in different ways. A standard alpha of 0.05 is used for the test as there is no medical or safety issues in connection with the results.

If the test statistic (p) is greater than the alpha, we fail to reject the null hypothesis and if the test statistic is less than or equal to the alpha there is a significant difference in the disagreement. To calculate the test statistic the following formula is used: (using the cell designations above)

$$p = \frac{(b - c)\text{^}2}{(b + c)}$$

4.0 Implementation

After the initial exploratory analysis and the resulting transformed data sets created, the remainder of the analysis performed in this project is centred around implementation of the classification networks and their relative performance on the created data set.

Training input to the models consisted of an X variable (Flux readings) and a Y variable (data labels). The data has already been shuffled and separated in the transformation stage of the project. Input shape of training data is X: 4622 x 2000 and Y: 4622. The validation data is also used during the training process and has an input shape of X: 771 x 2000 and Y: 771.

An early stopping call back is used to avoid overfitting of the models. Parameters of the call back are set with the goal of monitoring the loss of each model, recording the minimum and allowing a progression of up to 4 epochs if the loss value begins to increase. In the event this occurs, the contingency is to fall back to the best epoch and restore the weights.

The required input shape into the models was 2000 x 1. Therefore, before the training and validation data is passed it was reshaped as a Numpy array with its dimensions expanded by 1. The new shape of the training data is X: 4622 x 2000 x 1. This gives a single observation the shape of 2,000 x 1 and 4622 observations, accommodating input.

4.1 CNN

A batch size of 64 is used per epoch (Vanderburg & Shallue, 2018). Other batch sizes were attempted but 64 showed to be the best both in computational time and in results of the CNN. A summary of the created CNN can be seen in [*figure 48*].

Layer (type)	Output Shape	Param #
Conv1D-1 (Conv1D)	(None, 2000, 16)	96
Maxpool-1 (MaxPooling1D)	(None, 998, 16)	0
Conv1D-2 (Conv1D)	(None, 998, 32)	2592
Maxpool-2 (MaxPooling1D)	(None, 497, 32)	0
Conv1D-3 (Conv1D)	(None, 497, 64)	10304
Maxpool-3 (MaxPooling1D)	(None, 247, 64)	0
Conv1D-4 (Conv1D)	(None, 247, 128)	41088
Maxpool-4 (MaxPooling1D)	(None, 122, 128)	0
Conv1D-5 (Conv1D)	(None, 122, 256)	164096
Maxpool-5 (MaxPooling1D)	(None, 60, 256)	0
flatten_29 (Flatten)	(None, 15360)	0
Fully_Connected-1 (Dense)	(None, 512)	7864832
Fully_Connected-2 (Dense)	(None, 512)	262656
Fully_Connected-3 (Dense)	(None, 512)	262656
Fully_Connected-4 (Dense)	(None, 512)	262656
Sigmoid_Output (Dense)	(None, 1)	513

Total params: 8,871,489
Trainable params: 8,871,489
Non-trainable params: 0

Epoch 1/50

Figure 48: CNN Summary

The CNN was trained over a default of 50 epochs as well as using the call back metric to monitor loss in an attempt to avoid any over fitting. 50 epochs were chose as again it was part of the design from the state of the art version identified previously and all attempts were made to reproduce the work as accurately as possible to allow comparison with the Capsule Network.

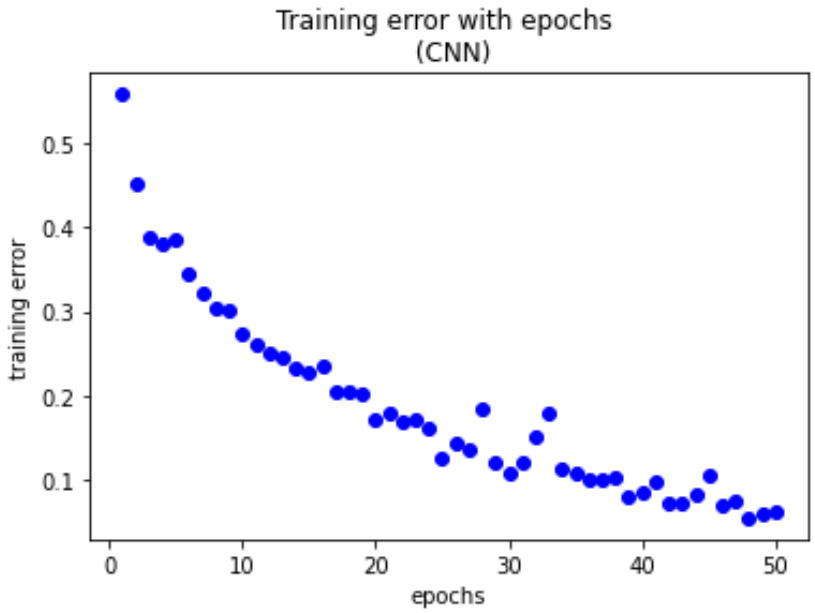


Figure 49: Training loss recorded per epoch

Loss (error) was recorded throughout each epoch and plotted. From [*figure 49*] above it can be seen that loss had minor increases at the 5th and 15th epoch. A series of increasing loss can be seen at the 30th epoch but doesn't reach the 4 epoch threshold set in the call back. The loss continued to decline to the 50th epoch. Model accuracy was also recorded throughout the training process and plotted in [*figure 50*].

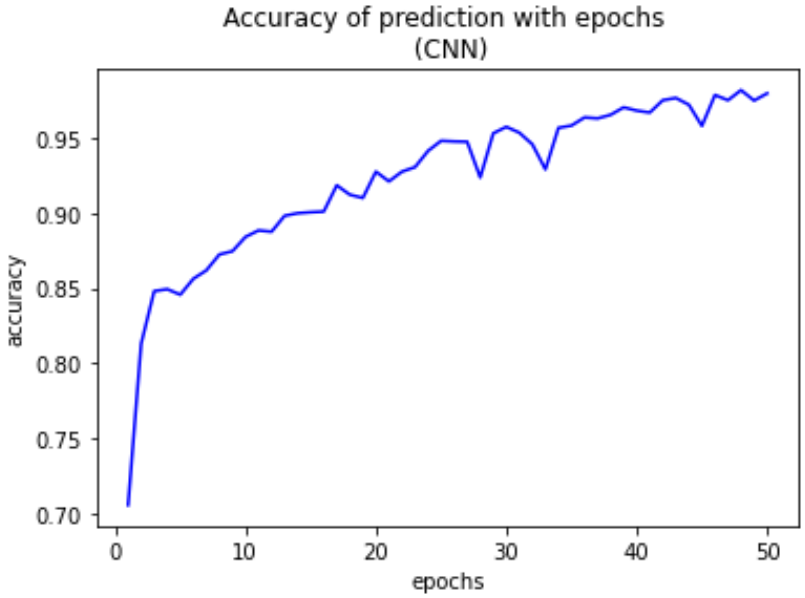


Figure 50: Training accuracy recorded per training epoch

From the accuracy graph produced, the model accuracy does increase over each epoch with exceptions at the same epoch range where loss also increased. Some instability in the model accuracy can be seen over the last 6 epochs of the training with a training accuracy of 0.98 reached. Validation accuracy was also used as part of training the model with a maximum validation accuracy reached at the 50th epoch of 0.95. Loss was used as the metric of emphasis in training the model as accuracy relies on the binary True/False labelling whereas the loss metric assigns lower loss to predictions that are closer to the class label and can be argued as the better metric when training a model as it implies how well a model is working after each optimization.

Class weights were created earlier in the transformation stage but as the model performed well in the training these weights were not applied at this stage. If the model did not behave as expected in the prediction stage, then these training steps would be revisited using the weights during the training process. The trained model was saved in H5 format used to make predictions on the test set.

4.2 Hypermodel CNN

Using the Hypermodel tuning options and parameters set in the model creation criteria the Hypermodel cycled through 180 possible model combinations over a 2 hour period twice. In total 360 models were tested taking approx. 4 hours.

```
Trial 180 Complete [00h 04m 55s]
accuracy: 0.9409346580505371

Best accuracy So Far: 0.9755517244338989
Total elapsed time: 01h 58m 49s
INFO:tensorflow:Oracle triggered exit
```

Figure 51: Hypermodel iteration 1 tuning results

```
Best accuracy So Far: 0.9755517244338989
Total elapsed time: 01h 53m 54s

Search: Running Trial #180

Hyperparameter    |Value           |Best Value So Far
conv_blocks       |4               |5
filters_0         |224             |48
pooling_0         |max             |avg
hidden_size       |70              |30
dropout           |0.1             |0.2
learning_rate     |0.00058891      |0.00016255
filters_1         |160             |64
pooling_1         |max             |avg
filters_2         |160             |96
pooling_2         |avg             |max
filters_3         |144             |160
pooling_3         |avg             |max
filters_4         |240             |16
pooling_4         |max             |max
tuner/epochs      |30              |30
tuner/initial_e...|0               |10
tuner/bracket     |0               |2
tuner/round       |0               |2
```

Figure 52: Hypermodel tunning progress

The best training accuracy achieved throughout the tunning process was recorded at 0.97 with the resulting model summary outputted.

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv1d_10 (Conv1D)           (None, 2000, 176)         704

 max_pooling1d_10 (MaxPoolin  (None, 998, 176)          0
 g1D)

 conv1d_11 (Conv1D)           (None, 996, 160)          84640

 max_pooling1d_11 (MaxPoolin  (None, 496, 160)          0
 g1D)

 conv1d_12 (Conv1D)           (None, 486, 32)           56352

 max_pooling1d_12 (MaxPoolin  (None, 241, 32)           0
 g1D)

 conv1d_13 (Conv1D)           (None, 231, 208)          73424

 max_pooling1d_13 (MaxPoolin  (None, 114, 208)          0
 g1D)

 conv1d_14 (Conv1D)           (None, 110, 208)          216528

 max_pooling1d_14 (MaxPoolin  (None, 53, 208)           0
 g1D)

 flatten_2 (Flatten)          (None, 11024)             0

 dense_10 (Dense)             (None, 64)                705600

 dense_11 (Dense)             (None, 352)               22880

 dense_12 (Dense)             (None, 192)               67776

 dense_13 (Dense)             (None, 256)               49408

 dense_14 (Dense)             (None, 1)                 257

=================================================================
Total params: 1,277,569
Trainable params: 1,277,569
Non-trainable params: 0
```

Figure 53: Hypermodel CNN summary

The architecture for the created CNN through Hypermodel tunning was significantly different from the CNN used in the previous section.

The number of layers to be used was the same but the parameters at each layer were different in terms of number of filters and kernel sizes to be used in convolutional layers and number of nodes to be used in fully connected dense layers.

The max pooling layers used the same window and stride sizes, but the learning rate was determined to be best at 0.0001 instead of the 0.001 used in the previous CNN model.

Summary of the architecture determined by the Hypermodel tunning:

Layer Type:	Parameters	Parameter Values
First Convolutional Layer	Filters, Kernel Size, Activation	176, 11, ReLU
Max Pooling Layer	Window size, Stride	5, 2
Second Convolutional Layer	Filters, Kernel Size, Activation	160, 5, ReLU
Max Pooling Layer	Window size, Stride	5, 2
Third Convolutional Layer	Filters, Kernel Size, Activation	32, 3, ReLU
Max Pooling	Window size, Stride	5, 2
Fourth Convolutional Layer	Filters, Kernel Size, Activation	208, 5, ReLU
Max Pooling	Window size, Stride	5, 2
Fifth Convolutional Layer	Filters, Kernel Size, Activation	208, 5, ReLU
Max Pooling	Window size, Stride	5, 2
Flatten	None	
Fully Connected Layer 1	Nodes, Activation	64, ReLU
Fully Connected Layer 2	Nodes, Activation	352, ReLU
Fully Connected Layer 3	Nodes, Activation	192, ReLU
Fully Connected Layer 4	Nodes, Activation	256, ReLU
Output Layer	Nodes, Activation	1, Sigmoid

The new hyper tuned model was then trained using the same methodology as the previous CNN over the same number of epochs and using the same training and validation data. The same call back metric monitoring loss was used and training accuracy along with training loss was recorded throughout each epoch with validation loss and accuracy being tested.

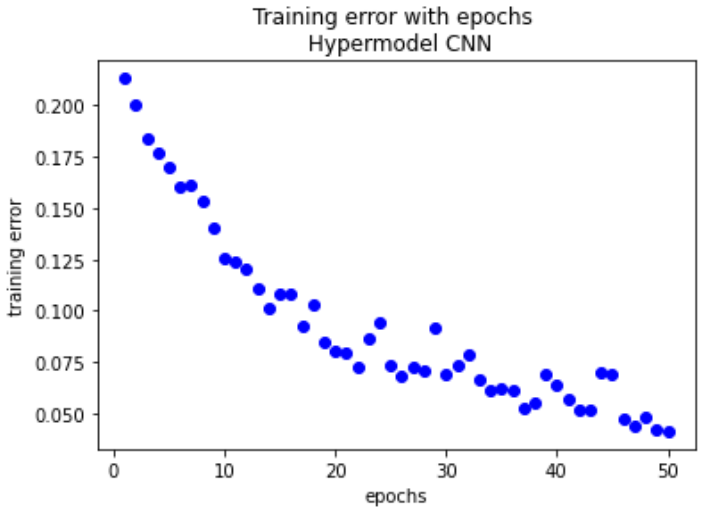


Figure 54: Training loss recorded per epoch

During training of the Hypermodel CNN the loss starts to show instability around the 15th epoch but never reaches the threshold set in the call back function of 4 continuous declines in loss. The maximum training accuracy reached at this point by the Hypermodel version was 0.98, greater than the 0.97 reached during the tuning process. Validation accuracy was also recorded with the highest validation accuracy of 0.89 reached at the 50th epoch. Validation loss though was significantly higher than training loss with a minimum validation loss of 0.29 reached.

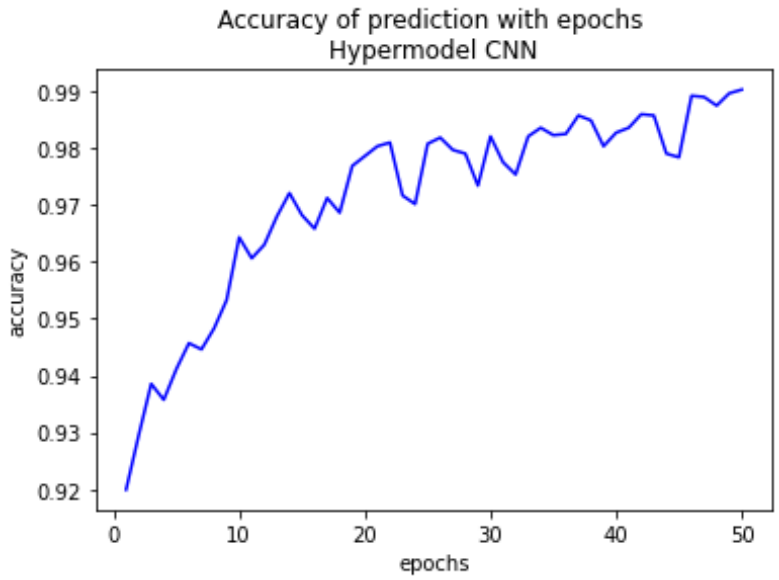


Figure 55: Training accuracy recorded per training epoch

From [*figure 55*] it can be seen that the training accuracy fluctuates greatly after the 15th epoch and shows instability in the model throughout the training process. The trained model was saved in H5 format and to make predictions on the test data.

4.3 Capsule Network

Implementation of the Capsule Network was slightly different to the CNN. The same training and test data sets were used, and input shapes managed using the same methods, but during the training phase no validation data was provided to the model.

Another difference was that before training data can be provided to the Capsule Network the Y variable (Labels) of the training set had to be changed from an integer to a categorical variable to allow for predictions to be made and checked in the secondary capsule layer in accordance with the number of classes selected.

As the number of routings (secondary capsule layers) that performed best with the provided data was not known, 5 Capsule Networks were generated each with a different number of secondary capsule layers ranging from 1 to 5. Each Capsule Network was trained over the same number of epochs as the previous models with the same call back metric monitoring loss. The Capsule network produces three loss metrics, the Capsule Network loss, the decoder loss, and the overall loss. Overall loss was chosen as the metric to monitor.

A batch size of 64 was again chosen per epoch with a learning rate of 0.001 to keep all measures taken during training as equal as possible across models, with training accuracy and loss recorded over each epoch. A summary of the Capsule Network can be seen in [*figure 56*].

Layer (type)	Output Shape	Param #	Connected to
input_30 (InputLayer)	[(None, 2000, 1)]	0	[]
conv1 (Conv1D)	(None, 1996, 112)	672	['input_30[0][0]']
max_pooling1d_10 (MaxPooling1D)	(None, 499, 112)	0	['conv1[0][0]']
conv2 (Conv1D)	(None, 499, 256)	143616	['max_pooling1d_10[0][0]']
primarycap_conv1d (Conv1D)	(None, 250, 40)	51240	['conv2[0][0]']
primarycap_reshape (Reshape)	(None, 5000, 2)	0	['primarycap_conv1d[0][0]']
primarycap_squash (Lambda)	(None, 5000, 2)	0	['primarycap_reshape[0][0]']
secondary_caps (CapsuleLayer)	(None, 2, 2)	40000	['primarycap_squash[0][0]']
mask_18 (Mask)	(None, None)	0	['secondary_caps[0][0]']
input_31 (InputLayer)	[(None, 2)]	0	[]
capsnet (Length)	(None, 2)	0	['secondary_caps[0][0]']
decoder (Sequential)	(None, 2000, 1)	1028560	['mask_18[0][0]']

Total params: 1,264,088
Trainable params: 1,264,088
Non-trainable params: 0

Figure 56: Capsule Network Summary

A Capsule Network with 1 routing was the first model to make use of the call back function and stopped training at epoch 20 rolling back to epoch 17.

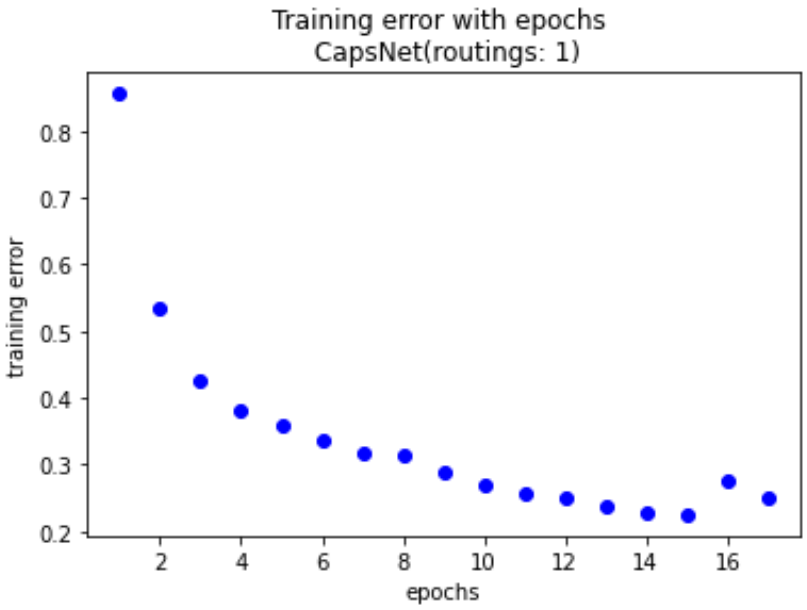


Figure 57: Training loss recorded per epoch with 1 routing

The training loss declined sharply after the first epoch and shows a steady rate of decline to epoch 15. Training accuracy over the 17 epochs didn't show as much promise as with the CNN with a peak accuracy reached at epoch 15 of 0.88. Although a continuous increase can be seen in [*figure 58*] the model began to suffer in accuracy after the 15th epoch.

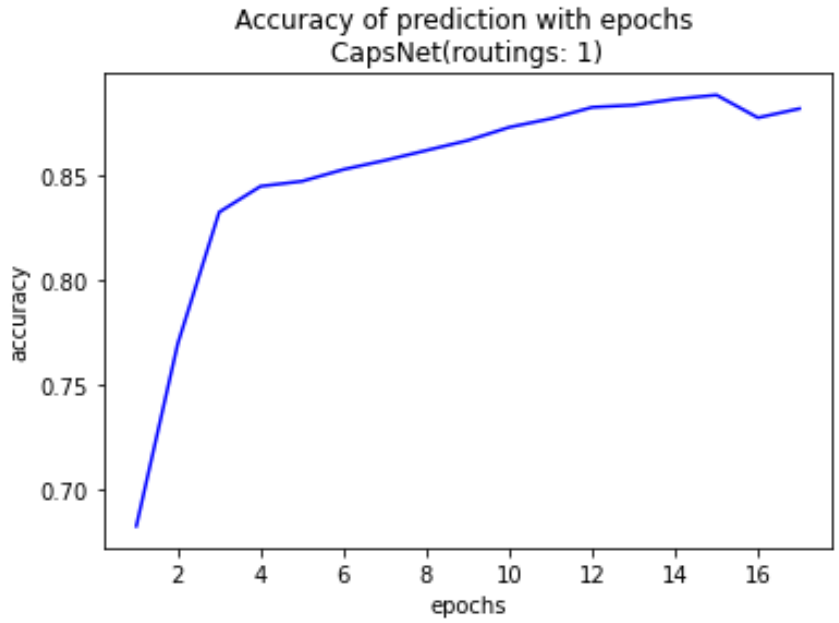


Figure 58: Training accuracy recorded per training epoch with 1 routing

The Capsule network with 2 routings ran best over 20 epochs until an increase in loss started causing the training process to stop and roll back to epoch 20.

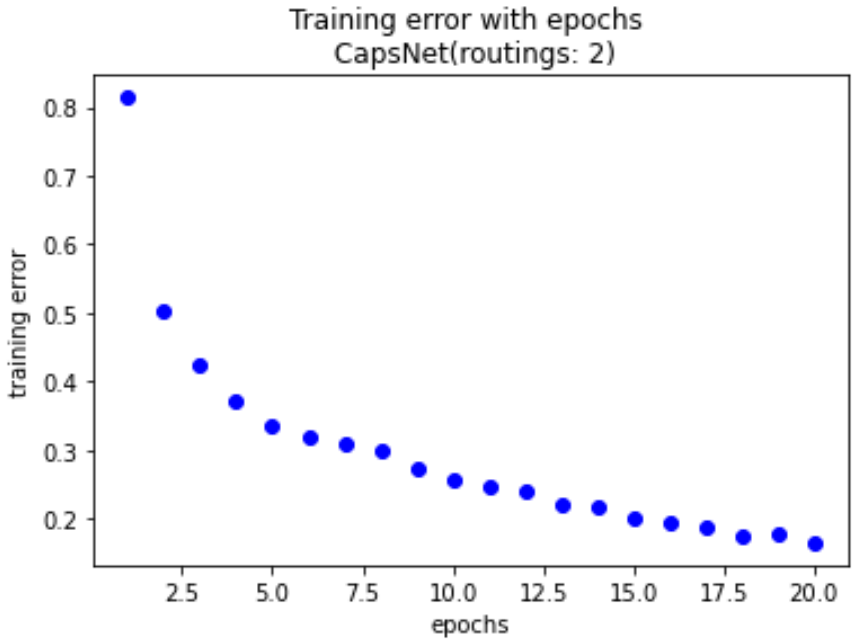


Figure 59: Training loss recorded per epoch with 2 routings

Again, a continuous steady decline in loss can be seen in [*figure 59*] with a sharp decline after the first epoch.

With 2 routings the Capsule Network displayed better performance in training accuracy reaching a peak of 0.91

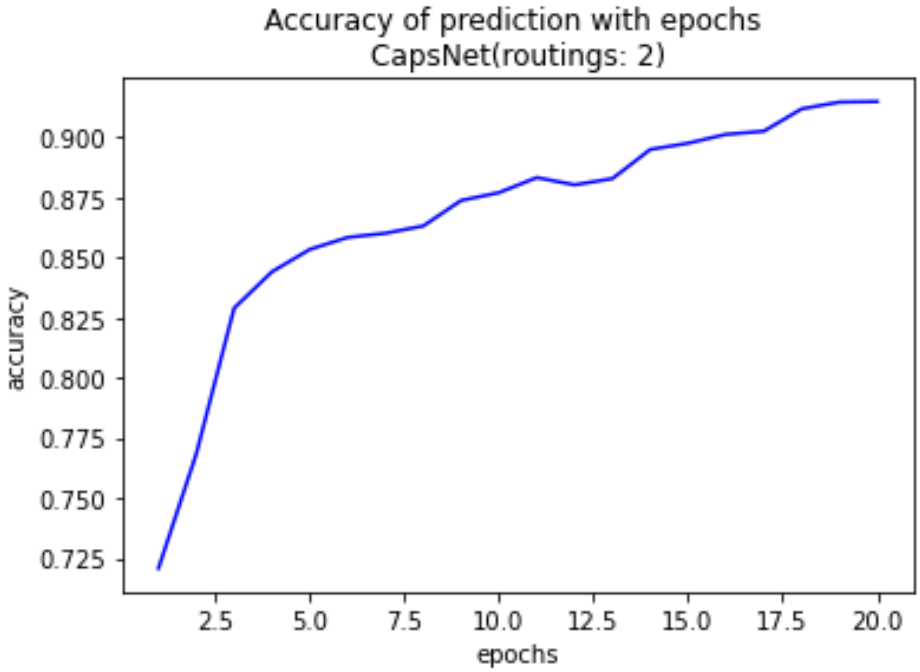


Figure 60: Training accuracy recorded per training epoch with 2 routings

3 routings was used next and displayed the most consistency in decline of loss over 20 epochs. At this point in the testing anything above 20 epochs displayed an increase in loss and triggered the call back function in training.

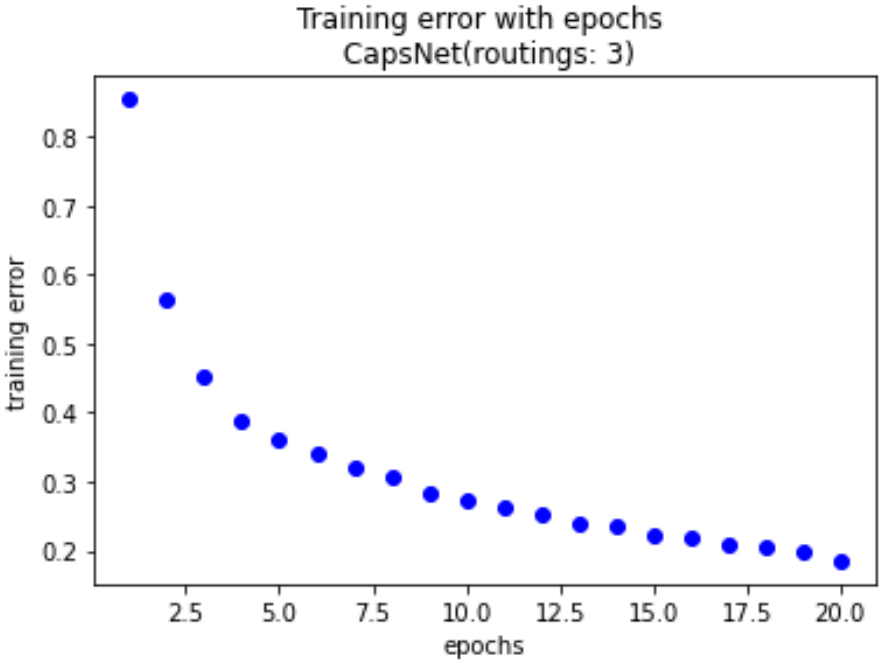


Figure 61: Training loss recorded per epoch with 3 routings

The use of 3 routings displayed similar accuracy to 2 routings with its peak at the same point over 20 epochs but did display a low accuracy over the first two epochs.

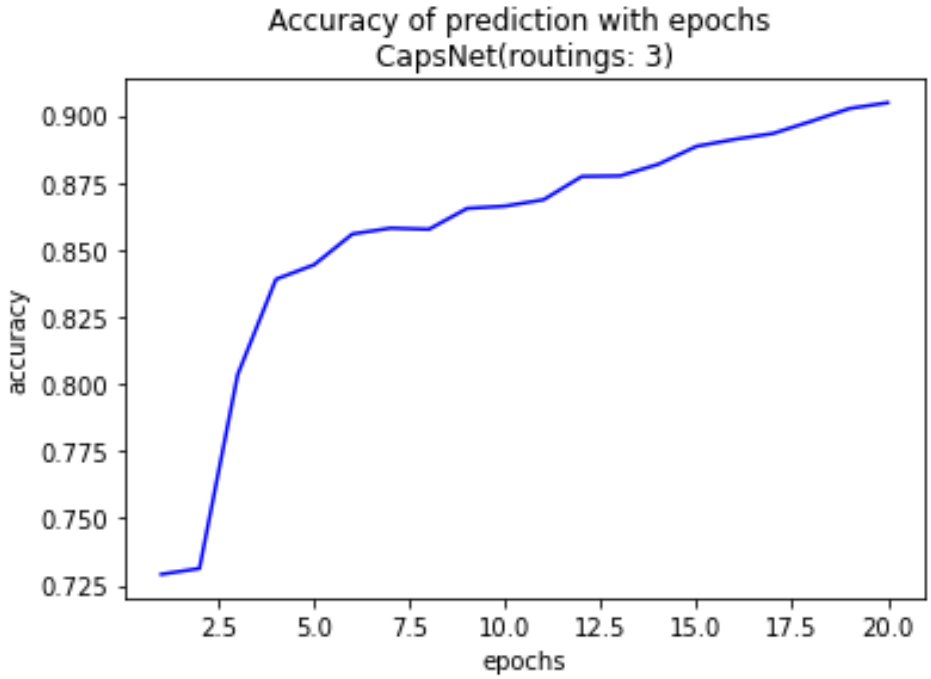


Figure 62: Training accuracy recorded per training epoch with 3 routings

A Capsule Network with 4 routings returned the worst training metrics with loss starting to increase after the 11th epoch with a substantial jump in loss on the 10th epoch.

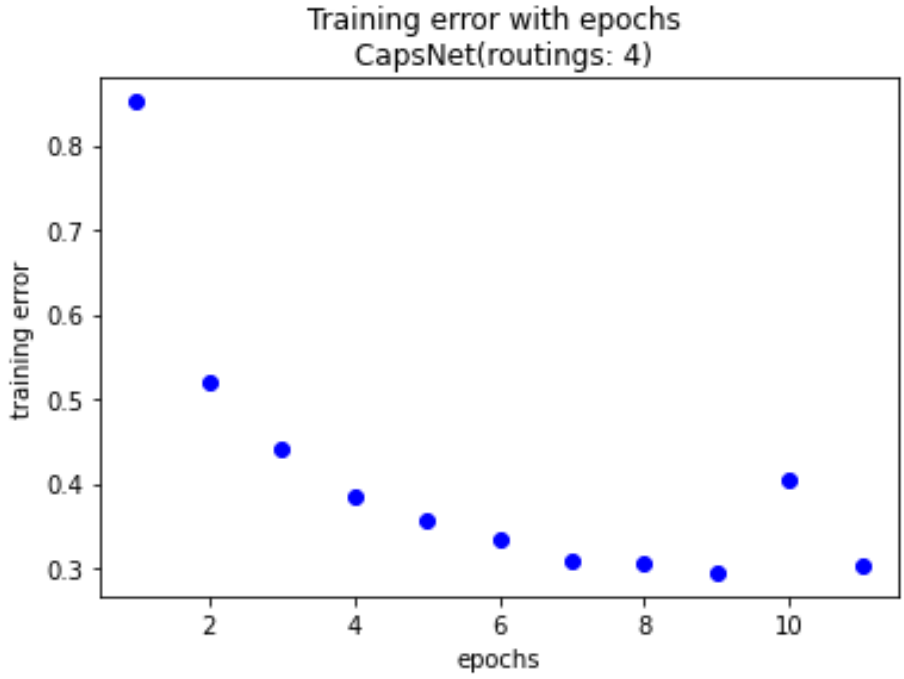


Figure 63: Training loss recorded per epoch with 4 routings

Training accuracy for a Capsule Network with 4 routings was also the worst performing, reaching a peak of 0.86 over the 11 epochs.

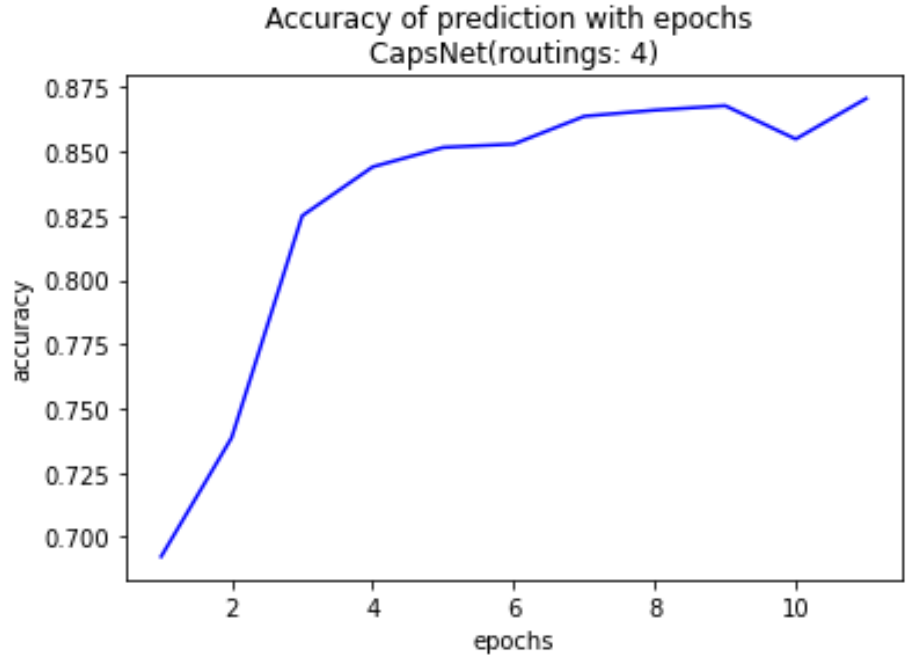


Figure 64: Training accuracy recorded per training epoch with 4 routings

For the Capsule Network with 5 routings, again the call back function was triggered at the 20th epoch. With a similar decline in loss to the 2 and 3 routings versions.

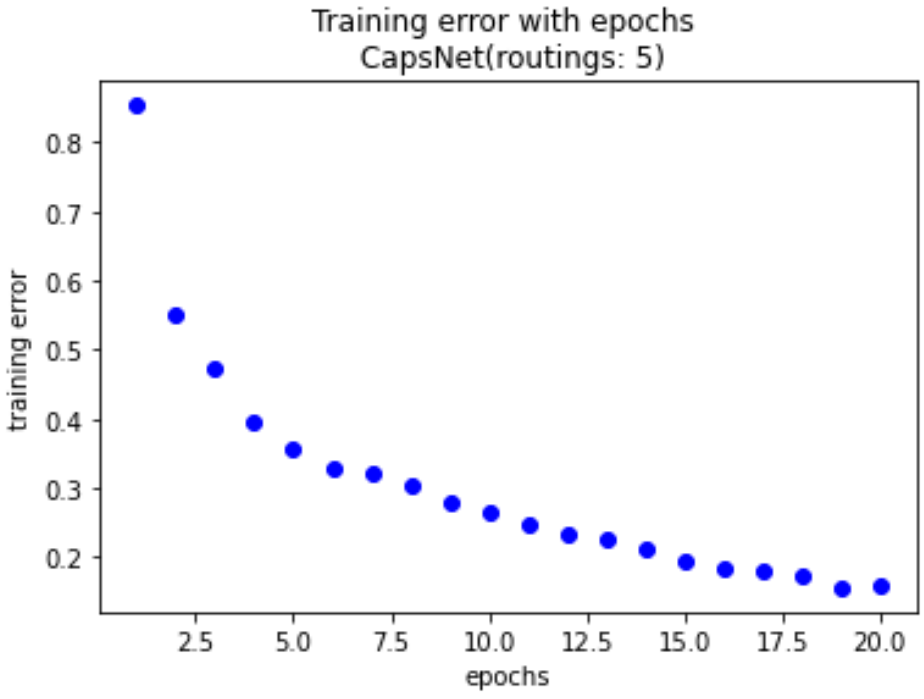


Figure 65: Training loss recorded per epoch with 5 routings

Training accuracy of the Capsule Network with 5 routings was again similar to the same as versions with 2 and 3 routings with the same peak accuracy achieved at the 20 epochs.

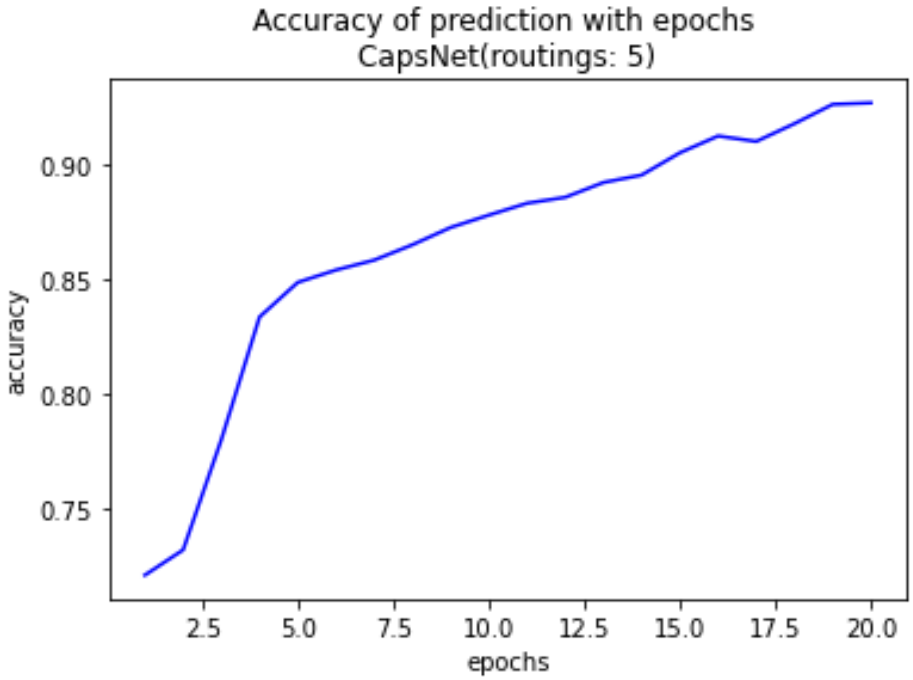


Figure 66: Training accuracy recorded per training epoch with 5 routings

Over the course of training the Capsule Networks with the various routings the results showed that any more than 20 epochs would result in the loss beginning to increase which was in contrast to the CNN which ran the entire 50 epochs.

The Capsule Networks did show more stability in the loss decline and accuracy increase over each epoch, but the training accuracy results were far below the results of the first CNN and the Hypermodel CNN. Although Capsule Networks with 2, 3 and 5 routings provided the best training results, all five instances were used for predictions on the test data.

Training Results

Model	Best Accuracy	Best Loss
CNN	0.98	< 0.1
Hypermodel	0.98	<0.05
Capsule Network (2 routings)	0.91	0.15
Capsule Network (3 routings)	0.91	0.18
Capsule Network (5 routings)	0.91	0.15

5.0 Results

Each of the models created were used to make predictions on the same test set with there accuracy, precision and recall recorded as well as the prediction made versus the true label in the test data. The predictions made against the true label were used to create contingency tables which were then used in the McNemar's calculation for testing the level of disagreement between the models.

Accuracy, precision and recall metrics were recorded using the Python package Scikit-Learn and the F1 score was calculated manually from these metrics. Confusion Matrix and ROC curve were plotted for each model with the AUC calculated. The false positive rates and true positive rates were calculated manually using the output from each confusion matrix.

5.1 CNN

The metrics recorded on the CNN predictions using the test set of data, along with the calculated F1 score and confusion matrix were as follows.

CNN Results	
Metric	**Test Data**
Accuracy	**0.95**
Precision	**0.91**
Recall	**0.90**
F1 score	**0.90**

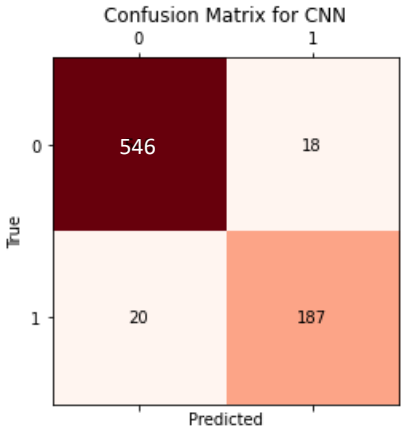


Figure 67: Confusion Matrix for CNN

The ROC curve was also plotted and an AUC of 0.937 was calculated. The True Positive Rate (TPR), as known as recall was calculated at 0.90, the same as above from the SK-Learn metrics and the False Positive Rate (FPR) was calculated at 0.03. It should be noted that the CNN created from the architecture presented in (Vanderburg & Shallue, 2018) performed as expected with 0.95 accuracy but did not perform to the same 0.985 AUC score presented in the paper.

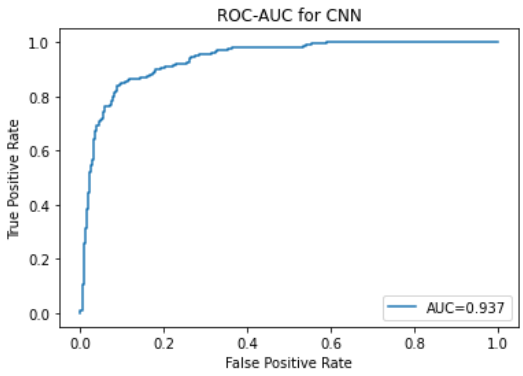


Figure 68: ROC graph for CNN

5.2 Hypermodel CNN

The Hypermodel predictions on the test data were recorded and metrics calculated along with the confusion matrix created are as follows.

Hypermodel CNN Results	
Metric	**Test Data**
Accuracy	**0.90**
Precision	**0.82**
Recall	**0.84**
F1 score	**0.83**

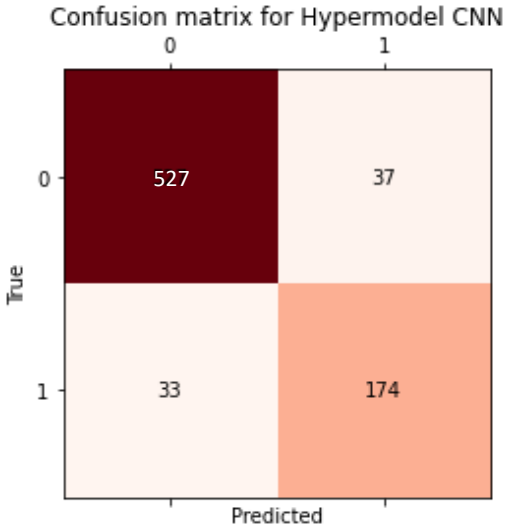


Figure 68: Confusion Matrix for Hypermodel CNN

The ROC curve for the Hypermodel was plotted and an AUC of 0.887 was calculated. The True Positive Rate (TPR) was calculated at 0.84, and the False Positive Rate (FPR) was calculated at 0.06

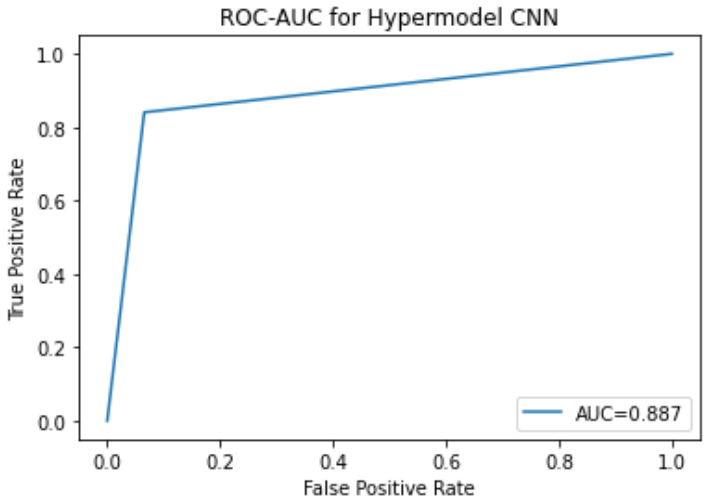


Figure 69: ROC graph for Hypermodel CNN

5.3 Capsule Network

All five Capsule Networks were used to make predictions on the test data. The Capsule Network with 3 routings performed the best in terms across all metrics and its results are shown below. The results from the other four Capsule Networks are presented in the appendix. All models scored a perfect precision score showing that the Capsule Network done a better job at identifying false positive exoplanets correctly.

Capsule Network (3 routings) Results	
Metric	**Test Data**
Accuracy	**0.96**
Precision	**1**
Recall	**0.85**
F1 score	**0.92**

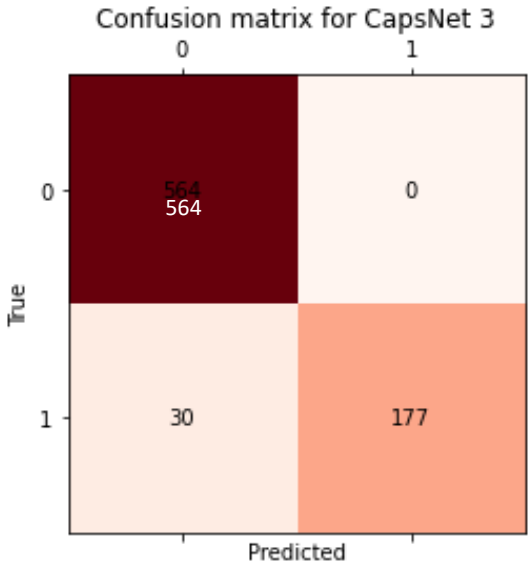


Figure 70: Confusion Matrix for Capsule Network-3 routings

The ROC curve for the best Capsule Network (3 routings) was plotted and an AUC of 0.93 was calculated. The True Positive Rate (TPR) was calculated at 0.85, and the False Positive Rate (FPR) was calculated at 0 due to the perfect precision displayed by the model.

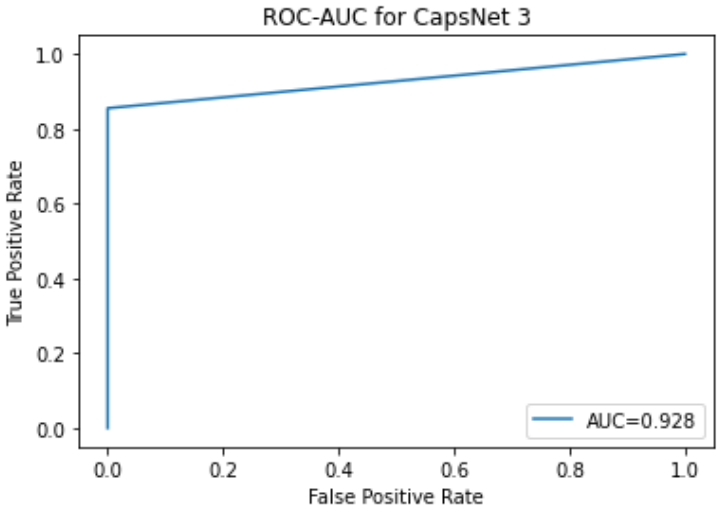


Figure 4: ROC graph for Capsule Network - 3 routings

5.4 McNemar's Test

As the Hypermodel version of the CNN scored significantly less than the other two models in the analysis and that the Hypermodel was a secondary to the original CNN it was left out of calculations for McNemar's test. This test is a statistical test used to determine the level of disagreement between the two models. For the test the alpha value was set at 0.05.

If **p > alpha** we fail to reject H_0, determining there is no difference in the level of disagreement. Alternatively, if **p <= alpha** we reject H_0, and determine there is a significant difference in the disagreement between the two models.

Using the results of the predictions a contingency table was created:

	Capsule Net Correct	Capsule Net Incorrect
CNN Correct	723	10
CNN Incorrect	18	20

The resulting test statistic calculated was 10 with 1 degree of freedom resulting in a p-value of 0.185. Meaning, because **p** (0.185) **> alpha** (0.05) the test concludes that there is the same proportion of errors between both models and the null hypothesis is failed to be rejected.

5.5 Overall

Comparing the two best models (CNN and Capsule Network), the overall results show that the Capsule Network performed better across all metrics with exception of recall and AUC.

The Capsule Network scored perfect precision meaning that it correctly identified all the false positive exoplanets in the data set. The CNN had better recall of 0.90 to the Capsule Networks 0.85 meaning that the CNN correctly classified 90% of known exoplanets while the Capsule Network correctly identified 85% of known exoplanets.

The Capsule Network returned a better F1 score of 0.92 to the CNNs 0.90 while the CNN returned a better AUC of 0.937 to the Capsule Networks 0.928. There was no significant difference in the proportion of disagreement between the two models in McNemar's test with a p-value of 0.185. The overall results of the analysis show that on making predictions on the test data:

- The Capsule Network performed better in Accuracy, Precision and F1 Score
- The CNN performed better in Recall and AUC.

Both Models Performance

Metric	CNN on Test Data	Capsule Network on Test Data
Accuracy	**0.95**	**0.96**
Precision	**0.91**	**1**
Recall	**0.90**	**0.85**
F1 score	**0.90**	**0.92**
AUC	**0.937**	**0.928**
TPR / FPR	**0.90 / 0.03**	**0.85 / 0.0**

6.0 Conclusions

Capsule Networks were designed in an effort to address the issues present in CNNs where possible feature loss can occur during the pooling stages of the network, through the use of dynamic routing. While the Capsule Network created in this project does display numerical superiority in some respects on the given data set over the state of the art CNN, it does not represent a statistical improvement and cannot be assumed it will be equally successful when applied to all future instances of this type of data. Although Capsule Networks were originally designed for computer vision tasks, they demonstrate the potential to be equally as successful as CNNs when applied in different problem domains.

A limitation to be noted in the creation of a Capsule Network is the need for custom built layers and calculations in the model. Although this can be considered a strength as it gives more flexibility in its design, the limitation appears with the advent of machine learning packages such as TensorFlow and Keras, where a convolutional neural network can be instantiated with relatively less effort and deployed quicker in a time sensitive environment with matching results. Another limitation of the Capsule Network implementation was the lack of a custom margin loss function as created in Hinton's paper (Hinton, et al., 2017). The loss function used instead was standard binary cross entropy and it isn't seen in this analysis if a custom loss would be more beneficial. A strength of the Capsule Network over the CNN is the number of epochs needed to train the models to reach similar test accuracy. Where the CNN trained for 50 epochs the Capsule Network trained to 20 epochs, although showing lower training scores, the Capsule Network produced better scores in accuracy, precision and F1 score on the test data.

In regard to the Hypermodel created in the project, a disadvantage was the time and computing constraints that meant only a number of possible configurations could be tested. The main strength of the Hypermodel though, was the ability to achieve significant scores within the constraints of these tunning parameters applied.

From the results presented in the analysis, it is the conclusion of the project that if a model is to be applied to a particular problem domain where CNNs have shown to be useful, and development time allows, Capsule Networks demonstrate the ability to address potential problems in CNNs and therefore should be considered a valid research path.

7.0 Further Development or Research

With additional time and resources, this project would benefit from testing of each model across multiple data sets in different configurations to conclude the model's effectiveness in general application. As discussed in the limitations of the project where a custom margin loss function was not created, with further work and a deeper understanding of the mathematics behind the custom margin loss function, it could be applied with benefit to the Capsule Network and improve its performance. Even though the CNN architecture used did provide comparative results to the state of the art CNN it was based upon, with more testing and development time available in tuning the Hypermodel version, it would be worth exploring if a different configuration was produced that would match or improve on the results of the CNN.

8.0 References

Albawi, S., Mohammed, T. A. & Al-Zawi, S., 2017. *Understanding of a Convolutional Neural Network.* Antalya: IEEE.

Dietterich, T. G., 1998. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Comput,* p. 1895–1923.

Fayyad, U., Piatetsky-Shapiro, G. & Smyth, P., 1996. *Knowledge Discovery and Data Mining: Towards a unifying framework.* s.l., s.n.

Hinton, G. E., Frosst, N. & Sabour, S., 2017. Dynamic routing between capsules. *Advances in neural information processing systems,* Volume 30.

Hinton, G. E., Sabour, S. & Frosst, N., 2018. *Matrix Capsules with EM Routing.* Toronto, ICLR.

Li, L. & Jamieson, K., 2018. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization.. *Journal of Machine Learning Research,* Volume 18, pp. 1-52.

NASA, 2016. *20 Inventions we wouldnt have without space travel.* [Online]
Available at: https://www.jpl.nasa.gov/infographics/20-inventions-we-wouldnt-have-without-space-travel
[Accessed 10 12 2021].

Osborn, H. P. et al., 2019. *Rapid classification of TESS planet candidates with convolutional neural networks,* Paris: EDP Sciences.

Sarmento, R. & Costa, V., 2017. *Comparative Approaches to Using R and Python for Statistical Data Analysis.* Hershey: IGI Global.

Stuart, M., 2019. *Towards Data Science.* [Online]
Available at: https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac
[Accessed 03 05 2022].

Valizadegan, H. et al., 2021. ExoMiner: A Highly Accurate and Explainable Deep Learning Classifier that Validates 301 New Exoplanets. *American Astronomical Society,* Volume 53, pp. 108-06.

Vanderburg, A. & Shallue, C. J., 2018. Identifying exoplanets with deep learning: A five-planet resonant chain around kepler-80 and an eighth planet around kepler-90. *The Astronomical Journal,* 155(2), p. 94.

Wickham, H., 2014. Tidy Data. *Journal of Statistical Software,* 59(10), pp. 1-23.

Zafar, I. et al., 2018. *Hands-On Convolutional Neural Networks with TensorFlow Solve computer vision problems with modeling in TensorFlow and Python..* 1 ed. Birmingham: Packt Publishing ltd.

9.0 Appendices

Results of the other Capsule Networks created not used in final model comparison:

Capsule Network (1 routings) Results	
Metric	Test Data
Accuracy	0.95
Precision	1
Recall	0.82
F1 score	0.90

Capsule Network (2 routings) Results	
Metric	Test Data
Accuracy	0.95
Precision	1
Recall	0.81
F1 score	0.89

Capsule Network (4 routings) Results	
Metric	Test Data
Accuracy	0.95
Precision	1
Recall	0.82
F1 score	0.90

Capsule Network (5 routings) Results	
Metric	Test Data
Accuracy	0.94
Precision	1
Recall	0.81
F1 score	0.89

National College of Ireland

Project Proposal

Discovery 2: An analysis of Machine Learning methods to predict exoplanet candidates

01/11/2021

BCHCE Computing (Evening)

Data Analytics

2021/2022

Jonathan Flanagan

x18143890

x18143890@sudent.ncirl.ie

Contents

Objectives

The main objective of my project is to investigate the most effective machine learning method used to classify exoplanet candidates from Astronomical Photometric Data between Convolutional Neural Networks and Capsule Neural Networks. To do this I will be testing their comparative effectiveness in terms of prediction accuracy, recall, precision and F1 score as well as using McNemar's or a 5×2 cv test for statistical difference, depending on the computing power available.

The project is driven by the hypothesis that there is a significant statistical difference in the performance of Capsule networks over Convolutional Neural Networks in classifying exoplanet candidates. The main objective can be broken down into several other objectives:

Data Collection

Data will be gathered from public datasets provided by the Mikulski Archive for Space Telescopes (MAST). All datasets are part of the Kepler space telescope mission. And will be collected through an API and webservices provided by MAST.

1. Data Preparation

There will be several steps to the overall preparation of the data, Kepler mission ID's will need to be extracted from the main list of objects of interest, pixel images will need to be downloaded using these Kepler ID's and light curve information will need to be extracted from images. The extracted light curves will need to be cleaned of possible instrumentation noise and other artifacts, normalised, and folded on their respective orbital period times and labelled for classification, so data can be split and used for training and testing with the machine learning models.

2. Model Creation

Two machine learning models will be developed (Capsule Neural Network and Convolutional Neural Network). I've chosen these two as Capsule Networks are a newer development and have shown in other cases to be better performing in accuracy to CNN's (Jaiswal, 2018). CNN's have been used in the past on these datasets with good results (97% average precision and 92% accuracy on planets in the two-class model) (H. P. Osborn, 2019)

3. Model Assessment

Assessment of the created models will be based on their level of accuracy in predicting candidates as well as their respective level of accuracy, recall, precision, and F1-Score. For local testing, a contingency table will then be created and used to calculated McNemar's statistic to evaluate if there is any significant statistical difference in the two models. For cloud deployment it may be possible to apply a 5×2 cv test.

4. Findings Report

The findings report will be the culmination of the project and provide insights on the analysis conducted throughout.

Background

Machine Learning has increased in popularity in recent years, with many developers coming up with real world applications for Machine Learning that can help gain insights from the ever increasing amounts of data being produced.

For my project, I plan to analyse the use of ML on one of those such use cases, the classification of exoplanet candidates. I chose this direction for my project as I have an interest in the capabilities of machine learning and its use cases as well as astronomy. When it comes to ever increasing amounts of data and ways to extract valuable information I find the combination of Space, Data and Machine Learning to be one of the most appealing.

Currently the most popular means to identify exoplanet candidates are by; Transit (75.5%) When a planet passes between the observer and its star, it dims the stars light by a measurable amount. Radial Velocity (19.4%) Orbiting planets cause stars to wobble, causing an observable shift in the colour of the stars light, Microlensing (2.6%) as a planet passes between the star and an observer the stars light is focused by the gravity of the planet and is measurable and Direct Imaging (1.2%) using special techniques that remove glare from orbiting stars exoplanets can be photographed. (NASA, 2021)

For my project I will be focussing on the Transit method of detection from the Kepler mission.

State of the Art

Searching for exoplanet candidates using machine learning is not a new concept. Some examples are in 2018 Christopher J. Shallue, and Andrew Vanderburg analysed exoplanet candidates using CNN's, Fully Connected Networks and Linear methods, measuring their models against current vetting processes. On test sets, the model ranked true planet candidates above false positives 98.8% of the time (Vanderburg, 2018) this analysis was focussed more on vetting candidates rather than discovering candidates from light curve samples.

A similar analysis was carried out in 2019 looking at Convolutional Neural Networks and what combination when applied gave the best results. The results of this analysis indicated that a two-dimension convolutional neural network would be an excellent choice for transit analysis, with all models with folding having an accuracy above 98% on test/training sets. (Jiang, 2019)

Another relevant analysis to my project was performed to measure the performance of CNNs versus Capsule Networks in terms of semi-supervised classification on the MNIST dataset where the Error rate on n = 10,000 for CNN GAN's was 0.0702 whereas CapsuleGANs was 0.0531 (Jaiswal, 2018) , a significant difference.

My project hopes to build on top of these analyses and apply Capsule Networks on the Kepler datasets and measure their performance. From my research I have not been able to find where this analysis has been completed, even though I will be using the Kepler datasets another similar mission, TESS, is still on going and providing new data, therefore this type of

analysis can provide an insight into the usage of newer models such as Capsule Networks and evaluate its usefulness on exoplanet candidate identification.

Data

The data from this project will be four sources within publicly available datasets from NASA missions. The data although from NASA is provided by the Mikulski Archive for Space Telescopes (MAST) https://archive.stsci.edu/ The data sources I will be using are from the Kepler telescope mission and can be accessed as part of several data products MAST maintains such as API (Astro query) or downloaded in bulk with curl commands.

The retrieved data will be in the form of Target Pixel Files and will need to be converted into Light Curve files. (Figure 1 below)

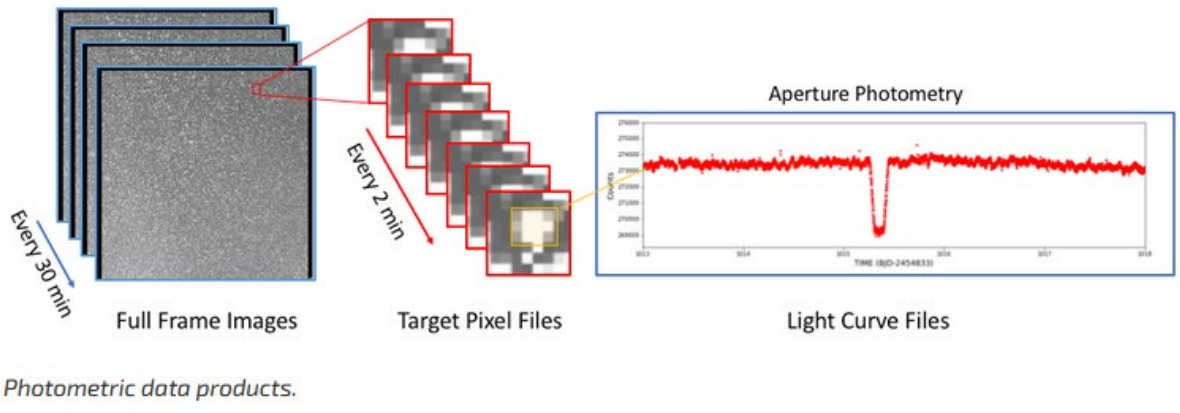


Figure 5 https://heasarc.gsfc.nasa.gov/docs/tess/images/tess_ffi_phot.png

The data will be split into two general sections. Training and Testing. Three main data sources will be used to create the labelled light curves, 1) List of 4,884 confirmed exoplanets with mission ID's, discovery method, and orbital (period) times. 3) List 9,564 objects of interest with Mission ID's, classification between confirmed and false positive, orbital times. 4) Electron flux data for each of the stars in the objects of interest table and converted to light curve data. The extracted light curves will need to be cleaned of possible instrumentation noise and other artifacts, normalised, and folded on their respective orbital period times then labelled for classification, so data can be split and used for training and testing with the machine learning models. Classification tests will be done multiple times per model chosen at different percentage distributions for training and testing. At this stage of the project these percentages have not been decided. The model testing will be the accuracy, recall, precision and F1 score of classifying light curves based on candidate or not.

From the light curve information extracted, I will be relying on the time and flux measurement attributes (The total flow of light measured over time, electrons per second) to identify possible candidates. The time and flux attributes will be my independent variables and my dependant variable will be the classification.

Cloud computing will then be used to process the full dataset and tested again. The technical specifications of the local machine as well as any cloud computing used to run the testing will be noted and compared but not used as a comparative measure for the Machine Learning process but for a runtime analysis.

Methodology & Analysis

The methodology I plan to use is KDD as I believe it suits the characteristics of my project the best. Breaking down the KDD methodology for my project (figure 2). The data selection is based on the problem domain and is the MAST Kepler Mission photometric data. Pre-processing will be the use of the mixture of the tables downloaded from MAST to create a secondary list to perform API calls with, retrieving the data. Transformation will be transforming the pre-processed flux data into a labelled dataset for classification. The data mining aspect will be the two Machine Learning models training and testing classification on the labelled datasets. The interpretation and evaluation step will be defined as investigating the appropriate machine learning method to analyse astronomical data for exoplanet candidates, with the hypotheses that Capsule Networks outperform CNNs in terms of metrics set out in the previous sections.

The data understanding for my project has its own challenges as I need to familiarise myself with the technical aspects of the data such as the flux and flux error attributes. As the data is semi structured this poses its own challenge of compiling the datasets into a labelled flux dataset. Another challenge is reading and understanding the technical documentation and gaining understanding from it to be able to apply, interpret and implement the machine learning methods effectively. The data preparation has been discussed in the previous section and will encompass the gathering of the data through the API and converting the pixel data to usable data.

The main analysis objective of my project will be on the modelling and model evaluation stages. These stages are where my project will gain the relevant insights and recommendations that will make up the final report.

During the stages of the project, different types of analysis will be performed, classification analysis will be performed on the dataset by each model chosen, while descriptive analysis will be used on the generated data from each model's performance metrics, testing for any statistical differences between them using both a 5×2 cv test (If possible, on cloud platform) and McNemar's test (for local testing).

5x2 Cross validation will be used using subsets of the main data to train and evaluate each model's performance on the chosen cloud platform as computational power will be more, this step will depend on the chosen cloud platform and the cost involved in running the models.

McNemar's test will be used for local testing as it has been demonstrated to be a useful test when computing power is limited and tests cannot be run multiple times due to time or computational cost restraints.

“For algorithms that can be executed only once, McNemar's test is the only test with acceptable type I error. For algorithms that can be executed 10 times, the 5 × 2 cv test is recommended, because it is slightly more powerful and because it directly measures variation due to the choice of training set.” (Dietterich, 1998)

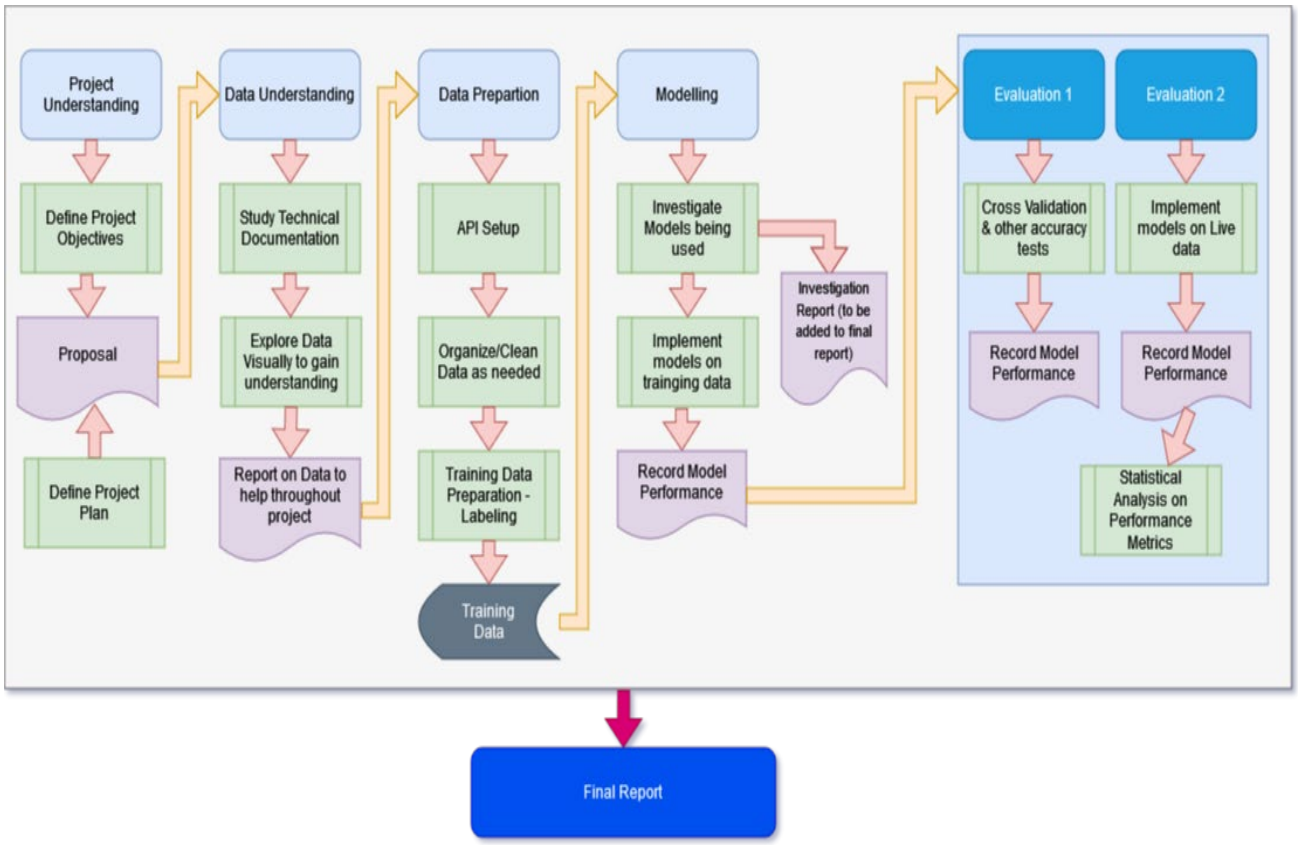


Figure 2: Preliminary Project Breakdown

Technical Details

For processing the data needed for my project, Python will be the language used. The IDE’s that will be used are Spyder for Python script building as well as Jupyter notebook for exploratory analysis on the datasets.

The machine learning methods will be implemented in Python and some visualization will be done in Tableau when exploring the data collected, both from the API as well as the data created by the performance metrics of each ML method.

Training data will be stored locally as csv files and the performance metrics will be exported to csv to be analysed.

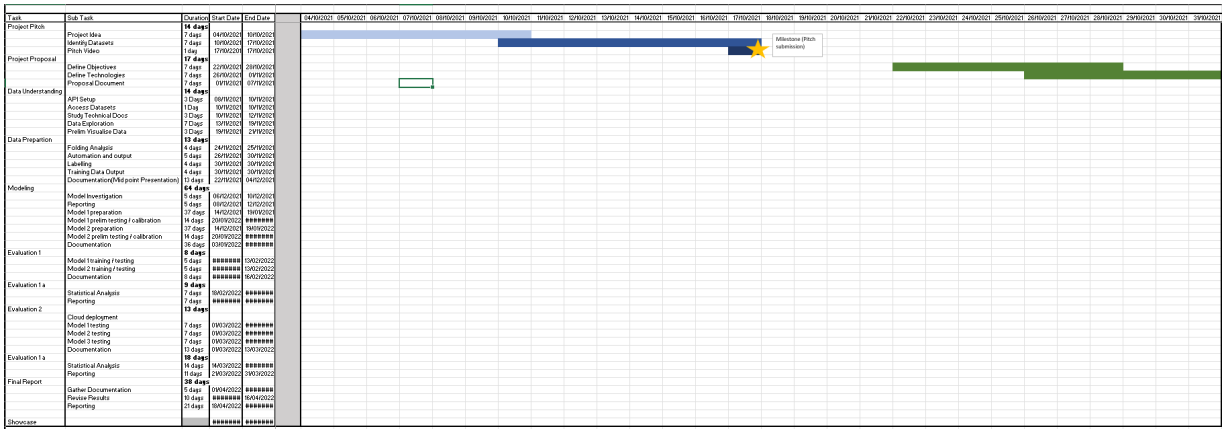
Libraries for the project below are the base libraries to be used in python.

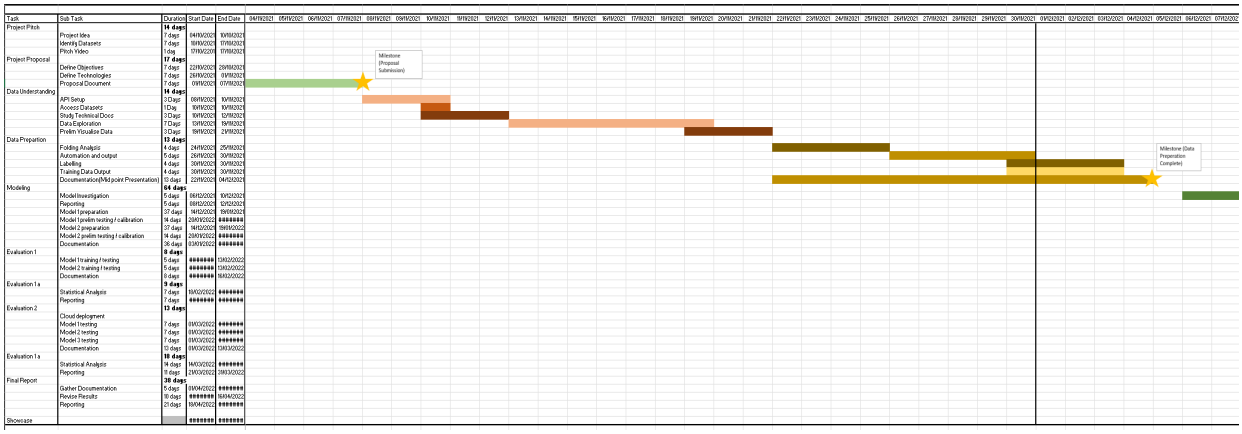
- Pandas – data manipulation
- Numpy - data manipulation & feature creation
- Matplotlib – visualization
- Lightkurve – Astroquery API and light curve analysis
- Keras & TensorFlow

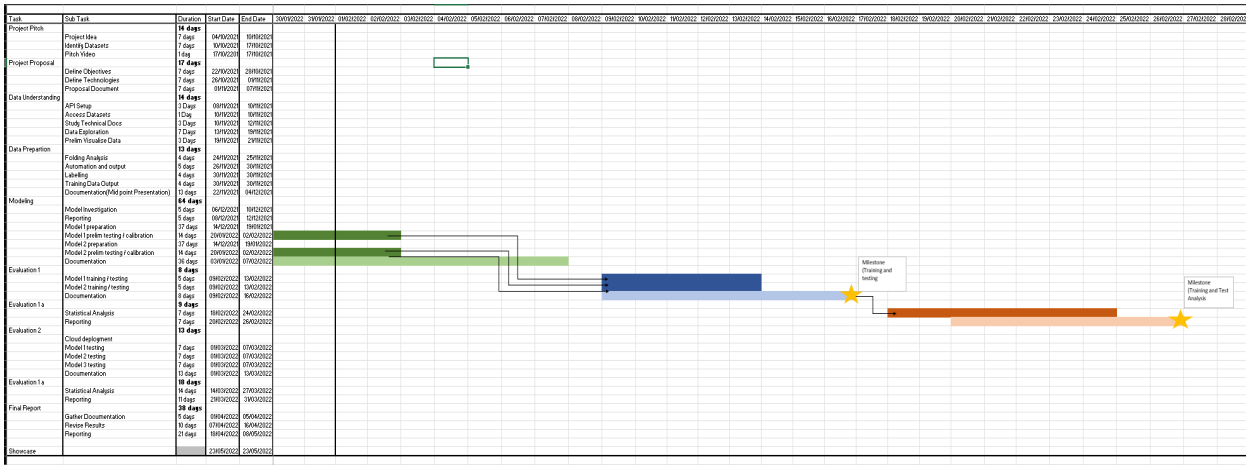
Project Plan

Some screen shots of a Gantt Chart created for the project plan. File embedded here:

Project_Gantt.xlsx







Proposal References

Dietterich, T. G. (1998). Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Comput*, 1895–1923.

H. P. Osborn, M. A. (2019). *Rapid classification of TESS planet candidates with convolutional neural networks.* Paris: EDP Sciences.

Jaiswal, A. W. (2018). Capsulegan: Generative adversarial capsule network. *European Conference on Computer Vision (ECCV)* (pp. 0-0). Munich: ECCV 2018 LNCS.

Jiang, P. C.-G. (2019). *Detecting Exoplanet Transits through Machine-learning Techniques with Convolutional Neural Networks.* Online: The Astronomical Society of the Pacific.

NASA. (2021, Nov 1). *Exoplanets*. Retrieved from nasa: https://exoplanets.nasa.gov

Vanderburg, C. J. (2018). *Identifying Exoplanets with Deep Learning: A Five-planet Resonant Chain around Kepler-80 and an Eighth Planet around Kepler-90.* Chicago: The American Astronomical Society.

9.2 Reflective Journals

Supervision & Reflection Journal

Student Name	Jonathan Flanagan
Student Number	18143890
Course	BSHCEDA4 – Computing (Evening) Data Analytics Stream

Month: October 2021

What?

First month back to college has been difficult to get back into the schedule of things and the fourth year workload being greater than any previous years (to be expected though). In terms of the project, we have our Project lecture Saturday mornings for one hour. So far it has been more of an introduction to what is expected and some guest lecturers on different aspects of the project cycle. Although parts have been relevant to the Data Analytics stream the main focus so far from guest lecturers has been more towards the software development side of things. I had been struggling with coming up with a project all summer, lacking focus in what direction to take. I've had a basic idea in what field I would like to try some analytics in but no real clear purpose when it comes to how to measure the project. After spending a few hours every few days brainstorming new ideas I concluded that I would just go with my first baseline idea and try build a project from that. After speaking with Enda after our first Project lecture it did give me hope that my project could have clear direction, I just needed to find it. I submitted my project idea knowing it was probably a bit too general of a scope, but still working hard on how to refine the idea into something more tangible if the general pitch got accepted. My project got accepted by my project supervisor Noel, but the second marker had questions and needed some more validation. Which I expected as I had the same concerns myself. Noel arranged a call with me, and we spoke for approx. 30mins, with his guidance he helped me narrow my scope to a more feasible and Analytics driven project by addressing the second markers queries one by one. I feel I needed that conversation with someone who understood the field and had gotten the gist of what I wanted to achieve, even though I couldn't verbalise it properly in my video pitch.

So What?

After my conversation with Noel, I felt like I had more focus and a clear view of what my project was going to entail at a high level. I think this was a great success in terms of being able to move forward with my project. I had already identified the data set I want to use but now I had the focus on what my hypotheses was and what I was going to be setting as the measurable aspect of my project.

A lot of challenges still remain, I will have to learn a lot of new scientific language and how to read the data set I will be using correctly, as well as investigating different machine learning techniques that we wouldn't come across in our college course.

Now What?
For the meantime my first and foremost challenge is to read as much literature as I can on the methods already used to explore the dataset I have chosen (as it has been analysed previously by a lot of people) and try to define myself apart from these methods. This will be a key aspect of my proposal due early next month. I also need to get myself acquainted with the newer machine learning methods I will be testing against such as Capsule Neural Networks. We are in the early stages of the project so my main focus at the minute is to read the current literature and set myself apart from in it in a novel way as part of my proposal.

Supervision & Reflection Journal

Student Name	Jonathan Flanagan
Student Number	18143890
Course	BSHCEDA4 – Computing (Evening) Data Analytics Stream

Month: November 2021

What?
This month has been a struggle time wise with the volume of ca's and project work. As far as project work is concerned, I'm behind on where I expected to be at this stage. Project proposal was submitted last month but no feedback has been received as of yet so I'm still following my initial plan and proposal ideas. I had planned to be at a stage where the literature review, API automation, and a prelim exploration has been completed with labelling and training data completed. Literature review is a work in progress still, API automation is almost complete, and some prelim exploration has been completed but no work on the machine learning applications have been started. We have covered some statistical analysis techniques in both Data application development and Business Analysis that will transfer nicely into the main software project. Some python learning has been completed outside of college to help with the project, Jupyter notebook and Spyder IDE are the main tools I'm using at the moment, I've practiced using these with Data Applications CA1 to get used them for use in the project.

So What?
Some items on the Gantt chart submitted will need to be reorganized with new importance to what should be in the midterm presentation. Literature review, Capsule Network research and a prelim analysis of the MAST data sets will be the main focus.

Now What?
No changes have been made to my proposal as of the end of this month so I will be following what I set out in the current proposal. Preparations for the midpoint presentation will be my main priority and reaching the milestones set out in original proposal. Following the rubric on the Moodle page and trying to meet everything set out in the rubric as a guide.
Once the preparations for the midterm project are completed machine learning packages in python will be researched and small iterations on the labelled dataset will be attempted.

Supervision & Reflection Template

Student Name	Jonathan Flanagan
Student Number	18143890
Course	BSHCEDA4 – Computing (Evening) Data Analytics Stream

Month: December 2021

What?

Milestones have all been met for the project, it has been a difficult month catching up on work missed as well as staying on top of CA deadlines. Practicing python and R samples from "Comparative Approaches to Using R and Python for Statistical Data Analysis" I've decided on Python as the language to use. This choice makes some aspects of the project I expected to be difficult much easier with libraries such as Lightkurve for processing the data I'm using.

API has been tested and works although processing takes 9 hours at a time, and errors occurring in the middle of the process have added some delays to the project, even with these delays though I've managed to meet everything set out in my Gantt Chart.

So What?

Being on target with the project gives me confidence that I will manage to get everything set I've set out, but I am still struggling with articulating the project work in the documentation, this will take practice, but I don't anticipate my report writing skills to increase dramatically over the course of the project.

Now What?

Moving onto the next stage of the project and starting to build the models I'll be testing, this stage seems like it is going to have the steepest learning curve, TensorFlow will be my library of choice for implementing the models and a brief read of the documentation seems like there is a lot to learn in a short period of time if I want to stay on top of the project work.

Before I start this though I need to concentrate on the midpoint submission and the presentation.

Supervision & Reflection Template

Student Name	Jonathan Flanagan
Student Number	18143890
Course	BSHCEDA4 – Computing (Evening) Data Analytics Stream

Month: January 2022

What?

With the midterm presentation finished and the Christmas break the first two weeks of January where hard to get back in the swing of the project, along with the start of the new semester and new modules to take on. Having had covid and at the end of December and the start of January it was difficult to get everything done while sick but somehow managed it and kept a decent enough grade but not as high as I would've wanted to achieve. In terms of the project, January has been spent installing and testing TensorFlow correctly on my machine and doing the tutorials on the TensorFlow website to get to grips with creating different neural networks. I've also spent time reading about capsule networks as there are very few working examples available online. Overall though I'm happy with the progress made in January with reading papers and setting up my machine correctly to implement my neural networks

So What?

My slow start to January has had an impact on the project timeline but not significant enough to worry or to change my Gantt chart. I feel more prepared now with developing the Convolutional Neural Network after completing some of the TensorFlow tutorials and reading the documents.

Now What?

Next steps for February will be to assess my cleaned dataset and see if any changes need to be made and to implement and train the Convolutional Neural Network while documenting the results. Further after that the CapsNet is going to be by far the hardest thing to and achieve as there are so few examples of working models (3 I could find at the time of writing this, one of which are the original paper on them) my plan is to have the CapsNet finished in March so I can deploy both the cloud at the beginning of April and spend the rest of April and May writing my final report.

Supervision & Reflection Template

Student Name	Jonathan Flanagan
Student Number	18143890
Course	BSHCEDA4 – Computing (Evening) Data Analytics Stream

Month: February 2022

What?

For February I managed to complete and train my Convolutional Network with accuracy at 98% predictions. I have recorded the accuracy, recall and F1 score details and charted the performance of the model over different epoch cycles, so far there seems to be no increase in performance after the 20 epoch threshold. I have tested it up to 100 epochs of training with performance metrics all flat lining around the 20 epoch mark. In the beginning I was achieving 30% accuracy only but with tweaking some of the settings in the network the accuracy increased dramatically. We have also received our CAs for Data Mining and Advanced BA so I can see these taking away from time I'd like to spend on my project, I'll have to carve some extra time out somewhere to combat this as I mistakenly didn't take these into account for my workload projections

So What?

The addition of the CAs to the workload will be difficult but I think manageable, for my project I'm on track and will concentrate the coming month on the CapsNet. I'm on track if not slightly ahead of schedule with documentation of the CNN complete. I was worried coming into Feb that I might end up behind in terms of timeline, but it has worked out well and I'm happy with the results so far. Time management though is still going to be a major focus for this month. One thing I've learned in the past four years, and especially this year is time management is a key tool not to be overlooked.

Now What?

CapsNets are going to be my main focus for the coming month as well as getting ready for cloud deployment. If I manage to get everything done at a similar pace as I have for Feb then I will attempt to setup a live API connection to the MAST database for the cloud to test also test on a live data feed, this may be a bit ambitious and is outside the scope of the project, but I have it as a consideration depending on what progress is made throughout March.

Supervision & Reflection Template

Student Name	Jonathan Flanagan
Student Number	18143890
Course	BSHCEDA4 – Computing (Evening) Data Analytics Stream

Month: March 2022

What?

In March, continuing on with the convolutional neural network, I wasn't entirely happy with the CNN even though the test results did show 98% from further testing the model was showing some signs of over fitting as the results on test data or generated data did not perform where I would have hoped. In attempt to improve the models generalised performance I redone some of the code to be able to automatically build and test several CNN's and output the best model. To limit the number of parameters I based the general architecture on the paper I have been using to compare my results to as it is one of the most comprehensive papers and the CNN in that paper is testing against NASA's own neural network for comparisons. (Vanderburg & Shallue, 2018) It took a few days of writing new code and testing and training and setting the hyperparameters, but the output model was slightly different to the papers model with only two differences. A smaller secondary kernel in the first layer and instead of 4 fully connected layers before the output there is only 1. This model did produce almost identical performance to the model in the paper with 95% accuracy on test data.

Secondary to this work at the start of the month but more importantly I began implementing the capsule network. This as expected is the hardest part of the project so far with a lot of documentation already ready and the maths studied to the best of my ability numerous versions of capsule networks have been attempted but do not work past the primary capsule network layer. The problem is trying to adjust the maths behind the capsule network to suit the data type that is being used in the project. Capsule networks so far in their implementation have been towards computer vision and show great promise. With that they use 4 dimensional inputs creating and 8 dimensional output which then has a 16 dimensional output for each pair between the 4 and 8D layers. The major issue being encountered is learning enough TensorFlow code to be able to reduce the dimensions to 1D at the initial layer, then 4D then 8D while keeping the same mathematical structure in the initial paper used to route throughout the nodes and produce an output. (Hinton, et al., 2017)

In conjunction to this, work in the CA was mounting and needed to be addressed along with starting the report for the final project. Notes have been taken and graphs produced with a draft of what I'd like to contain in the final report.

So What?

CA work has limited the time spent on the project as well as the changes made to the CNN at the beginning of the month slowed down the progress that I could make on the capsule network. After speaking with my project supervisor, he seems happy with the progress in the implementation but has suggested to start concentrating on the project report. He also suggested that a cloud deployment may not be necessary as it was initially suggested under the assumption that I would not have the computing power locally to be able to run and test the models, luckily, I do so cloud deployment isn't of great concern at the minute but if time permits, I may deploy it as a useable model.

Now What?

For the coming month getting the capsule network completed and running is the primary concern. The TABA's will be coming out also so they will eat into the time available. The project report needs to be moved from draft stages. I feel that I have the time to complete all objectives, but it will depend on how much time completing the capsule network takes and any tweaks or testing that need to be completed.

Supervision & Reflection Template

Student Name	Jonathan Flanagan
Student Number	18143890
Course	BSHCEDA4 – Computing (Evening) Data Analytics Stream

Month: April 2022

What?

April has been a busy month. TABAs were difficult and pretty much straight after CA submissions with tight deadlines which left little time for project work. Thankfully I was able to carve out enough time to keep on track with what I had planned to get done for the project this month. I successfully finished the capsule network, although the hypothesis being tested that this type of network would outperform the CNN as so far this type of network has shown great promise in computer vision tasks, it looks like in one or two dimensional data the feature recognition doesn't translate as well and the CNN outperforms the Capsule Network. The best I have achieved so far with the Capsule Network is 95% in training and 83% accuracy on test data. The number of tuneable parameters is quite high so takes a while to train and adjust. Their concept is fascinating though, and I do believe that with more time and better knowledge of the mathematics behind how they work the model could improve significantly. It was a challenge to overcome all the errors each step of the way through the model creation, but I learned a great deal about how these types of models work and the routing by agreement methods used. The report is starting to take shape and will be finished on time. The capsule network took longer to implement definitely has been the biggest challenge as there is no readily available documentation in TensorFlow about how to build one, it has really been trial and error and following the maths described in the paper along with learning TensorFlow code. (Hinton, et al., 2017) The best approach was blending the Keras API with direct TensorFlow code and has resulted in a highly tuneable model that works.

So What?

Now that all the code and models are complete for the project all results noted and all graphs created with about three quarters of the project report completed. I will spend the remaining time tidying up the report, making it as concise as possible without inflating and finishing of the smaller tasks in the project such as showcase page and picture and the tidying up the GitHub page for presentation.

Now What?

The last few final steps for the project are set in place. I have been slightly behind the planned timetable of what I set out, but I still have time to polish up some aspects and have the completed project ready on time. When everything is completed, the final step will be the presentation and the video to create. I've mapped out the final two weeks before the deadline to include report revisions, tidying up the GitHub page, creating the presentation and video.

9.3 Other materials used

MAST Website used for documentation relating to data retrieval:

https://archive.stsci.edu/

Kepler Objects of Interest Documentation:

https://exoplanetarchive.ipac.caltech.edu/docs/API_kepcandidate_columns.html#kic_param

Pandas Package Library Documentation:

https://pandas.pydata.org/docs/

TensorFlow Documentation:

https://www.tensorflow.org/overview

Keras Documentation:

https://keras.io/

Keras Hypermodel Documentation:

https://keras.io/api/keras_tuner/

Scikit-Learn Documentation:

https://scikit-learn.org/stable/supervised_learning.html#supervised-learning

LightKurve API documentation:

https://docs.lightkurve.org/reference/index.html

CUDA Nvidia Toolkit:

https://developer.nvidia.com/cuda-toolkit

cudNN:

https://developer.nvidia.com/cudnn