5 / 2021

SAPIENZA
UNIVERSITÀ DI ROMA

# Transit exoplanet detection via Machine Learning

Author : Senad Beadini

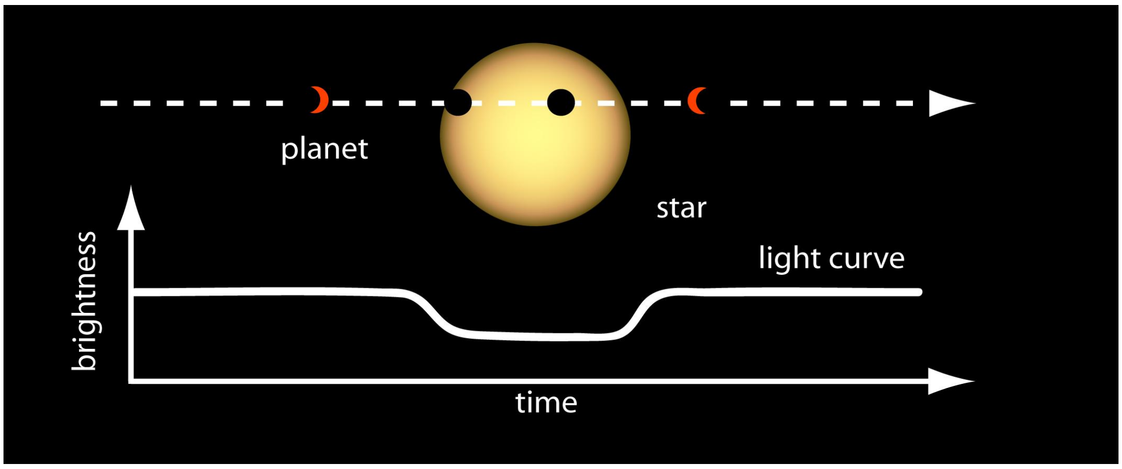
Master's Degree in Computer Science
- Project of Machine Learning exam

## Abstract

Scientists use data collected by space telescopes to find new information that allows us to learn more about the universe. The NASA Kepler space telescope has been collecting light from thousands of stars for many years to detect the presence of exoplanets.
An exoplanet is a planet that orbits a star, just like the Earth; however these systems are hundreds or thousands of light years away from Earth, so it is essential to have tools that can assist scientists in understanding whether a given star is likely to have exoplanets. The data collected by space telescopes is huge and new artificial intelligence techniques enable advanced data analysis and powerful predictive models.
In this project I used a dataset from the kaggle site, coming from the Mikulski Archive, a large archive of astronomical data for classifying the light curve of the stars to check the presence of the exoplanets. First of all, I'm going to apply different feature engineering techniques to the dataset and then I will present a neural network ( CNN ), which is the state of the art in Deep Learning for time series classification ( TSC ), and a SVC model which will be used for classifying the brightness flux of stars. I compare these 2 models in this activity and show how a Deep Learning approach in this problem achieves excellent performance compared to the standard Machine Learning (SVM) approach. Since the measure of brightness is a standard in these applications, these predictive models can be useful for future works with other and new larger dataset.

# 1 Introduction

The physical principle on which the research of exoplanets is based is simple. If there is a planet in orbit around a star, it may happen that the planet obscures the star, essentially it passes in front of its star in our direction. The Kepler space telescope equipped with high-precision instruments collects the brightness of thousands of stars for years to capture these variations in brightness. e
The K2 mission provides the community with an opportunity to continue Kepler's ground-breaking discoveries in the field of exoplanets and expand its role into new and exciting astrophysical observations [ 6 ]; observations are taken at one of two timing settings: long (30-min) or short (1-min) cadence. K2 observations entail a series of sequential observing "Campaigns" of fields distributed around the ecliptic plane. Each ecliptic Campaign is limited by Sun angle constraints to a duration of approximately 80 days [ 6 ].
We can see this image which represents the physical phenomenon [ 1 ] :



Once the data is gathered by the spatial telescope is processed by research departments and the light curve is extracted. The latter is a function of the time, as we can see in the image, and only in this moment we can use predictive models for classification.
In this project, I' going to use  a kaggle dataset that contains the brightness flux of over 5000 stars from campaign 3 of the Kepler mission [ 3 ] to classify the stars' signals into 2 categories using the state of the art in Deep Learning Time Series Classification ( TSC ).
First of all, I'm going to apply different feature engineering techniques to the stars' signal. In particular,  I'll apply a gaussian filter to the signal to try smoothing and reduce noise; In the end I'll apply the Fast Fourier Transform ( FFT ) to pass in the domain of the frequencies of the signals; this operation may allow a better interpretation of the data by the deep learning models. The performance of the models will be compared to a test set to find the best one.

## 2 Dataset

The dataset used in this project is a Kaggle's public dataset [ 3 ]. This dataset comes from the Mikulski Archive for Space Telescopes ( MAST ), a public database/archive where it is possible to find several data related to Astrophysics.
Now I'm going to show the properties of the dataset.
The data is divided into 2 csv files. One for training and another for testing.

| | LABEL | FLUX.1 | FLUX.2 | FLUX.3 | ... | FLUX.3194 | FLUX.3195 | FLUX.3196 | FLUX.3197 |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 2 | 93.85 | 83.81 | 20.10 | ... | 39.32 | 61.42 | 5.08 | -39.54 |
| **1** | 2 | -38.88 | -33.83 | -58.54 | ... | -11.70 | 6.46 | 16.00 | 19.93 |
| **2** | 2 | 532.64 | 535.92 | 513.73 | ... | -11.80 | -28.91 | -70.02 | -96.67 |
| **3** | 2 | 326.52 | 347.39 | 302.35 | ... | -8.77 | -17.31 | -17.35 | 13.98 |
| **4** | 2 | -1107.21 | -1112.59 | -1118.95 | ... | -399.71 | -384.65 | -411.79 | -510.54 |

Above we can see the first 5 lines of the training dataset. In particular we can notice that every row represents a star and contains a time series data about its brightness. Thus each column represents the brightness of a star at a precise moment. For that reason the dataset is huge in width.

Train dataset:

- 5087 rows or observations.
- 3198 columns or features.
- Column 1 is the label vector. Columns 2 - 3198 are the flux values over time.
- 37 confirmed exoplanet-stars and 5050 non-exoplanet-stars.

Test dataset :

- 570 rows or observations.
- 3198 columns or features.
- Column 1 is the label vector. Columns 2 - 3198 are the flux values over time.
- 5 confirmed exoplanet-stars and 565 non-exoplanet-stars.

Because a “Campaign” lasts approximately 80 days [ 6 ] we can infer that :

Observation in our dataset are per every star 3197 then

$$80 \text{ days} = 6912000 \text{ seconds}$$

$$\Downarrow$$

$$(6912000) / (3197 \text{ columns}) \simeq 2162 \text{ seconds}$$

$$\Downarrow$$

$$2162 \text{ seconds} \simeq 36 \text{ minutes}$$

There is an observation every ~36 minute.

Thus, we can conclude that :

$$\text{frequency} \simeq \frac{1}{2162} \simeq 0{,}00046 \text{ Hz}$$

## 3 Data Preprocessing

Luckily the dataset is clean, for instance there are no null values, there are no formatting problems, there are no missing values, there are no complex formats to manage; all values are real numbers. But for a better interpretation of the data I need to do feature engineering.

The column label "LABEL" contains 2 or 1 values. 2 indicated that the star is confirmed to have at least one exoplanet in orbit. A better thing is to have value 0 and 1 instead of 1 and 2. To see the data better I'll show the light curve of some stars using Python MatplotLib. Since, in the field of Digital Signal Processing it is common to apply to signals (continuous or discrete) filters that allow to clean, remove the noise or amplify the signals I'll apply to the data a gaussian filter.
The gaussian filter is a very useful tool, which smooths the signal source and decreases the noise. It is reasonable to think that in this application that filter can somehow make the light curve cleaner and less irregular. However, this filter could remove important information from the signal so it is essential to design the filter well.
Formally the gaussian filter 1D can be expressed as follow :

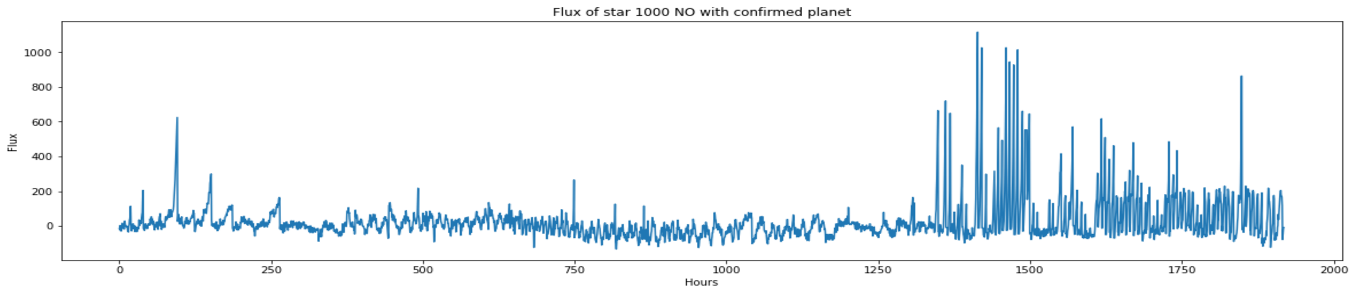
$$G(x\,,\,\sigma) \;=\; \frac{1}{\sqrt{2\pi}\,\sigma}\,e^{-\frac{x}{2\sigma^2}}$$

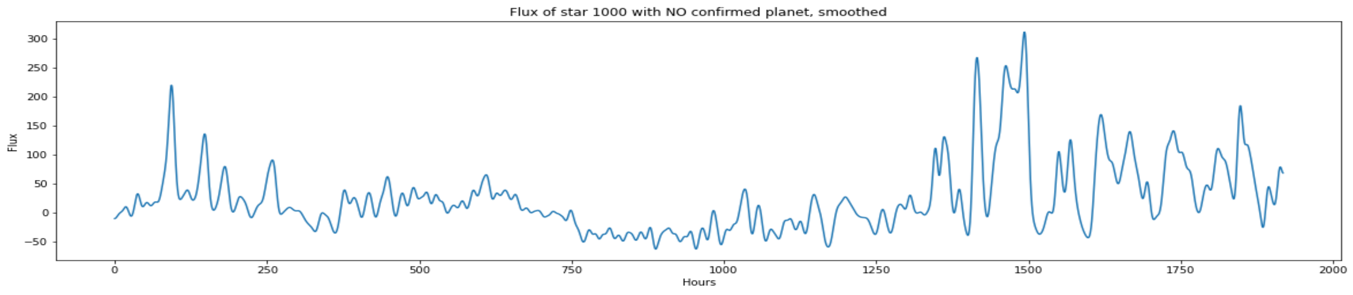
where σ is a parameter.
The choice of σ could be difficult, in terms of machine learning σ is a hyperparameter of the gaussian filter. I chose the value σ = 5 that allows a soft "smoothing effect" and it appears not to delete information from the data.
Without loss of generality, from now the data will be considered normalized ( with the function "normalized" of the Scikit Learn library, which implements a L1 normalization ).
Another thing to keep in mind is that in Digital Signal Processing it is common to switch to the frequency domain of a signal to see the components of the signal. In our application this can be done because it is reasonable to assume that a transiting planet changes the frequency of the stellar flow; another advantage is that we pass from a time-dependent signal to a time-independent series of numbers. This can be done using the Fourier transform.
Formally, the Fourier transform $F$ of a function $f$ is defined as follows [ 4 ] :

$$F(\,s\,) \;=\; \int_{-\infty}^{+\infty} f(x)\; e^{-2\pi\, i\, x\, s}\, dx$$

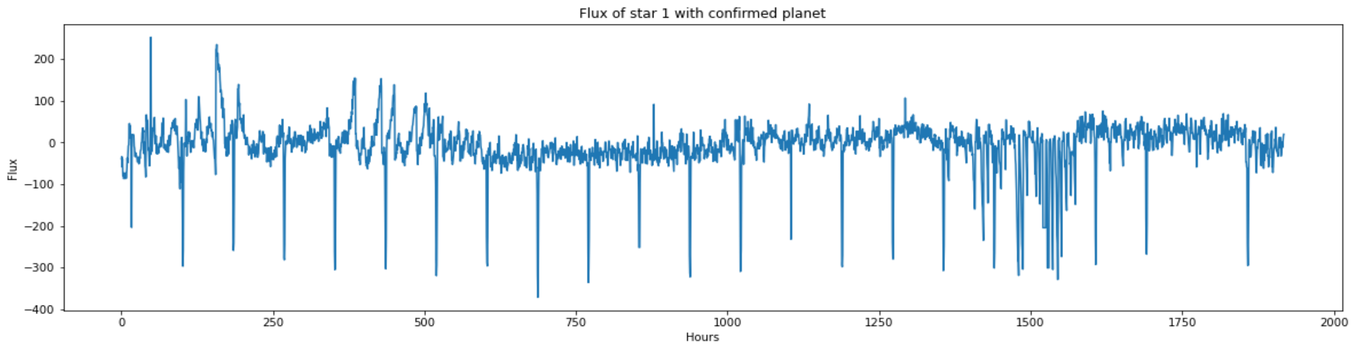
This operation allows us to obtain a new function in the domain of frequencies.

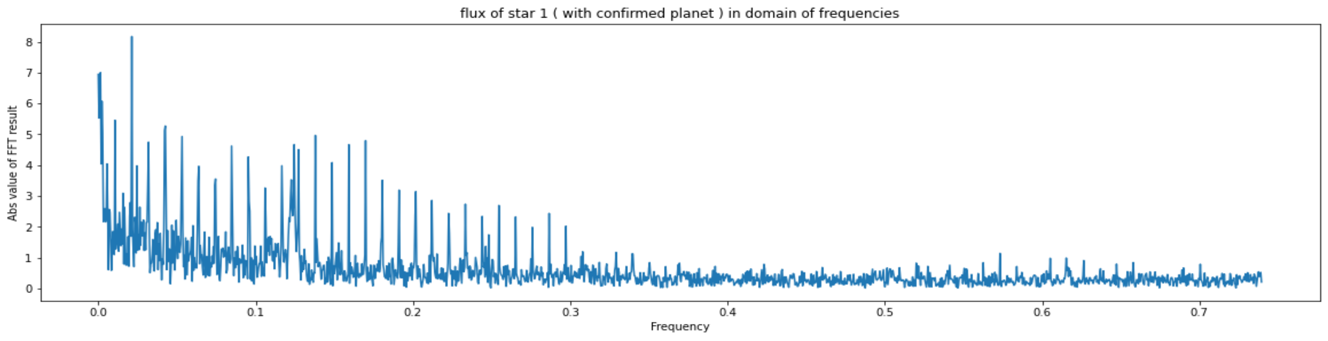
**Light curve of the Star 1000 with NO confirmed planet, first original then smoothed ( No normalized ) :**





**Light curve of the Star 1 with confirmed planet, first original then smoothed (No norm. ). In the end, amplitude of the FFT data ( this time the signal was normalized with L1 ) :**

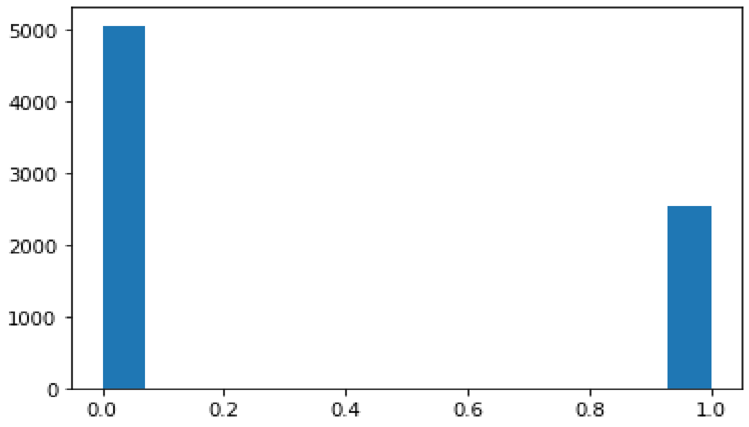




What else can I do to make the data more robust?
We can see that in the training set there are :

- **37** confirmed exoplanet-stars and **5050** non-exoplanet-stars.

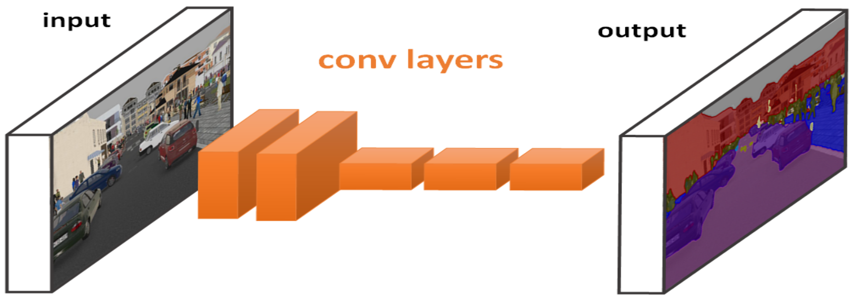
Thus, we can use an oversampling method to make the dataset more balanced.
I'm going to use a simple random oversampling technique of 50% to grow up the number of data representing the minority class.
After applying oversampling to the dataset we obtained a new dataset with the following distribution of the classes :

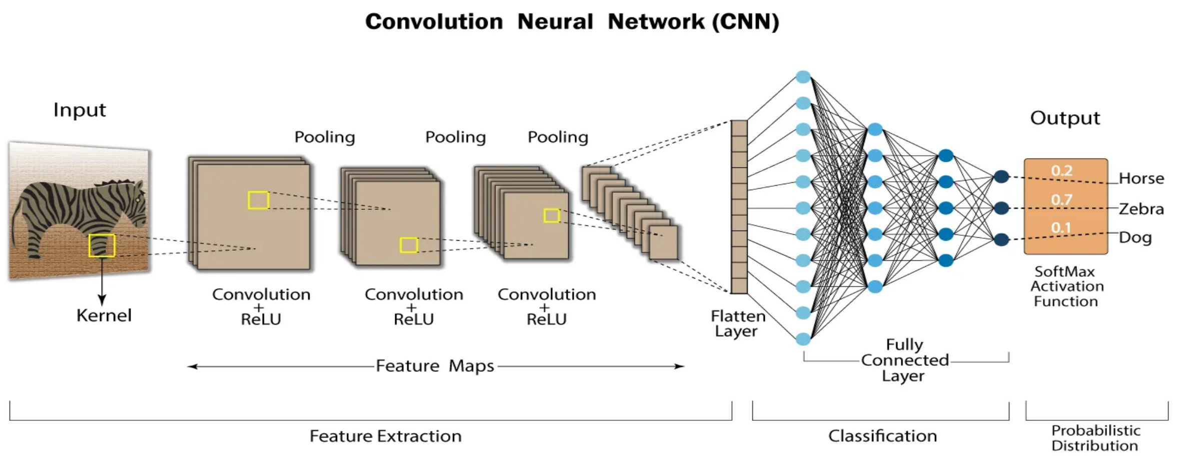


Now the dataset is more balanced.

# 4 Machine Learning Models

Fawaz, Hassan Ismail, et al. in [ 8 ] show the state of the art of Deep Learning in the classification of time series data ( TSC ). In particular, according to [ 8 ] the state of the art in deep learning in the field of time series data classification seems to be ResNeT ( Residual Neural Network ) and FCN/CNN ( Fully Convolutional Neural Network ).
In this project I'm going to use a CNN inspired by the model presented in [ 7 ] and a SVM model. Let's go deeper into these 2 models.
FCN showed great performance in semantic segmentation on image [ 9 ]. FCNs essentially are convolutional networks ( CNN ) with a different architecture in the last layers.
The following two examples are FCN and CNN models ( in Computer Vision applications ).





A second approach chosen by me, in order to classify the times series data, was a SVC (Support Vector Classifier). The choice was based on [ 12 ] and [ 13 ]. SVC is a machine learning model [ 10 ] which can achieve excellent performance in different tasks. Compared to deep neural networks, they have fewer parameters and can work ( like Neural  Networks ) even with large dimension data.

## SVC Model

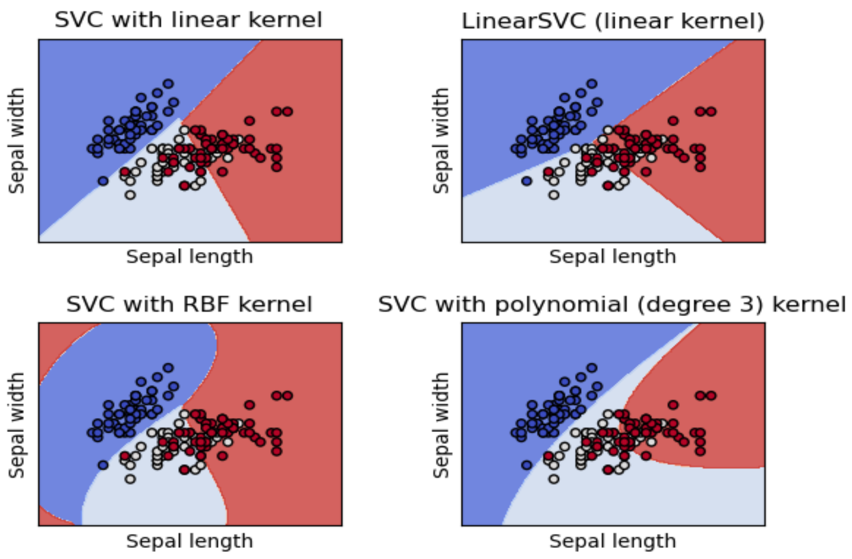
The main advantages of SVC are:

- operates in high-dimensional space in efficient way
- versatility : different kernel functions are enabled

A SVC, in a two-class classification problem, describes data as k-dimensional vector x with corresponding labels y in { -1, 1 }. Then SVC separates data with a hyperplane in order to maximize the distance from the nearest point to the plane.
Since I use the data in frequency domain, the classifier takes as input a vector corresponding to the amplitude of a FFT transformed signal, then it maps the input vector in a high-dimensional feature space using a Radial Basis Function “RBF” :

$$K(x, y) = e^{-\gamma|x-y|^2}$$

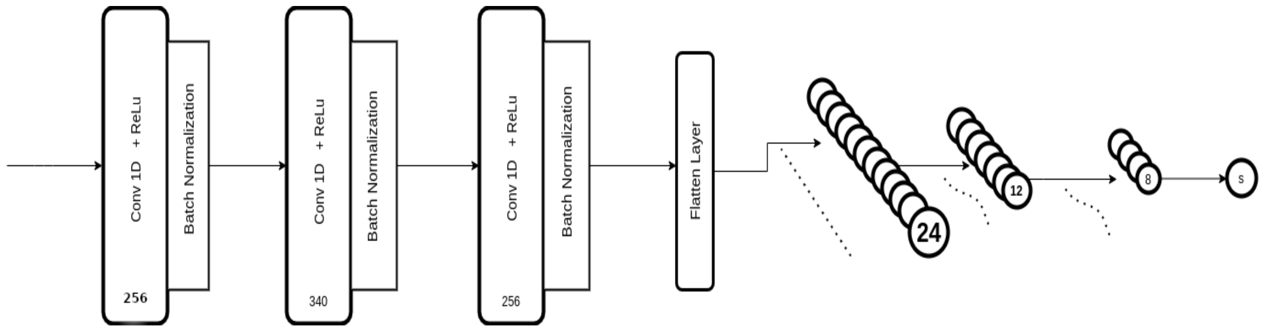
in the end the classifier outputs a class prediction.
The kernel has two parameters C and gamma. A high C aims at classifying all training examples correctly; a lower C will encourage a larger margin. Gamma defines how much influence a single training example has; the larger gamma is, the closer other examples must be to be affected. So once I have chosen the parameters the SVC model is ready to perform.



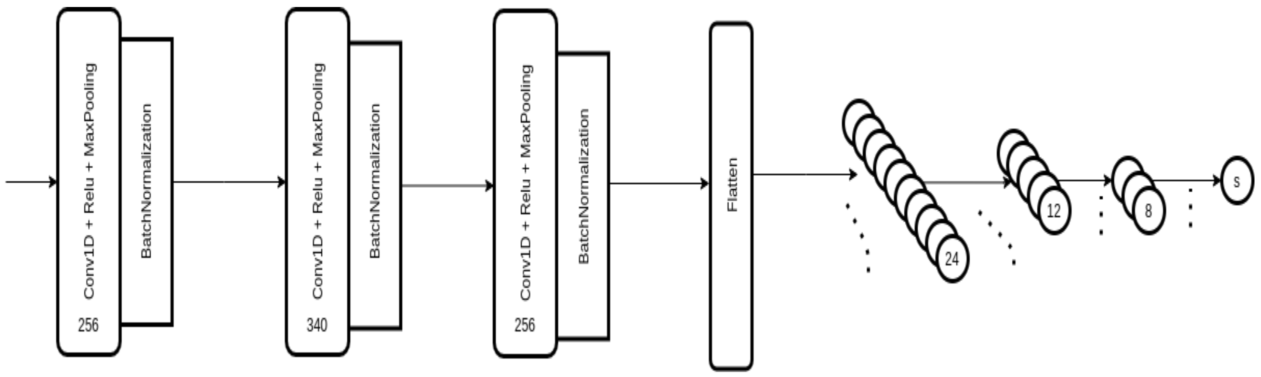
An example of SVC with different kernels function [ 11 ].

## CNN Model

Let's take a look in detail at the architecture of the network that I'm going to use.
As I said before, the model chosen by us is CNN. The specific architecture I have chosen for the CNN, based ( not equals ) on paper of Wang, Zhiguang, Weizhong Yan, and Tim Oates [ 7 ], is the following;



However, I have decided to add 3 max pooling layers after each conv1d layer to the above schema to decrease the number of parameters. So the final architecture is the following :


Conv1D + Relu + MaxPooling
256
BatchNormalization
Conv1D + Relu + MaxPooling
340
BatchNormalization
Conv1D + Relu + MaxPooling
256
BatchNormalization
Flatten
24
12
8
s

Obviously since the problem is a binary classification problem the best choice is the binary cross entropy as loss function. I underline that I have added the dropout technique in 2 layers. Here below, in the next page, we can see all the details of the network.

```
Model :
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv1d (Conv1D)              (None, 3197, 256)         2304      
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 640, 256)          0         
_________________________________________________________________
batch_normalization (BatchNo (None, 640, 256)          1024      
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 640, 340)          522580    
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 128, 340)          0         
_________________________________________________________________
batch_normalization_1 (Batch (None, 128, 340)          1360      
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 128, 256)          348416    
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 26, 256)           0         
_________________________________________________________________
batch_normalization_2 (Batch (None, 26, 256)           1024      
_________________________________________________________________
flatten (Flatten)            (None, 6656)              0         
_________________________________________________________________
dropout (Dropout)            (None, 6656)              0         
_________________________________________________________________
dense (Dense)                (None, 24)                159768    
_________________________________________________________________
dropout_1 (Dropout)          (None, 24)                0         
_________________________________________________________________
dense_1 (Dense)              (None, 12)                300       
_________________________________________________________________
dense_2 (Dense)              (None, 8)                 104       
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 9         
=================================================================
Total params: 1,036,889
Trainable params: 1,035,185
Non-trainable params: 1,704
```

The number of parameters deriving from the use of the max pool layers decreases by a factor of 20. In fact we pass from about 20 million parameters to about 1 million. Thanks to that change, the computational complexity of the model has decreased and obviously we have that the training lasts less.

## 5 Code and Implementation

The code was written in 2 files. The first file makes all data processing operations include training and evaluation while the second takes care about building the models.
Since the models used are very complex ( the CNN has more than one million of parameters ) I've used an Google Colab GPU environment to run the code.
The code is available in the following github repository;

URL : https://github.com/senad96/exoplanet-detection-via-DeepLearning_v1

## 6 Results and Performance

The following parameters for the CNN ( designed in Keras ) are be chosen:

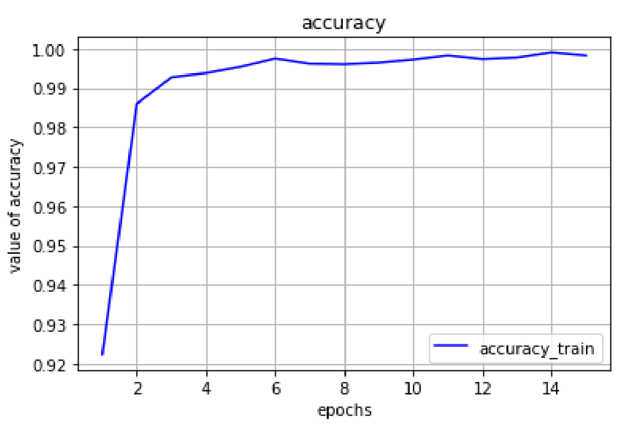
- optimizer = Adam
- learning rate = 0.001
- padding ( for each conv1d layer ) = same
- kernel size ( for each conv1d layer respectively ) = 8, 6, 4
- dropout = 0.3
- epochs = 15 or more
- batch_size = 10

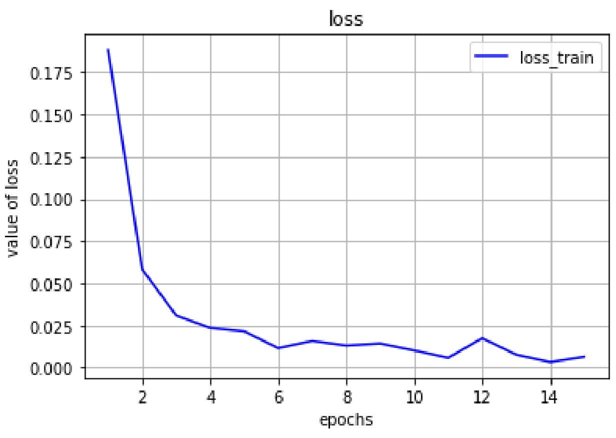
In SVC model I setted the following hyper-parameters :

- C = 100
- gamma = 0.001
- kernel = rbf

The choice of C and gamma is critical for SVM's performance. For this reason, the parameters presented above have been searched using cross-validation ( "gridsearchCV"). In this research, given a set of candidates parameter to try, and a score function, the process generates several candidate models to train using training data. Based on the score function, the best set of parameters is chosen.

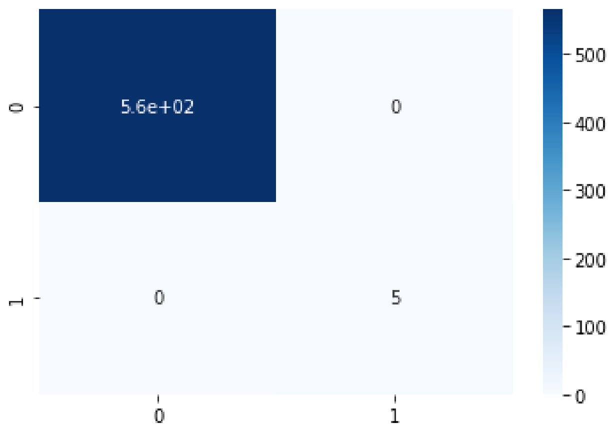
1) We can see the performance of the network during training :





The Accuracy tends to 1 and the loss function tends to 0. This agrees with our theoretical hypotheses. Let's see the predictions on the test set.

**Confusion matrix :**



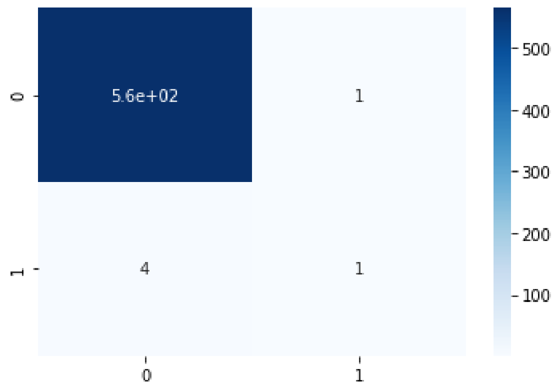
**Other metrics :**

```
                            precision    recall  f1-score   support

  NO exoplanet confirmed       1.00      1.00      1.00       565
 YES exoplanet confirmed       1.00      1.00      1.00         5

                accuracy                           1.00       570
               macro avg       1.00      1.00      1.00       570
            weighted avg       1.00      1.00      1.00       570
```

These data tell us that the neural network can predict 5 out of 5 exoplanets in the test set and correctly predict the others negative examples. So the network reaches an 100% accuracy.
An important thing to say is that the optimizer Adam has achieved better results than the Stochastic Gradient Descent optimizer algorithm.
In fact with the SGD optimizer method in keras I got not more than 4 predictions out of 5 ( 99.12 % accuracy ) with some false positive examples whereas with Adam, as I showed above, I obtained the maximum accuracy on the test set.
Another important thing to note is that in this project I have not used a validation set during the training of the CNN. Why did I make this choice ? basically due to the unbalanced dataset. In the training set I had only 37 positive examples and the other positive examples were given by the oversampling technique; for that reason each creation of a validation set ( having different dimensions ) for the neural network caused a performance degradation. We've based the hyper-parameters tuning on [ 7 ], [ 8 ], in the speed of convergence and in the loss function value during the training.
After all, the results of the network are incredible. Even if we had not 100% accurate results ( as in the case with SGD ) predictive models of this type would be very useful to scientists and astrophysicists to focus only on some stars and discard others.

2 ) With the SVC model we obtain  the following results :

**Confusion matrix :**



**Other metrics :**

```
                           precision    recall  f1-score   support

  NO exoplanet confirmed        0.99      1.00      1.00       565
 YES exoplanet confirmed        0.50      0.20      0.29         5

                accuracy                            0.99       570
               macro avg        0.75      0.60      0.64       570
            weighted avg        0.99      0.99      0.99       570
```

We can see that the SVC model finds only an exoplanet; it failed to recognize the other 4 positive examples. Moreover, we can see that the deep neural network approach in this application has better results than SVM approach. I've tried also to avoid the use of the gridsearchCV for example setting "reasonably" the value of C and changing the kernel but I never got better performance.

## 7 Future Work

In the field of applied astrophysics the transit detection of exoplanets is an important technique for the discovery of new planets [ 2 ].
It would be very interesting to test these models ( in particular CNN ) presented in this project in new even larger datasets. Recall that in the MAST archive ( Mikulski Archive for Space Telescopes ) it is possible to obtain the light curves of thousands of other stars collected by the kepler telescope [ 5 ]. However these data are not in a form directly usable by a machine learning model and therefore would require to be downloaded and processed to extract the light curves.
In this project due to a limited time it was not possible to test the models in new datasets. In case someone would like to do this the code has been constructed to be able to change a number of reduced variables to adapt the models to new datasets. In particular the things to change in the code would be:

- Put the new datasets in the variable "data_train", "data_test"
- Check the target label in the training set and test set is the first column
- The CNN input number. In our case the network accepts 3197 long sequences.
  In other cases it could be less or more.
  NOTE : In the scenario where the signals are of different lengths, it is possible to make them all of the same length by choosing a fixed number of values after the application of FFT ( for example frequencies with greater amplitude).
- Eventually change the hyper-parameters ( sigma of gaussian filter, layers of CNN etc. )

Note that it could be possible to run a greater number of epochs ( CNN model ) than 15 to reach my accuracy.

# References

[ 1 ] NASA Content Administrator, *Aug. 7, 2017*
https://www.nasa.gov/mission_pages/kepler/multimedia/images/transit-light-curve.html
date: 19/12/2020

[ 2 ] Dattilo, Anne, et al. "Identifying Exoplanets with Deep Learning. II. Two New Super-Earths Uncovered by a Neural Network in K2 Data." *The Astronomical Journal* 157.5 (2019): 169.

[ 3 ] https://www.kaggle.com/keplersmachines/kepler-labelled-time-series-data
date : 11/12/2020

[ 4 ] https://en.wikipedia.org/wiki/Fourier_transform
date : 9/12/2020

[ 5 ] https://archive.stsci.edu/missions-and-data/k2
date : 17/12/2020

[ 6 ] https://www.nasa.gov/mission_pages/kepler/overview/index.html
date : 17/12/2020

[ 7 ] Wang, Zhiguang, Weizhong Yan, and Tim Oates. "Time series classification from scratch with deep neural networks: A strong baseline." *2017 International joint conference on neural networks (IJCNN)*. IEEE, 2017.

[ 8 ] Fawaz, Hassan Ismail, et al. "Deep learning for time series classification: a review." *Data Mining and Knowledge Discovery* 33.4 (2019): 917-963.

[ 9 ] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[ 10 ] Hearst, Marti A., et al. "Support vector machines." *IEEE Intelligent Systems and their applications* 13.4 (1998): 18-28.

[ 11 ] https://scikit-learn.org/stable/modules/svm.html,
date : 19/12/2020

[ 12 ] Ramon, Manel Martinez, et al. "Signal classification with an SVM-FFT approach for feature extraction in cognitive radio." *2009 SBMO/IEEE MTT-S International Microwave and Optoelectronics Conference (IMOC)*. IEEE, 2009.

[ 13 ] Wang, Zhimeng, et al. "Fast Fourier Transform-based Support Vector Machine for Subcellular Localization Prediction Using Different Substitution Models." *Acta biochimica et biophysica Sinica* 39.9 (2007): 715-721.