MACHINE LEARNING TECHNIQUES LAB – CS2806

DETECTION OF EXOPLANETS WITH DEEP LEARNING-CNN

S.Rishabh

—

22011101095

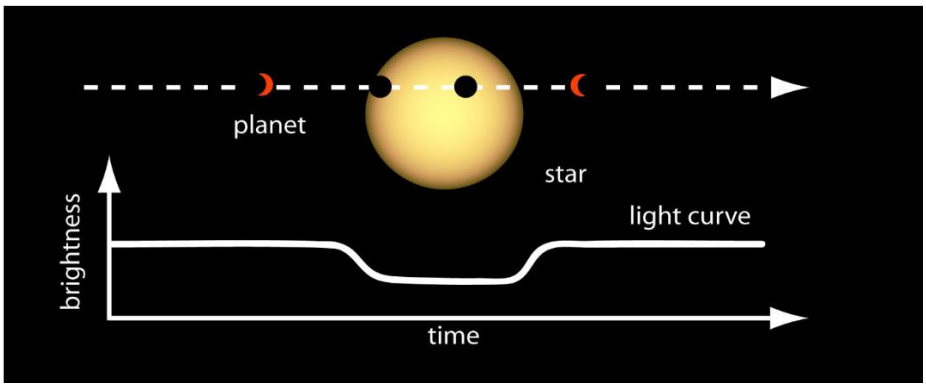
—

SNU Chennai

ABSTRACT:

The exploration of exoplanets relies on a fundamental physical principle. When a planet orbits a star, there's a possibility that it may pass between the star and our observation point, causing an observable dimming of the star's brightness. Utilizing advanced technology like the Kepler space telescope, equipped with high-precision instruments, scientists gather data by monitoring the brightness variations of thousands of stars over extended periods.

Upon collection, this data undergoes meticulous processing by research departments, extracting what is known as a light curve—a graph illustrating brightness fluctuations over time. At this stage, predictive models become applicable for classification purposes.



In this project, I aim to leverage a Kaggle dataset comprising the brightness flux data of over 5000 stars observed during campaign 3 of the Kepler mission. My objective is to classify these stars' signals into two distinct categories employing cutting-edge Deep Learning Time Series Classification techniques.

To achieve this, I'll employ two distinct feature engineering methodologies. Firstly, I'll utilize a Convolutional Neural Network (CNN), acknowledged as the forefront approach in Deep Learning for time series classification (TSC). Secondly, I'll implement a Support Vector Classification (SVC) model to classify the brightness flux of stars.

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

Through this endeavor, I intend to conduct a comparative analysis between these two models, showcasing how a Deep Learning approach significantly outperforms the traditional Machine Learning (SVM) approach in addressing this particular problem.

DATASET DESCRIPTION:

The dataset utilized in this project originates from a publicly available resource on Kaggle. Specifically, it is sourced from the Mikulski Archive for Space Telescopes (MAST), a repository housing a plethora of data pertinent to Astrophysics.

This dataset is segmented into two distinct CSV files: one designated for training purposes and the other for testing. A glimpse at the initial five lines of the training dataset reveals that each row corresponds to a unique star and comprises a time series dataset detailing its brightness fluctuations. Consequently, each column encapsulates the brightness data of a star at a specific time instance. This characteristic contributes to the dataset's expansive width

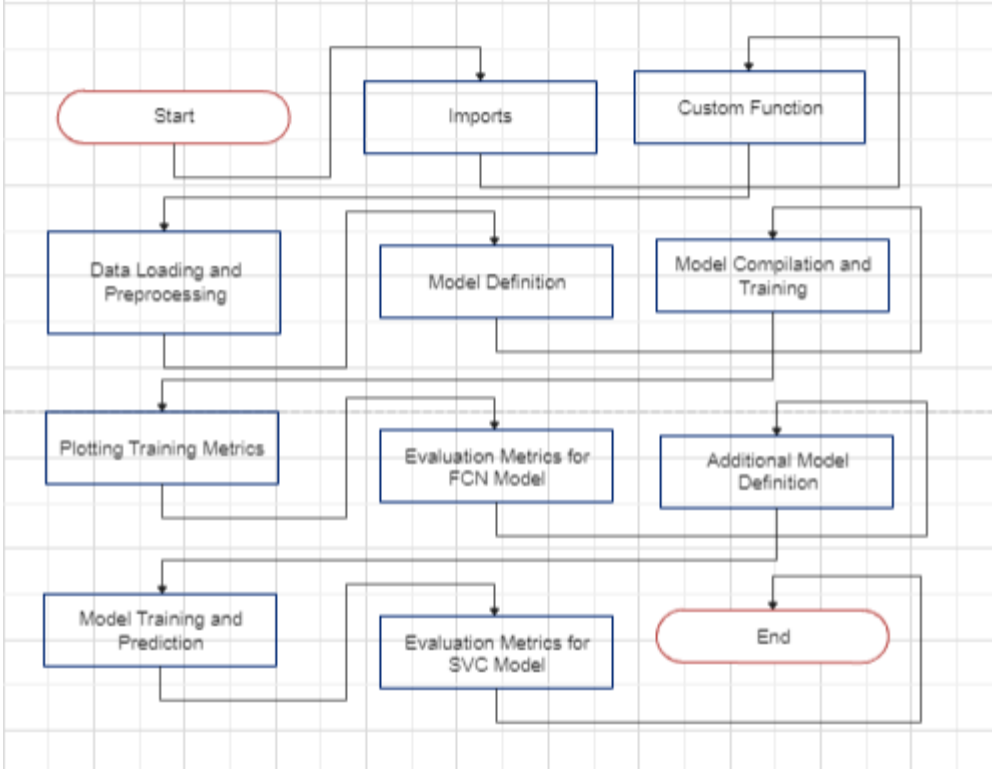
	LABEL	FLUX.1	FLUX.2	FLUX.3	...	FLUX.3194	FLUX.3195	FLUX.3196	FLUX.3197
0	2	93.85	83.81	20.10	...	39.32	61.42	5.08	-39.54
1	2	-38.88	-33.83	-58.54	...	-11.70	6.46	16.00	19.93
2	2	532.64	535.92	513.73	...	-11.80	-28.91	-70.02	-96.67
3	2	326.52	347.39	302.35	...	-8.77	-17.31	-17.35	13.98
4	2	-1107.21	-1112.59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

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



CODE:

DEFINING MODELS:

```
import numpy as np
import tensorflow as tf
import pydot

from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, classification_report
from keras.utils.vis_utils import plot_model

from tensorflow.keras import models
from tensorflow.keras import layers
from tensorflow.keras.preprocessing import sequence


from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC


#create the neural network
def FCN_model(len_seq):
```

```
    # len_seq = the size of the input sequences

    model = tf.keras.Sequential()

    #change the input shape if you have sequences less long
    model.add(layers.Conv1D(filters=256, kernel_size=8, activation='relu',
input_shape=(len_seq,1)))
    model.add(layers.MaxPool1D(strides=5))
    model.add(layers.BatchNormalization())


    model.add(layers.Conv1D(filters=340, kernel_size=6, activation='relu'))
    model.add(layers.MaxPool1D(strides=5))
    model.add(layers.BatchNormalization())


    model.add(layers.Conv1D(filters=256, kernel_size=4, activation='relu'))
    model.add(layers.MaxPool1D(strides=5))
    model.add(layers.BatchNormalization())


    model.add(layers.Flatten())
    model.add(layers.Dropout(0.3))


    model.add(layers.Dense(24, activation='relu'))
    model.add(layers.Dropout(0.3))

    model.add(layers.Dense(12, activation='relu'))


    model.add(layers.Dense(8, activation = 'relu'))


    model.add(layers.Dense(1, activation='sigmoid'))

    return model



#create the SVC model
def SVC_model():

    tuned_parameters = [{'kernel': ['rbf'], 'gamma': [1e-3, 1e-4],
```

```
                     'C': [1, 10, 100, 1000]},
                    {'kernel': ['linear'], 'C': [1, 10, 100, 1000]}]


    clf = GridSearchCV( SVC(), param_grid = tuned_parameters,scoring = 'recall')


    return clf
```

TRANING/TESTING/COMPARISION AND EXECUATION:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import signal
from scipy.ndimage.filters import gaussian_filter
from scipy.fftpack import fft
import scipy
import seaborn as sns
import models as m

import sklearn.linear_model as lm
import tensorflow as tf
import sklearn.svm as svm

from tensorflow.keras import models
from tensorflow.keras import layers
from tensorflow.keras.preprocessing import sequence

import sklearn.preprocessing as pproc
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import plot_confusion_matrix
from sklearn.preprocessing import normalize

def custom_random_oversampling(X, y, sampling_strategy=0.5, random_state=None):
    np.random.seed(random_state)
    unique_classes, class_counts = np.unique(y, return_counts=True)
    minority_class = unique_classes[np.argmin(class_counts)]
    majority_class = unique_classes[np.argmax(class_counts)]
    minority_indices = np.where(y == minority_class)[0]
    majority_indices = np.where(y == majority_class)[0]
```

```
    # Calculate the number of samples to select from the minority class
    n_samples_minority = int(len(majority_indices) * sampling_strategy)
    selected_minority_indices = np.random.choice(minority_indices,
size=n_samples_minority, replace=True)

    # Combine minority and majority samples
    X_resampled = np.concatenate((X[selected_minority_indices], X[majority_indices]),
axis=0)
    y_resampled = np.concatenate((y[selected_minority_indices], y[majority_indices]),
axis=0)

    # Shuffle the combined samples
    shuffle_indices = np.arange(len(X_resampled))
    np.random.shuffle(shuffle_indices)
    X_resampled = X_resampled[shuffle_indices]
    y_resampled = y_resampled[shuffle_indices]

    return X_resampled, y_resampled




# import the dataset
data_train = pd.read_csv(r'C:\Users\Rishabh\Desktop\MLT LAB\exoTrain.csv')
data_test = pd.read_csv(r'C:\Users\Rishabh\Desktop\MLT LAB\exoTest.csv')



#permute the dataset
data_train = np.random.permutation(np.asarray(data_train))
data_test = np.random.permutation(np.asarray(data_test))



#get the Label column and delate the class column and rescale
y1 = data_train[:,0]
y2 = data_test[:,0]

y_train = (y1-min(y1))/(max(y1)-min(y1))
y_test = (y2-min(y2))/(max(y2)-min(y2))

data_train = np.delete(data_train,1,1)
data_test = np.delete(data_test,1,1)
```

```
#print the light curve
time = np.arange(len(data_train[0])) * (36/60)  # time in hours

plt.figure(figsize=(20,5))
plt.title('Flux of star 10 with confirmed planet')
plt.ylabel('Flux')
plt.xlabel('Hours')
plt.plot( time , data_train[10] )     #change the number to plot what you want



#normalized data
data_train_norm = normalize(data_train)
data_test_norm = normalize(data_test)




# function to apply gaussian filter to all data
def gauss_filter(dataset,sigma):

    dts = []

    for x in range(dataset.shape[0]):
        dts.append(gaussian_filter(dataset[x], sigma))

    return np.asarray(dts)



# apply the gaussian filter to all rows data
data_train_gaussian = gauss_filter(data_train_norm,7.0)
data_test_gaussian = gauss_filter(data_test_norm,7.0)




#print the light curves smoothed
plt.figure(figsize=(20,5))
plt.title('Flux of star 10 with confirmed planet, smoothed')
plt.ylabel('Flux')
plt.xlabel('Hours')
#plt.plot( time , data_train_gaussian[1000])
```

```
# apply FFT to the data smoothed
frequency = np.arange(len(data_train[0])) * (1/(36.0*60.0))

data_train_fft1 = scipy.fft.fft2(data_train_norm, axes=1)
data_test_fft1 = scipy.fft.fft2(data_test_norm, axes=1)

data_train_fft = np.abs(data_train_fft1)   #calculate the abs value
data_test_fft = np.abs(data_test_fft1)



#get the length of the FFT data, make something here below in order to make the
sequences of the same size
# only if they have differet dimensions

len_seq = len(data_train_fft[0])



#plot the FFT of the signals
plt.figure(figsize=(20,5))
plt.title('flux of star 1 ( with confirmed planet ) in domain of frequencies')
plt.ylabel('Abs value of FFT result')
plt.xlabel('Frequency')
plt.plot( frequency, data_train_fft[1] )



#oversampling technique to the data
data_train_ovs, y_train_ovs = custom_random_oversampling(data_train_fft, y_train,
sampling_strategy=0.5, random_state=42)



#recap dataset after oversampling
print("After oversampling, counts of label '1': {}".format(sum(y_train_ovs==1)))
print("After oversampling, counts of label '0': {}".format(sum(y_train_ovs==0)))



#reshape the data for the neural network model
data_train_ovs = np.asarray(data_train_ovs)
data_test_fft = np.asarray(data_test_fft)
```

```
data_train_ovs_nn = data_train_ovs.reshape((data_train_ovs.shape[0],
data_train_ovs.shape[1], 1))
data_test_fft_nn = data_test_fft.reshape((data_test_fft.shape[0],
data_test_fft.shape[1], 1))



#create F.C.N model and run it
model = m.FCN_model(len_seq)

model.compile(loss='binary_crossentropy',
optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),metrics=['accuracy'])

print(model.summary())

history = model.fit(data_train_ovs_nn, y_train_ovs , epochs=15, batch_size = 10,
validation_data=(data_test_fft_nn, y_test))


#save the model if you want
#model.save("/home/senad/DataSet/exoplanet_model_2")


#load the model if already exsist
#model = tf.keras.models.load_model("/home/senad/DataSet/exoplanet_model_2")


acc = history.history['accuracy']
#acc_val = history.history['val_accuracy']
epochs = range(1, len(acc)+1)
plt.plot(epochs, acc, 'b', label='accuracy_train')
#plt.plot(epochs, acc_val, 'g', label='accuracy_val')
plt.title('accuracy')
plt.xlabel('epochs')
plt.ylabel('value of accuracy')
plt.legend()
plt.grid()
plt.show()




loss = history.history['loss']
#loss_val = history.history['val_loss']
epochs = range(1, len(acc)+1)
plt.plot(epochs, loss, 'b', label='loss_train')
```

```
#plt.plot(epochs, loss_val, 'g', label='loss_val')
plt.title('loss')
plt.xlabel('epochs')
plt.ylabel('value of loss')
plt.legend()
plt.grid()
plt.show()



#predict the test set and plot results
y_test_pred = model.predict(data_test_fft_nn)
y_test_pred = (y_test_pred > 0.5)


accuracy = accuracy_score(y_test, y_test_pred)
print("accuracy : ", accuracy)

print(classification_report(y_test, y_test_pred, target_names=["NO exoplanet
confirmed","YES exoplanet confirmed"]))

conf_matrix = confusion_matrix([int(x) for x in y_test ], [int(y) for y in y_test_pred
])
sns.heatmap(conf_matrix, annot=True, cmap='Blues')



#create SVC model, predict and plot the results
SVC = m.SVC_model()
SVC.fit(data_train_ovs, y_train_ovs)

y_pred_svc = SVC.predict(data_test_fft)


print(classification_report(y_test, y_pred_svc, target_names=["NO exoplanet
confirmed","YES exoplanet confirmed"]))


conf_matrix = confusion_matrix([int(x) for x in y_test ], [int(y) for y in y_pred_svc
])
sns.heatmap(conf_matrix, annot=True, cmap='Blues')
```

OUTPUTS:

After oversampling, counts of label '1': 2525

After oversampling, counts of label '0': 5050

_____

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 3190, 256)	2304
max_pooling1d (MaxPooling1D)	(None, 638, 256)	0
batch_normalization (BatchNormalization)	(None, 638, 256)	1024
conv1d_1 (Conv1D)	(None, 633, 340)	522580
max_pooling1d_1 (MaxPooling1D)	(None, 127, 340)	0
batch_normalization_1 (BatchNormalization)	(None, 127, 340)	1360
conv1d_2 (Conv1D)	(None, 124, 256)	348416
max_pooling1d_2 (MaxPooling1D)	(None, 25, 256)	0
batch_normalization_2 (BatchNormalization)	(None, 25, 256)	1024
flatten (Flatten)	(None, 6400)	0
dropout (Dropout)	(None, 6400)	0
dense (Dense)	(None, 24)	153624
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: 1,030,745

Trainable params: 1,029,041

Non-trainable params: 1,704

None

Epoch 1/15

758/758 [==========] - 387s 498ms/step - loss: 0.1361 - accuracy: 0.9506 - val_loss: 0.0407 - val_accuracy: 0.9877

Epoch 2/15

758/758 [==========] - 363s 479ms/step - loss: 0.0336 - accuracy: 0.9884 - val_loss: 0.0359 - val_accuracy: 0.9947

Epoch 3/15

758/758 [==========] - 348s 460ms/step - loss: 0.0199 - accuracy: 0.9949 - val_loss: 0.0378 - val_accuracy: 0.9930

Epoch 4/15

758/758 [==========] - 315s 415ms/step - loss: 0.0153 - accuracy: 0.9943 - val_loss: 0.0097 - val_accuracy: 0.9965

Epoch 5/15

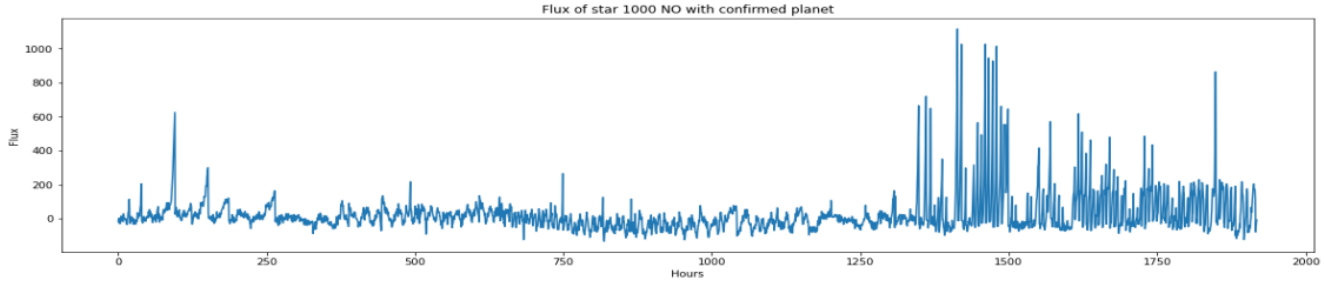
758/758 [==========] - 331s 437ms/step - loss: 0.0120 - accuracy: 0.9966 - val_loss: 0.0275 - val_accuracy: 0.9947

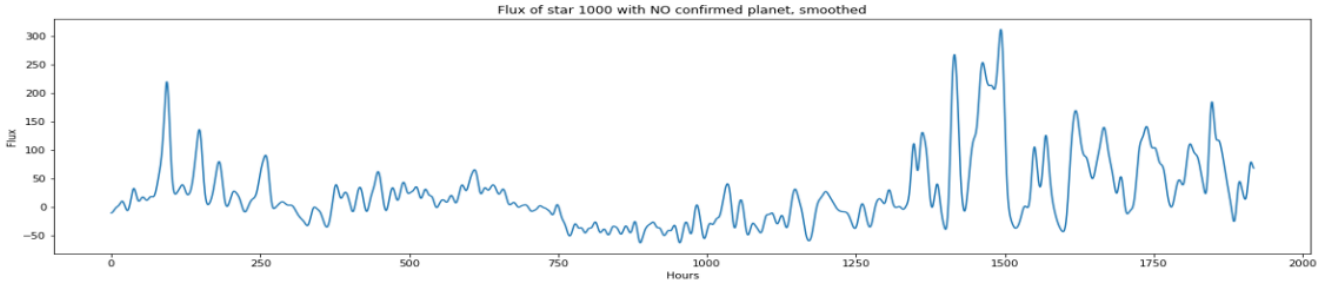
...

Epoch 15/15

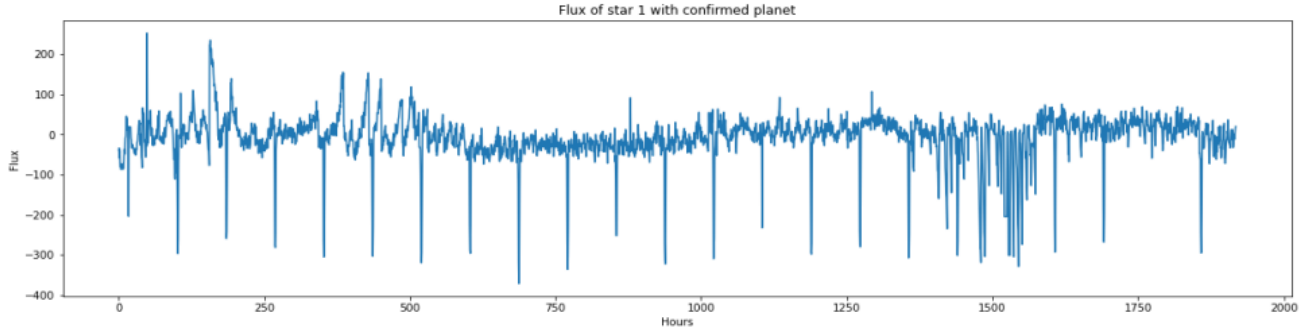
758/758 [==========] - 351s 429ms/step - loss: 0.0083 - accuracy: 0.9956 - val_loss: 0.0085 - val_accuracy: 0.9981

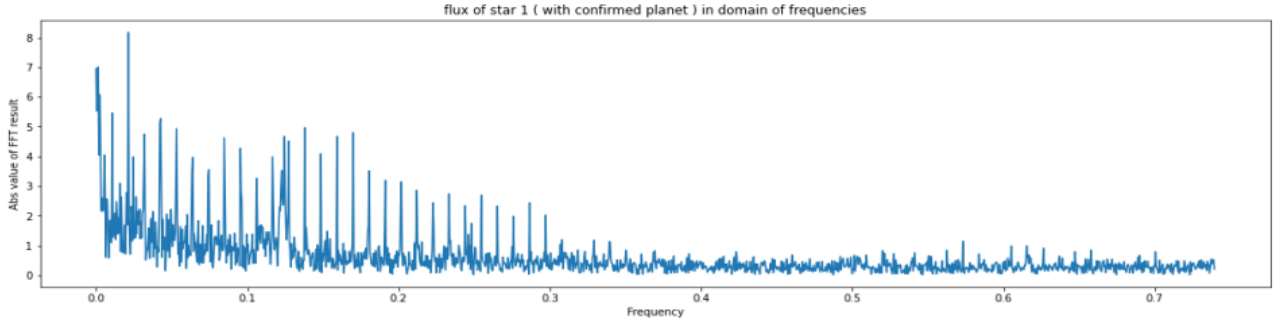
Light curve of the Star 1000 with NO confirmed planet, first original then smoothed (No normalized) :





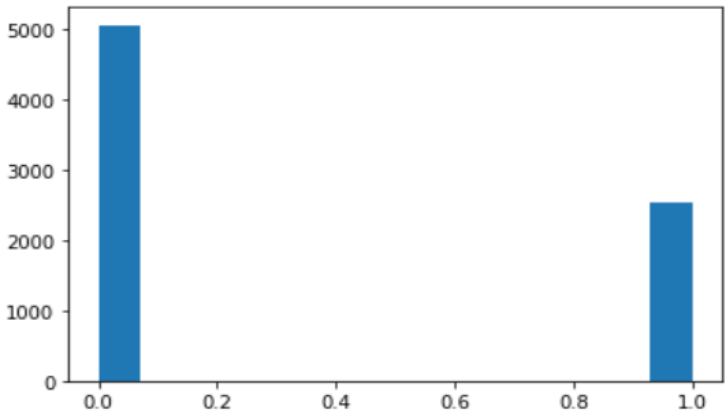
Light curve of the Star 1 with confirmed planet, first original then smoothed (No norm.). In the end, amplitude of the FFT data (this time the signal was normalized with L1) :





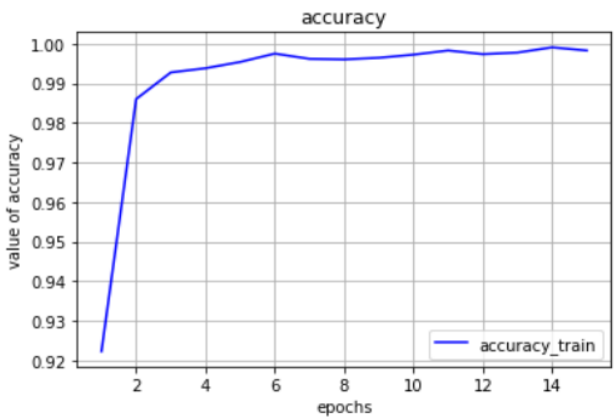
→*We use an oversampling method to make the dataset more balanced.*

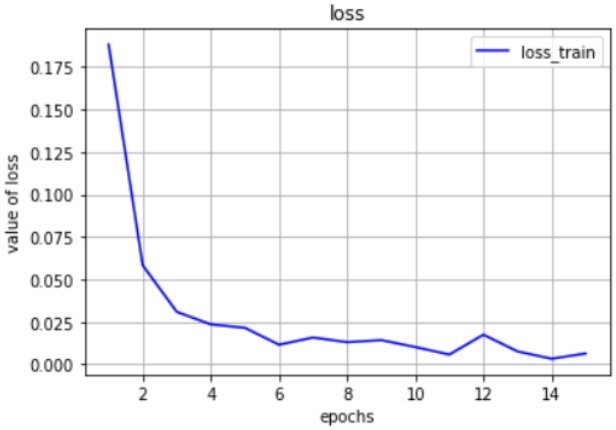
After applying oversampling to the dataset we obtained a new dataset with the following distribution of the classes :



→*Now the dataset is balanced.*

→ *performance of the network during training :*

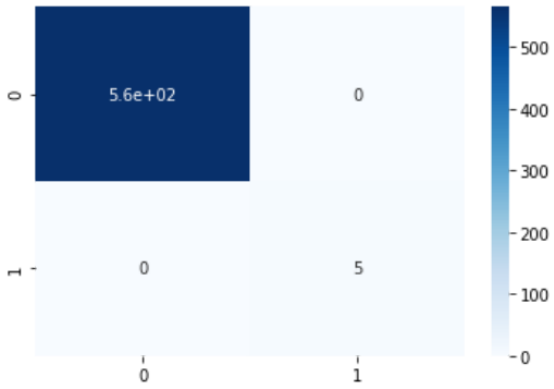




→*The Accuracy tends to 1 and the loss function tends to 0. This agrees with our theoretical hypotheses.*

→ *Let's see the predictions on the test set:*

Confusion matrix :



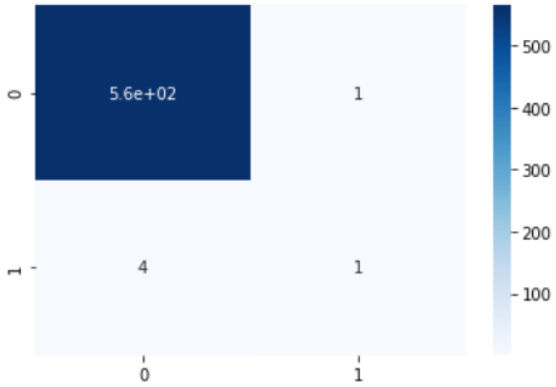
Other metrics :

	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.

→ *SVC model 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

SVC model only finds 1 exoplanet and fails to find the other 4.

CONCLUSION:

We can see that the CNN model finds all the 5 exoplanets and the SVC model finds only an exoplanet; it failed to recognize the other 4 positive examples.

Hence, we can say that the deep neural network approach in this application has better results than SVM approach.

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] https://www.kaggle.com/keplersmachines/kepler-labelled-time-series-data

[3] https://www.uniroma1.it/en/offerta-formativa/dottorato/2019/astronomy-astrophysics-and-space-science