UPPSALA UNIVERSITET

TVE-F 19021
Examensarbete 15 hp
Maj 2019

Detecting exoplanets with machine learning

A comparative study between convolutional neural networks and support vector machines

Sofia Dreborg
Maja Linderholm
Jacob Tiensuu
Fredrik Örn

UPPSALA UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Detecting exoplanets with machine learning

Sofia Dreborg, Maja Linderholm, Jacob Tiensuu, Fredrik Örn

In this project two machine learning methods, Support Vector Machine, SVM, and Convolutional Neural Network, CNN, are studied to determine which method performs best on a labeled data set containing time series of light intensity from extrasolar stars.

The main difficulty is that in the data set there are a lot more non exoplanet stars than there are stars with orbiting exoplanets. This is causing a so called imbalanced data set which in this case is improved by i.e. mirroring the curves of stars with an orbiting exoplanet and adding them to the set. Trying to improve the results further, some preprocessing is done before implementing the methods on the data set. For the SVM, feature extraction and fourier transform of the time-series are important measures but further preprocessing alternatives are investigated. For the CNN-method the time-series are both detrended and smoothed, giving two inputs for the same light curve. All code is implemented in python.

Of all the validation parameters recall is considered the main priority since it is more important to find all exoplanets than finding all non exoplanets. CNN turned out to be the best performing method for the chosen configurations with 1.000 in recall which exceeds SVM’s recall 0.800. Considering the second validation parameter precision CNN is also the best performing method with a precision of 0.769 over SVM's 0.571.

Handledare: Kerstin Ersson
Ämnesgranskare: Teresa Zardán Gómez de la Torre
Examinator: Martin Sjödin
ISSN: 1401-5757, 19021

Populärvetenskaplig sammanfattning

Begreppet maskininlärning innebär att en modell genom att hantera stora mängder data lär sig mönster så bra att den kan använda dessa kunskaper för att klassificera ny data utan att bli specifikt programmerad för detta. Två maskininlärningsmetoder, "Support Vector Machine", SVM, och "Convolutional Neural Network", CNN, har i detta projekt implementerats på data från NASAs Keplerteleskop. Teleskopet har under nio år mätt ljusintensiteten från stjärnor bortom vårt solsystem och denna ljusintensitet kan analyseras med en metod som kallas transitmetoden. Metoden utnyttjar att den uppmätta ljusintensiteten från stjärnan sjunker något om en planet passerar framför den då planeten blockerar en andel av stjärnans strålar. Detta möjliggör upptäckt av planeter som kretsar kring stjärnor långt ifrån vårt eget solsystem.

Studien besvarar vilken av maskininlärningsmetoderna som ger bäst resultat för en viss datamängd, det vill säga. vilken metod som får flest rätt när den gissar om det är eller inte är en planet i omlopp runt en viss stjärna. Metoderna tillhör båda den del av maskininlärning som kallas "supervised machine learning" som innebär att modellerna tränas på dataset där svaret är givet, vilket innebär att metoden lär sig genom att se många exempel på hur tidigare data har klassificerats.

Av de modellkonfigurationer som testats visade sig CNN vara den metod som presterade bäst på detta dataset.

Contents

1 Introduction

Extra solar planets, also known as exoplanets, are planets that orbit stars outside of our solar system. The hunt for exoplanets first started in 1983 when William Borucki, working for NASA's Ames Research Center, started the search for Earth-sized planets in solar systems beyond our own. He and his team worked with a technique called transit photometry which measures a slight decrease in light intensity when a planet crosses in front of its star. In 1995 they discovered the first confirmed exoplanet, a hot half Jupiter-sized planet orbiting around a sun-like star.

Much later, after being rejected by NASA four times, the Kepler mission was launched on the 6th of March in 2009. Part of the mission's scientific goal was to look for exoplanets, especially such which could support life. The telescope continued to collect data until October 30th in 2018 and contributed to incredible scientific progress and discoveries [1].

Kepler's first exoplanet discovery was made in 2010, but the data set used in this study is from Campaign-3, during the winter 2014/2015. [2] The data is measurements of light intensity over time recorded over approximately 80 days. Each observed star is also labeled as either positive or negative (depending on if it has an exoplanet in orbit).

The question to answer is how well the machine learning methods SVM- Support Vector Machine and CNN- Convolutional Neural Network perform on a data set containing time-series of the light intensity from extrasolar stars. The task is called a binary classification problem since there are only two classes, true for an exoplanet and false for no exoplanet.

The method Support Vector Machine (SVM) classifies data into two categories with a so-called hyperplane. For a two dimensional set the hyperplane is a line that best separates the two classes of data. The dimension of the problem is determined by the number of features that describes the data. For features in n-dimensions the hyperplane's dimension is n-1 and accordingly the feature extraction is an important step of the preprocessing of data. The ambition is to create a hyperplane that best separates the data-set and this is the hyperplane with the largest margin between the hyperplane and the nearest data points on both sides.

The convolutional neural network (CNN) is a version of neural network that utilizes convolutions to detect patterns and reduces the size of the input data. A convolutional layer uses a convolution window that sweeps the input data. CNN requires far less parameters since all input data and hidden parameters are not connected in contrast to other neural network versions. In addition to convolutional layers, CNN can use pooling, normalization and dropout layers to reduce data size and prevent overfitting.

Machine learning can be divided into two branches, supervised and unsupervised machine learning. The goal of supervised machine learning is to build a model that can predict a known answer as well as possible. Unsupervised machine learning is when the model, given an unlabeled data-set, learns patterns in the given data and therefore can classify the data. Both SVM and CNN are in the supervised machine learning branch since both methods are trained on data-sets where there is known whether it is an exoplanet or not in orbit around a star [3].

To evaluate the performance of the machine learning algorithms different metrics are used. Precision and recall are calculated and plotted against each other. For the trained algorithm the predictions are compared in a confusion matrix.

2 Theory

2.1 Exoplanet detecting

When a planet for an observer crosses in front of a star a so called transit occurs. The phenomenon have been used for centuries to observe movements and properties of planets in our solar system and for this project it is used for detecting exoplanets. When a body crosses in front of a star it blocks a portion of the rays which will result in a decrease in light intensity [4]. This is illustrated in figure 1

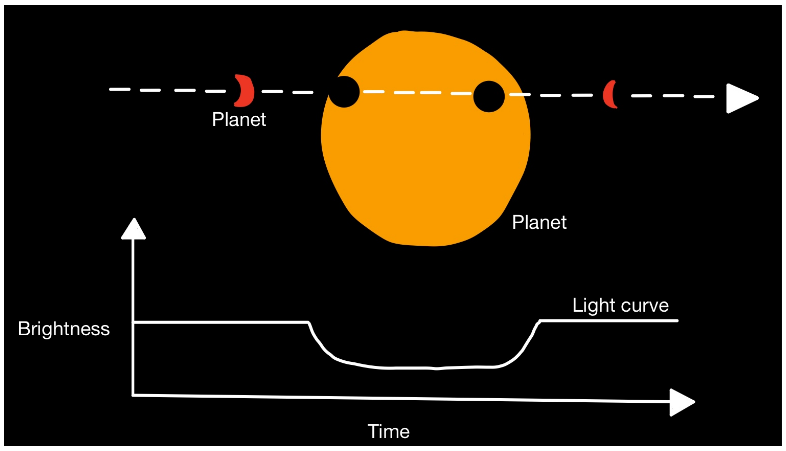


Figure 1: Graphical description of a transit, when the planet crosses in front of the star the brightness of the star will decrease

The shape of the light curve contains information about orbiting planets such as size or orbiting radius. A bigger planet will block a larger portion of the star causing a larger dip in brightness. For a star with large orbiting radius the time of the transit will be longer [5]. There are other methods for detecting exoplanets, for example radial velocity where due to Newton's law of universal gravitation a star with an orbiting planet will wobble slightly causing shifts in the color of the stars light. Today (11th of April 2019) there are 3933 confirmed exoplanets where 77.4% of them have been detected with the transit method, 18.5% by radial velocity and 3% with other methods [6].

2.2 Data set

For each star the light intensity is measured over time and binary labeled with either "1" for "no exoplanet in orbit" or "2" for "at least one exoplanet in orbit". The data used in this project originates from observations made by the NASA Kepler space telescope. Disturbance noise from the telescope has been removed and detections from other campaigns has been added to boost the data set. Since the number of stars without an orbiting planet is much greater than the number of stars with an orbiting planet the data set is unbalanced with ratio 99% of the stars labeled as having no exoplanet in orbit. Confirmed exoplanets from other campaigns are also included to make the data less unbalanced, though the contribution will not make enough compensation in order for the data set to be classified as balanced. This imbalance is one of the greatest challenges with the data. The data is downloaded as csv-files from a host at kaggle.com. The host has converted the files from the open source data hosted at the Mikulski Archive [7] to which the Kepler telescope beams down data to Earth with constant time intervals.

The data is split into one training set and one test set. The training set consists of 5087 observed stars with 37 confirmed exoplanet-stars and the test set of 570 observed stars with five confirmed exoplanet-stars. Both data sets are measured with 3198 time steps, corresponding to about 80 days of observations [8].

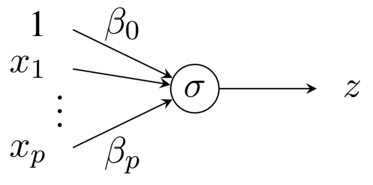
2.3 Neural Networks

Neural networks are a branch of supervised machine learning and can be used for both regression and classification of data. It's based on the concepts of the simpler models linear- and logical regression and is used within image analysis, speech recognition and language translation [3]. The architecture of a neural network model is inspired by how neurons in the human brain works and is useful for recognizing non-trivial patterns between input and output data. A neural network consists of at least three so called layers where each layer consists of a set of neurons. The first layer is the input layer where the data is received, the number of neutrons in this layer is equal to the number of data points. The last layer is the output layer where the number of nodes, for a classification problem, is equal to the number of classes. Inbetween the input layer and output layer there are at least one hidden layer where the nodes in the hidden layer transforms the input with weights before passing them on in order to get a prediction for the output. The term "deep learning" is used to describe complex networks with many hidden layers with different weights, which can complete more complex tasks [9].

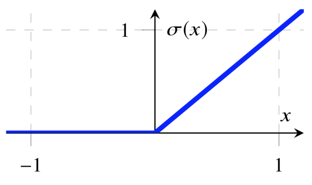
A neural network is constructed by using a generalized linear regression model, which is a linear regression model with the parameters β_i to which a scalar activation function σ is applied. The generalized linear regression model is a non-linear function which predicts the output z from the input $\mathbf{x} = [1\ x_1\ x_2\ ...\ x_p]^\top$, which for a neural network with one hidden layer it is given by:

$$z = \sigma(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p) \tag{1}$$

The activation function may be any chosen function, but for this project the rectified linear unit (ReLU) is used. The regression parameters and the activation function are graphically described in figure 2



(a)



(b)

Figure 2: Graphical description of a generalized linear regression (Figure 2a) and the ReLU-function which is defined by $\sigma(x) = \max(0, x)$ (Figure 2b). Original image from [3] courtesy of Andreas Lindholm, Niklas Wahlström, Fredrik Lindsten, Thomas B. Schön, Department of Information Technology Uppsala University

When adding hidden layers to the neural network the output of the first hidden layer is the input to the second hidden layer and so on. Denoting the output from neuron i as h_i and the number of neurons in a certain layer as M, then the set of output from a certain layer is $\{h_i\}_{i=1}^{M}$. For simplification of notations, define the offset vector $\mathbf{b}$ and the weight vector $\mathbf{W}$. Let l denote the layer where $l = 1, 2, ..., L$ and let p be equal to the number of input variables for the layer l.

$$\mathbf{b}^{(l)} = [\beta_{01} \ldots \beta_{0M}] \quad \mathbf{W}^{(l)} = \begin{pmatrix} \beta_{11}^{(j)} & \cdots & \beta_{1M}^{(j)} \\ \vdots & \cdots & \vdots \\ \beta_{p1}^{(j)} & \cdots & \beta_{pM}^{(j)} \end{pmatrix} \quad \mathbf{b}^{(L)} = [\beta_0^{(L)}] \quad \mathbf{W}^{(L)} = \begin{pmatrix} \beta_1^{(L)} \\ \vdots \\ \beta_M^{(L)} \end{pmatrix} \tag{2}$$

For a classification problem a softmax function is used on the output $\mathbf{z}$ to assign a label to the output data. Hence a deep neural network can be defined with:

$$\begin{aligned} \mathbf{h}^{(1)} &= \sigma(\mathbf{W}^{(1)\top}\mathbf{x} + \mathbf{b}^{(\mathbf{1})\top}) \\ \mathbf{h}^{(2)} &= \sigma(\mathbf{W}^{(2)\top}\mathbf{h}^{(1)} + \mathbf{b}^{(\mathbf{2})\top}) \\ &\vdots \\ \mathbf{h}^{(L-1)} &= \sigma(\mathbf{W}^{(L-1)\top}\mathbf{h}^{(L-2)} + \mathbf{b}^{(\mathbf{L-1})\top}) \\ \mathbf{z} &= \mathbf{W}^{(L)\top}\mathbf{h}^{(L-1)} + \mathbf{b}^{(\mathbf{L})\top} \\ [\mathrm{p}(1|\mathbf{x}_\mathrm{i})\ \mathrm{p}(2|\mathbf{x}_\mathrm{i})\ \ldots \mathrm{p}(\mathrm{K}|\mathbf{x}_\mathrm{i})]^\top &= \mathrm{softmax}(\mathbf{z}). \end{aligned} \tag{3}$$

A visualization of a deep neural network is shown in figure 3.

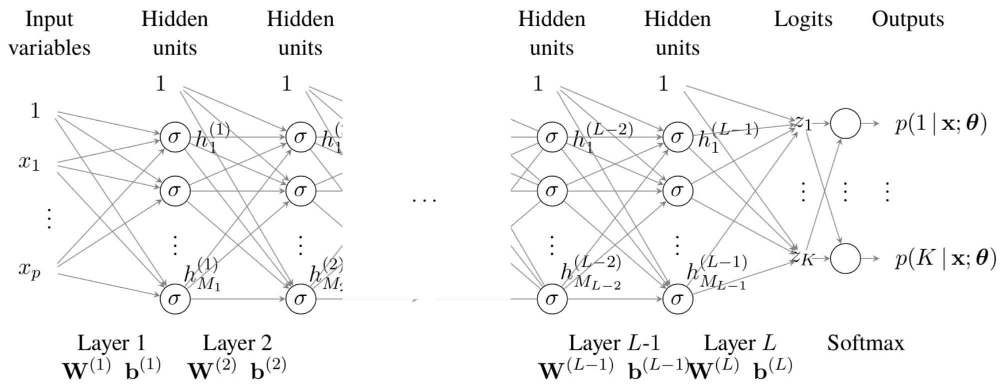

Input variables
Hidden units
Hidden units
Hidden units
Hidden units
Logits
Outputs
1
x_1
x_p
$h_1^{(1)}$
$h_{M_1}^{(1)}$
$h_1^{(L-2)}$
$h_{M_{L-2}}^{(L-2)}$
$h_1^{(L-1)}$
$h_{M_{L-1}}^{(L-1)}$
z_1
z_K
$p(1 \mid \mathbf{x}; \boldsymbol{\theta})$
$p(K \mid \mathbf{x}; \boldsymbol{\theta})$
Layer 1 $\mathbf{W}^{(1)}$ $\mathbf{b}^{(1)}$
Layer 2 $\mathbf{W}^{(2)}$ $\mathbf{b}^{(2)}$
Layer L-1 $\mathbf{W}^{(L-1)}$ $\mathbf{b}^{(L-1)}$
Layer L $\mathbf{W}^{(L)}$ $\mathbf{b}^{(L)}$
Softmax

Figure 3: Graphical description of a deep neural network for a classification problem. Original image from [3] courtesy of Andreas Lindholm, Niklas Wahlström, Fredrik Lindsten, Thomas B. Schön, Department of Information Technology Uppsala University

The softmax functions input parameters $z_1, ..., z_K$ are called logits. To learn the classification network the cross entropy loss function is used. With this approach the vector of predicted probabilities is compared to the one hot encoded output vector. From this comparison the cross entropy loss function is minimized and the networks parameters are optimized. The cross entropy loss function also helps to avoid numerical problems when the the probability for a prediction, $p(k|\mathbf{x}_i, \theta)$ is close to zero, by compensating for the effects caused by the softmax function. The cross entropy loss function is given by the following equation: [3]

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} L(\mathbf{x}_i, \mathbf{y}_i, \theta) \quad \textit{where} \quad L(\mathbf{x}_i, \mathbf{y}_i, \theta) = -\sum_{k=1}^{K} y_{ik} log\, p(k|\mathbf{x}_i; \theta) \tag{4}$$

2.4 Convolution neural network

The 1-dimensional convolution neural network (1D CNN) uses kernels that sweeps the input vector and convolves the parameters with a set of weights. The convolution layer does not connect each input variable and hidden unit with it's own unique parameter like a dense layer would. Hence the convolution layer requires far less parameters and encodes the same amount of information with fewer variables. Several convolution layers can be used to distinguish different features in the data. The stride controls the step size the kernels take when sweeping the data, which can be used to reduce the size of the input to the next layer [3]. This is illustrated in figure 4. Pooling can also be used to reduce the size of the input. The pooling function replace a net of the inputs with some statistic representation of neighbouring inputs. An example is the max pooling function that operates within a set interval an replaces the outputs with the maximum value within that interval [10].

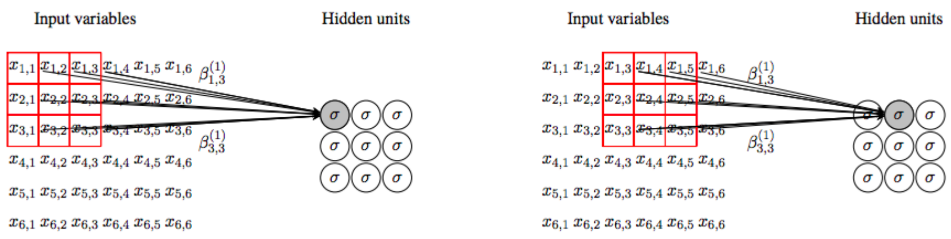


Figure 4: Graphical description of a convolutional window sweeping the input parameters. Original image from [3] courtesy of Andreas Lindholm, Niklas Wahlström, Fredrik Lindsten, Thomas B. Schön, Department of Information Technology Uppsala University

To capture more details in a convolution layer, several kernels can be added. Each kernel have it's own set of parameters and hidden units, which are called channels. The hidden units in one layer is collected into a tensor of dimension; rows x columns x channels. The kernels in a CNN architecture depends on all channels in previous layers, hence a weight tensor, W, containing all kernel parameters would be of dimensions; kernels rows x kernels columns x input channels x output channels. Both dense and convolution layers can be implemented to the network. For classification problems the last layer would be a softmax layer to squeeze the output between $[0, 1]$.
To train a CNN the parameters $\hat{\theta}$ have to be optimized. This is done by numerical optimization of the loss function. The Adam optimizer is a combination of two different optimizations methods. Namely stochastic gradient decent were the gradient for the loss function is calculated and root mean square propagation were the average of recent gradient values is calculated. The Adam optimizer changes the learning rate for the networks weights as the training progresses [11].

2.5 Support vector machines

Support vector machines are used for classifying works by separating the different classes with a hyperplane. The best separation is achieved by maximizing the distance between the hyperplane and the nearest training points, the so-called margin [12]. This is illustrated in figure 5. In the figure, the concept is illustrated in two dimensions, where the hyperplane becomes a line, but theoretically this can be done with any number of dimensions. The dimension of the hyperplane is always one dimension less than the dimension of the data points.

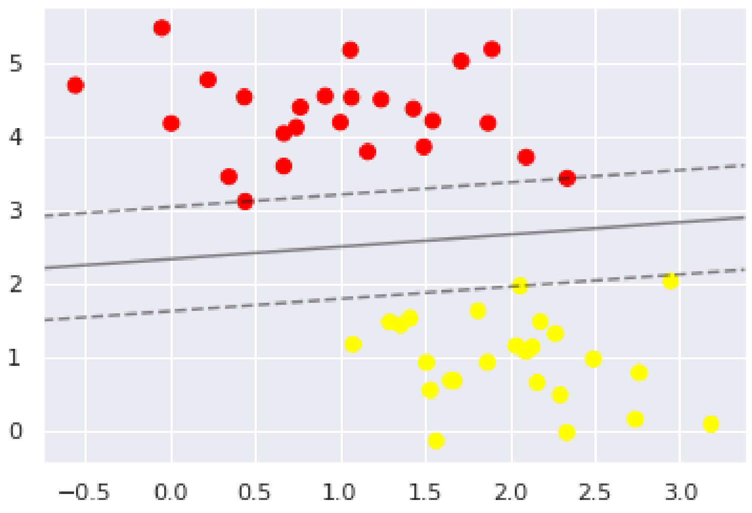


Figure 5: 2D example of classes separated by a hyperplane. The support vectors are the dots on the dashed line. Source code for image [14]

The classes are often denoted as +1 and -1 and the classifier resulting from the hyperplane classifies:

$$\begin{array}{ll} \mathbf{w}^T\mathbf{x} + b \geq 0 & class + 1 \\ \mathbf{w}^T\mathbf{x} + b < 0 & class - 1 \end{array} \tag{5}$$

In the book "Bayesian Reasoning and Machine Learning", Barber [15] describes how to construct a more robust classifier. The data from the different classes should be separated from each other by some finite amount to achieve a margin. For the trained data, this results in the classification boundaries

$$\begin{array}{ll} \mathbf{w}^T\mathbf{x} + b > \epsilon^2 & class + 1 \\ \mathbf{w}^T\mathbf{x} + b < \epsilon^2 & class - 1. \end{array} \tag{6}$$

Since the problem can be re-scaled arbitrary, ϵ is often set to 1. When maximizing the margin, i.e the distance between the two closest points from the different classes (support vectors) along the vector $\mathbf{w}$, the calculation ends up as:

$$\frac{\mathbf{w}^T}{\sqrt{\mathbf{w}^T\mathbf{w}}}(\mathbf{x}_+ - \mathbf{x}_-) = \frac{2}{\sqrt{\mathbf{w}^T\mathbf{w}}} \tag{7}$$

This maximization can be rewritten as a minimization for $\mathbf{w}^T\mathbf{w}$. The problem is constrained by having to make correct classifications for all y resulting in the *quadratic programming* problem:

$$\min \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} \qquad \text{s.t} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \qquad i = 1, ..., N \tag{8}$$

The quadratic program is then minimized using a suitable optimization algorithm. However, this formulation is built on the assumption that the classes are linearly separable, which might not be the case. Barber describes how to tackle this using a soft margin support vector machine. This is done by relaxing the constraint by adding a slack factor ξ_i for each data point, where $\xi_i \geq 0$. The value of ξ_i is equal to the violation of the decision boundary for the data point $\mathbf{x}_i$ and 0 if there is no violation. The data point is still on the correct side of the decision boundary as long as $\xi_i < 1$. This use of slack variables relaxes the constraint to

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i. \tag{9}$$

The size of the violations should still be limited and to achieve this, the cost function to be minimized (8) should also include a term with the sum of all ξ_i or their squares, two common methods which are further explained by Barber [15]. In this study the squared hinge loss has been used for *LinearSVC*, while the *SVC* classifier uses the ordinary hinge loss [12]. This means the final formulation of the cost function is as in equation 10 for the *SVC* and the same but with ξ_i^2 for *LinearSVC*.

$$\min \quad (\frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i) \qquad \text{s.t} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \qquad i = 1, ..., N \tag{10}$$

2.5.1 Class imbalance

The soft margin support vector machine is sensitive to imbalanced data sets and there are several ways to tackle this. In this project the focus is on class weights which means that the model is manipulated to prioritize the minority class above the majority class. The hyperplane that separates classes in SVM tend to become skewed towards the minority class which cause the model to perform well for the majority class but underperform for the minority class. Another consequence of an imbalanced data set is that the support vectors becomes unbalanced. The class weights improve this problem by adding a higher cost for misclassification of the minority class than for misclassification of the majority class, resulting in the cost function shown in equation 11, with the same boundaries as in equation 10. Hopefully, this causes less skewness and a better performing model [16].

$$\min \quad (\frac{1}{2}\mathbf{w}^T\mathbf{w} + C^+ \sum_{i|y_i=1} \xi_i + C^- \sum_{i|y_i=-1} \xi_i) \tag{11}$$

2.5.2 Kernels

In machine learning kernel methods are a way to make non-linear data separable by mapping the non-linear data set into a higher dimensional space. In this higher n-dimensional space the data is linearly separable and it is possible to find a hyperplane in n-1 dimensions to separate the classes. In figure 6 and 7 an easy example of the power of kernels are illustrated. The data points are clearly linearly inseparable in figure 6, but it is an obvious difference in radius between the data types. The kernels increases the dimension of the set by adding the radius dimension and in this dimension the classes are linearly separable.

As the support vector machine assumes linear data sets, which rarely exists in the nature, the so called kernel-trick is very useful. Kernels enable processing data in high dimensions as the method compute the inner product of the data pairs in the feature space. This makes the method computationally efficient even in high dimensions [17].

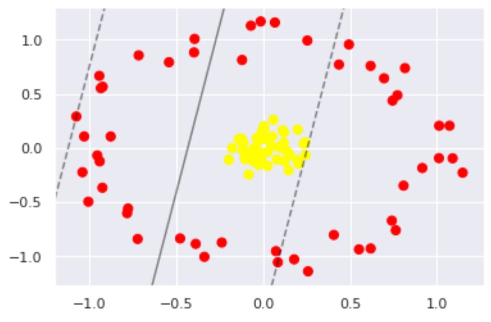


Figure 6: Linearly inseparable data points in 2 dimensions. Source code for image: [14]

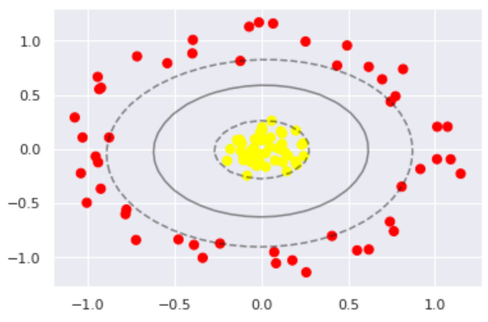


Figure 7: The same data set in 3 dimensions after adding the radius as a new dimension. Now the data is linearly separable in the radius dimension. Source code for image [14]

2.6 Classification of time series

Comparing and classifying time series can be challenging because one can choose both how the time series is represented and how to measure the difference between different series. The data can be represented as measurements at different points in time and the difference as how much the values of the measurements differ. This can be done either directly from the data or after some kind of processing or transformation has been made. Alternatively, they can be represented by features, or properties, which in some way describe the series. These features can be simple statistical measurements, like mean value or variance, but it could just as well be features specifically chosen to be useful to classify the time series in question. The choice of features can be more important than the choice of classifier [18]. In this study, something that can separate the time series where with exoplanet transits from the ones without is needed. The deep learning CNN-algorithm finds patterns by itself but for the SVM a choice of features must be made by the user. Another advantage of using features instead of using the time series data directly, is that it allows the method to classify signals of different length, since it will use the same number of input parameters anyway. In this study, *catch22* [19], a time series classification tool-box made to be used cross-disciplinary, is used to select the features.

2.7 Evaluation metrics

The goal of the algorithms is to be able to find exoplanets from the light curves and to compare them there should be distinct quantities measuring the performance. It is often common to use accuracy (correct predictions/total predictions) when evaluating machine learning models, but for imbalanced data sets, like the one used in this study, it can be misleading [20]. Though, accuracy is defined as:

$$A = \frac{TP + TN}{TP + TN + FP + FN} \tag{12}$$

Finding all or as many as possible of the exoplanets (not wrongly classifying exoplanet stars as non-exoplanet stars) was considered the most important task of the algorithm, so recall was chosen as the main performance metric. Recall is defined as

$$R = \frac{TP}{TP + FN} \tag{13}$$

Secondly, the algorithms should be able to tell when there is no exoplanet orbiting the star; an algorithm predicting exoplanets around all stars would not miss any positive sample but it would not be very useful anyway. Therefore, precision was chosen as the secondary performance-metric. Precision is defined as

$$P = \frac{TP}{TP + FP} \tag{14}$$

Both these metrics (13, 14) use the same abbreviations as in table 1. Choosing this performance metric, which is better suited for the imbalanced data, was inspired by Gabriel Garza [21]. To tackle the problem with imbalance, another study [20] done on exoplanet detection uses the F-measure, which weighs together the two measurements. The F-measure can consider the two metrics equally important, resulting in the F_1 score

$$F_1 = 2\frac{P \cdot R}{P + R} \tag{15}$$

To instead lend more weight to one or the other, the F_β score can be used instead

$$F_\beta = \frac{(\beta^2 + 1)P \cdot R}{\beta^2 P + R} \tag{16}$$

Choosing $0 < \beta < 1$ lends more weight to precision while $\beta > 1$ lends more weight to recall [22]. $\beta = 1$ results in the F_1 score. All of these metrics are available in the *sklearn* library and have their optimal value at 1 and worst at 0.

An easy way to check the performance of a machine learning algorithm is to use a confusion matrix, which is a two dimensional matrix that shows how many of each prediction the model have made. An ideal algorithm would therefore only have true negatives and true positives. Since it only handles positives and negatives, the confusion matrix (in this form) can only handle binary classification. The structure is shown below in table 1 [23].

Table 1: Confusion matrix: TN - True negative, FP - False positive, FN - False negative, TP - True positive.

		Predicted	
		0	1
Actual	0	TN	FP
	1	FN	TP

3 Method

The machine learning algorithms were written and implemented in python. Since training and evaluation requires good computational power all training and evaluation was done on Google Colab using GPU as hardware accelerator to reduce the runtime.

3.1 Packages

The convolution neural network was constructed using keras which is an open source library for python. The package contains premade methods for constructing CNN architecture.

The support vector machine algorithm was created using the scikit-learn package which contains methods for constructing a variety of different machine learning algorithms. To try out different configurations and variations of support vector classification the two sklearn methods *SVC* and *LinearSVC* where used. The evaluation metrics and *GridSearchCV*, used to find the best model and configuration, are also from the sklearn-library. To extract features from the time series, *catch22* was used.

3.2 Preprocessing

In the original data set the data is labeled with either "1" for no exoplanet in orbit or "2" for an exoplanet in orbit. In order for some coding features to work the data is reclassified with "0" for no exoplanet in orbit or "1" for an exoplanet in orbit.

Since the data set is imbalanced and the number of positive examples are in minority, more positive examples were generated. The number of positive labeled curves were doubled by mirroring the positive curves and adding them to the data set. For the CNN method even more examples were created by advancing the curves in time.

Figure 8 and 9 show a few examples of light curves of both confirmed exoplanets and confirmed non exoplanets. The figures illustrates the big variety in shape of the signal.

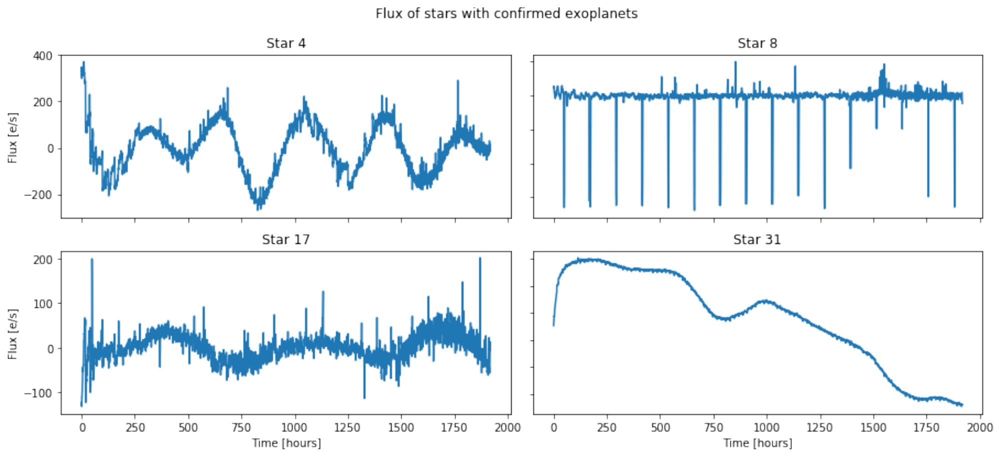


Figure 8: A few examples of light curves with a confirmed exoplanet

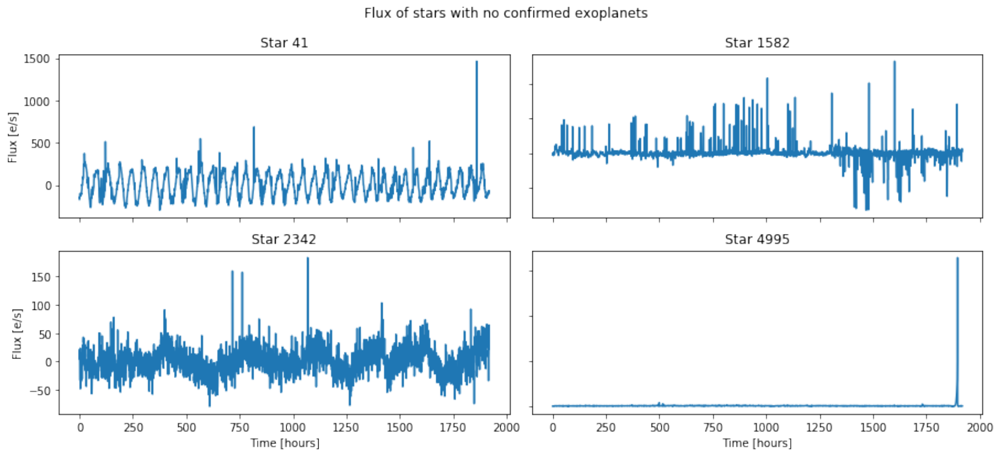


Figure 9: A few examples of light curves with a no confirmed exoplanets

The data is preprocessed in order to improve the performance of the model. For the CNN-method the data is detrended by smoothing a light curve with a gaussian filter and subtract it from the original light curve. This will result in a more flat representation of the curve. The flattened curve is mean normalized and fed into the network along with a zero mean unit variance normalization of the original curve, resulting in each star having two input signals. For the SVM-method the same normalized version of the original curve as for the CNN-method is used as an input.

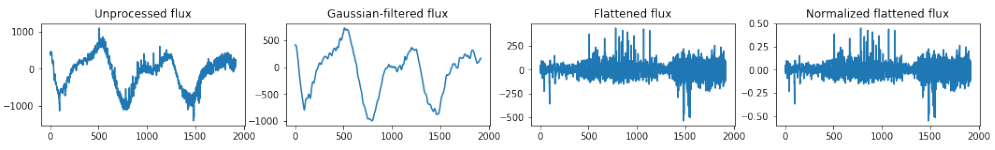


Figure 10: Illustration of the preprocessing for detrending a light curve

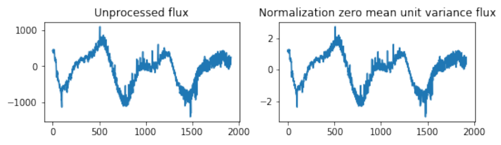


Figure 11: Illustration of the preprocessing for normalizing a light curve

In figure 10 and 11 the preprocessing steps are shown for both of the preprocessing methods. For the current star, the CNN-method would have both of the rightmost curves shown in the two figures as inputs while the SVM-method only would have the rightmost curve in figure 11.

3.3 Convolutional neural network

The CNN architecture consists of 17 layers. The first layers are a mixture of convolutional, max pooling and batch normalization layers. These layers are used to extract different patterns in the time series. How the convolutional layer and the max pooling layers operate is described in section 2.4. The batch normalization layer transforms the output of the previous layer to keep the mean activation close to 0 and the activation standard deviation close to 1. The flatten layer compresses the 3-dimensional tensor from the previous layer into a 1-dimensional vector. The dropout layers are applied to prevent the model from overfitting to the training data. The last layer will convert the output of the network into a probability, illustrating how likely it is to find an exoplanet orbiting the current star.

light flux with 3197 observations

$\Downarrow$

1d convolution layer: filters=8, kernel size=11, activation=linear

$\Downarrow$

1d max pooling layer: strides = 4

$\Downarrow$

batch normalization

$\Downarrow$

1d convolution layer: filters=16, kernel size=11, activation=ReLU

$\Downarrow$

1d max pooling layer: strides = 4

$\Downarrow$

batch normalization

$\Downarrow$

1d convolution layer: filters=32, kernel size=11, activation=ReLU

$\Downarrow$

1d max pooling layer: strides = 4

$\Downarrow$

batch normalization

$\Downarrow$

1d convolution layer: filters=64, kernel size=11, activation=ReLU

$\Downarrow$

1d max pooling layer: strides = 4

$\Downarrow$

flatten to 1d layer

$\Downarrow$

drop out of 0.5

$\Downarrow$

dense layer: nodes=64, activation=ReLU

$\Downarrow$

drop out of 0.25

$\Downarrow$

dense layer: nodes=64, activation=ReLU

$\Downarrow$

dense layer: nodes=1, activation=sigmoid

To train the network a batch generator function was written in python. The batch generator takes 32 stars from the training set and trains the network on these stars. New stars are extracted from the data set until all stars in the training set have been used. The process of extracting all stars in batches is repeated until a set number of epochs have been iterated. To initialize the training the model is first trained for a small number of epochs with a learning rate $\gamma = 1 * 10^{-5}$ to ensure the loss function converges. The model is then train for more epochs and with a larger learning rate, $\gamma = 4 * 10^{-5}$. For both training configurations the Adam optimizer was used.

3.4 Support vector machine

For development of the SVM-method the following parameters and methods have been varied:

- **Normalization:** Before using any classification algorithm or extracting features, the signal could be normalized.
- **Choosing input data:** Three different types of input types were tried, (1) the data points in the time domain, (2) the fourier transformed data and (3) features extracted from the series with *catch22*.
- **Different kernels:** With the *SVC* the rbf kernel, polynomial kernel, with different degrees, and linear kernel where tried. The *LinearSVC* kernel was also used, which is similar to the *SVC* with linear kernel, but should scale better. [12]
- **Class weights:** To counter the imbalance of the data set, different class weights were tried.

To generate precision recall-curves the SVC model has an option (*probability=True*) for generating probabilities, which the original model does not [12].

3.4.1 Fast fourier transform

As an alternative to using the data directly in the time domain, while still not doing a feature extraction, the discretely fourier transformed data was tested as input for the SVM. This was done with the *numpy.fft*-method. The frequency information of the signal was thought to be interesting since orbiting planets should result in repeating decreases in light.

3.4.2 Feature extraction

To extract features from the time series *catch22*, which is a set of 22 time series features, filtered from the much bigger feature set toolbox *hctsa*, which contains 7658 different features. The *catch22*-features have been chosen by testing the performance of all *hctsa*-features on 93 different sets of time series. The reduction of feature causes *catch22* to have a 7% reduced accuracy compared to the *hctsa* but a 1000-fold reduction in computation time and the scaling with the time series length is near linear. This feature set was created to enable a method with a minimal amount of features that still shows a strong classification performance across a provided collection of time-series problems [19]. Another feature extractor called *tsfresh* [13] was implemented on the data set but since the extracting of features took a long time to compute it was replaced by the simpler *catch22*. The long computation time was considered disproportionate to the benefit of having more features. *catch22* is also a lot less complex than *tsfresh* and does not require MatLab, which *hctsa* does. Since *catch22* is based on C, the features were extracted on a Linux computer and saved into csv-files [19].

3.4.3 Comparing model configurations

To compare the different SVM-models, the model selection tool *GridSearchCV* was used. *GridSearchCV* can take all the different model-parameters to be varied as input and evaluate them by using cross-validation on the training set. This means that the training set is split/folded into multiple sets (which preserves the rate of positive/negative samples), the algorithm is trained on one part of the training set and evaluated on the other. This is done to examine how the model might perform on unseen data, before trying it on the final test set. This procedure was done multiple times and the average result from the different splits/folds was returned [24]. The user chooses which score to evaluate and since this study handles an imbalanced data set, recall and precision has been used, with recall considered a higher priority than precision. To weigh both these metrics in, but placing more priority on recall the F_β score with $F\beta = 2$ (see equation 16) was selected as scorer in the grid search. The validation was done with three-fold cross validation. The varied parameters tested with grid search were: *kernel*, *class_ weight*, and *gamma*. The classifiers were all tried with the (1) time domain data, (2) fourier transformed data and (3) feature based inputs. Since lots of parameters were varied and each model was trained multiple times in the cross validation the code is quite computationally heavy. Therefore the code was run on Google Colab. The best performing kernel and input data format were explored further.

4 Result

4.1 Training data

The models where trained and evaluated on the training data with 5124 observed stars (of which 37 are the reversed duplicates). The convolutional neural network was trained for a batch size equal to 32 and the threshold was chosen to 0.79. The best performing SVM-method was the *SVC* with a linear kernel, using fourier transformed, non-normalized data and balanced class weights. The results of a selected training session are presented in a confusion matrix in table 2 below and with different performance metrics in table **??**, where the AUC is the area under precision-recall curve.

Table 2: Confusion matrix for CNN 2a) and SVM 2b) on the training data

a)		Predicted	
		0	1
Actual	0	4995	55
	1	0	74

b)		Predicted	
		0	1
Actual	0	5009	41
	1	2	72

Table 3: Training results for CNN and SVM

Model	Precision	Recall	$F_{\beta=1}$	Accuracy	AUC
CNN	0.574	1.000	0.729	0.989	0.766
SVM	0.637	0.973	0.770	0.992	0.579

For the training data, the relationship between recall and precision for different classification thresholds are presented as a precision-recall curve in figure 12.

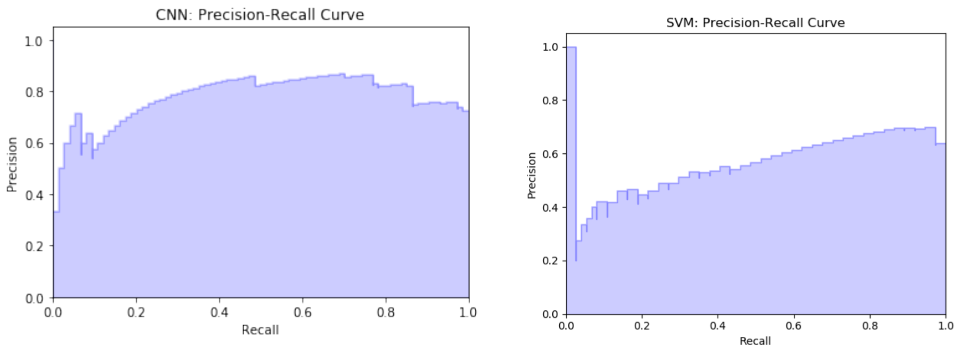


Figure 12: Precision-recall curve for CNN and SVM on the training data

4.2 Test data

The models where evaluated on a test data set with 575 observed stars (of which five are the reversed duplicates). As in previous section, the test results are presented in a confusion matrix in table 4 below and with different performance metrics in table **??**, where the AUC is the area under precision-recall curve. The precision-recall curve is also presented in figure 13.

Table 4: Confusion matrix for CNN 4a) and SVM 4b) on the test data

a)

		Predicted 0	Predicted 1
Actual	0	562	3
Actual	1	0	10

b)

		Predicted 0	Predicted 1
Actual	0	559	6
Actual	1	2	8

Table 5: Test results for CNN and SVM

Model	Precision	Recall	$F_{\beta=1}$	Accuracy	AUC
CNN	0.769	1.000	0.870	0.995	0.672
SVM	0.571	0.800	0.667	0.986	0.508

Table 6: Test results for CNN and SVM

Model	Precision	Recall	Accuracy
CNN	0.769	1.000	0.995
SVM	0.571	0.800	0.986

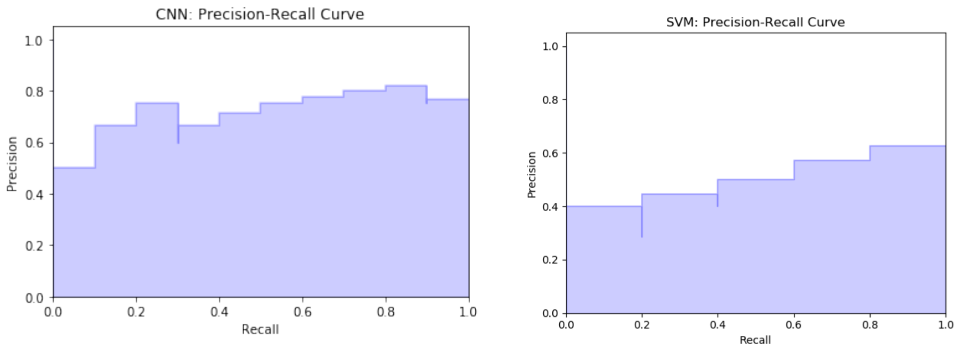


Figure 13: Precision-recall curve for CNN and SVM on the test data

4.3 Configuring the SVM

When trying out the different SVM configurations with *GridSearchCV* it was clear that the fourier transformed input data outperformed both the original time domain input data and the *catch22* feature based input. Results for F_β values from the runs on the different input data types are presented in figures 14, 15 and 16.

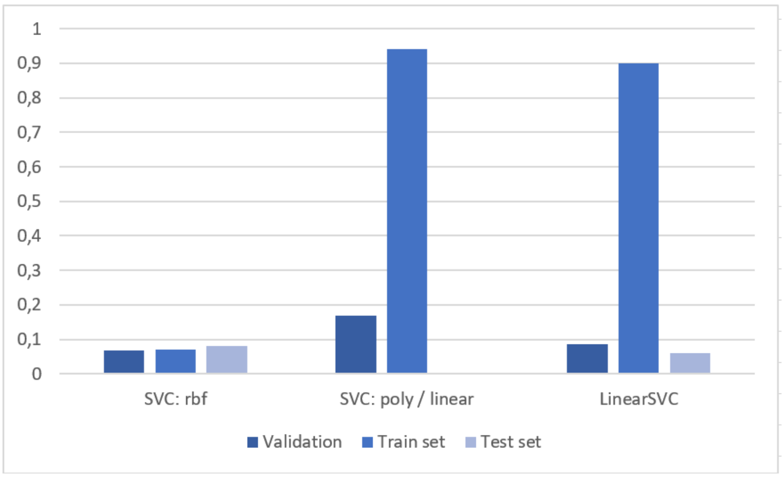


Figure 14: **Time input -** Results for the best performing configurations of each kernel trained on the original time input data. The different bars represent F_β scores ($\beta = 2$) in *GridSearchCV* validation, on the training set and on the test set . After the *GridSearchCV*, the best performning model was trained on the full training set and evaluated on the test set. (The poly/linear -kernel had the result $F_\beta = 0$ on the test data.)

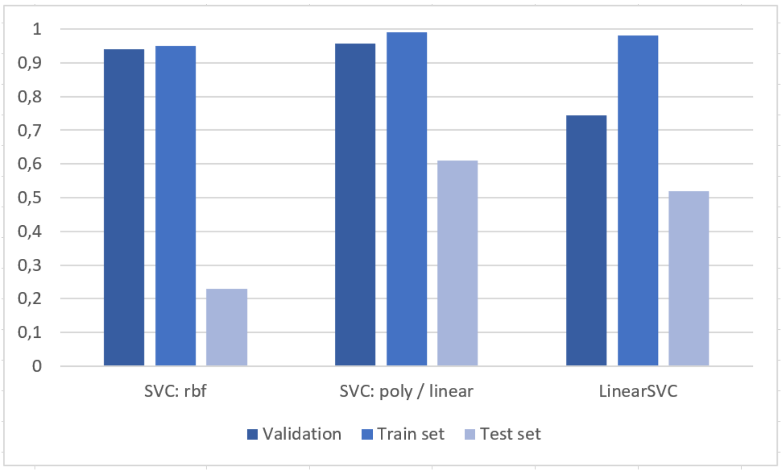


Figure 15: **Fourier input -** Results for the best performing configurations of each kernel trained on fourier transformed input data. The different bars represent F_β scores ($\beta = 2$) in *GridSearchCV* validation, on the training set and on the test set . After the *GridSearchCV*, the best performing model was trained on the full training set and evaluated on the test set. (The rbf-kernel had the result $F_\beta = 0$ on the test data.)

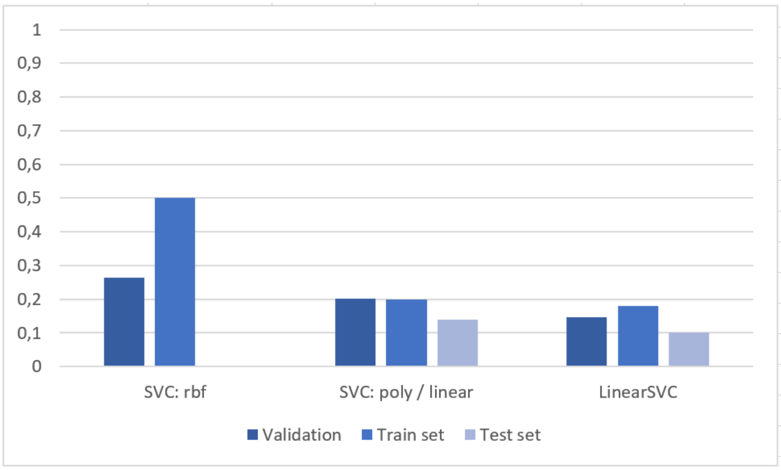


Figure 16: **Feature input -** Results for the best performing configurations of each kernel trained on feature input data. The different bars represent F_β scores ($\beta = 2$) in *GridSearchCV* validation, on the training set and on the test set. After the *GridSearchCV*, the best performning model was trained on the full training set and evaluated on the test set.

From the *GridSearchCV* results, the SVC model with a linear kernel performed best on the fourier transformed data. The model was tested further and the normalization of data turned out to impact the result, which can be seen in table 7.

Table 7: Confusion matrices for the SVC with linear kernel and fourier transformed input data. **a)** Normalized data training result **b)** Non-normalized data training result **c)** Normalized data test result **d)** Non-normalized data test result

a)

		Predicted	
		0	1
Actual	0	5050	0
	1	0	74

b)

		Predicted	
		0	1
Actual	0	5009	41
	1	2	72

c)

		Predicted	
		0	1
Actual	0	562	3
	1	4	6

d)

		Predicted	
		0	1
Actual	0	559	6
	1	2	8

The different *gamma* values used (*'auto'* and *'scale'*) did not affect the performance of the SVC-model. The tested class weights did not change the result drastically and none of them performed better than the *class_weight = 'balanced'* option available for the model.

5 Discussion

5.1 Exoplanet detecting

There is one main difficulty in measuring a transit. The exoplanet must pass directly between its star and the observer. For an observer standing on Earth this means that the exoplanet solar system must be in the same plane as Earth's solar system or for some other reason pass directly between Earth and its star. Since very few of the solar systems in the galaxy are orbiting in the same plane as ours only a very small part of the possible exoplanets will be detected with this method. Another phenomena that causes problems is the fact that a transit happens in a very short period of time compared to the entire orbit of the planet in question and the fact that the kepler campaigns only last for about 80 days (around 22 % of Earth's orbital cycle) means that there is a high risk of missing planets with longer orbits. Binary stars are also causing trouble since the method tends to misjudge binary stars as planets. However, the transit method is still by far the most superior method for detecting exoplanets [25].

5.2 Data set

The data set used in this project contains a large amount of observed non exoplanet stars compared to the amount of confirmed exoplanet stars. This means that the data set is clearly imbalanced and some problems occur consequently. This is partly prevented by mirroring the data with confirmed exoplanets and adding them to the set for both methods. The CNN method also added additional positives by rolling the samples in time to generate more positives. This procedure increases the number of exoplanet stars compared to the amount of non exoplanet stars in the data set which simplifies the prediction for the model.

Similarly the positive accuracy for both models tend to be a bit misleading since the accuracy is a measure of the total amount of correctly predicted data points. Due to the imbalance of the data set it is possible to get a high accuracy even if the model only predicts negatively for all stars.

Another consequence of the imbalanced data set is that it sometimes generates a low precision. Precision is a way to validate a model by dividing all the true positives by all predicted positives. Since the number of non exoplanets is bigger than the number of exoplanets the amount of falsely predicted positives can become a lot bigger than the amount of true positives. Consequently a small number can be divided by a large number and accordingly the precision stays lower than it could have done for a balanced data set.

Worth mentioning is that the data set used in this project was launched for the public because a group of scientists at NASA believed that all the exoplanets in this data set had already been discovered. The group of scientists working with exoplanets at NASA is called NExSS and contains top scientists from a variety of scientific fields to bring a broad expertise to the group. [26] The group who has labeled the data set and launched it to the public because they believed that all exoplanets were discovered. Although this conclusion is well-founded and made by experts in this field there is still a possibility that some data points are mislabeled [8]. By looking at light curves, it is also easy to see that the patterns characterizing the positive and negative samples are not that obvious, illustrated in figures 8 and 9.

5.3 Model performances

As seen in table 2, classification on the training set with CNN resulted in no false negatives, giving recall a value of 1.000, but 55 false positives causing a lower value in precision of 0.574. The SVM classified 41 false positives resulting in a precision value of 0.637 on the training set, which is higher than for CNN and two false negatives resulting in a recall of 0.973, which is lower than for CNN. Though solely evaluating the model on the training data can be misleading since the model is familiar with that data set and might have a high variance. Therefore the results for the test data may be more representative since the data is new to the model.

The results on the test data is shown in table 2. For the test data the SVM got a recall of 0.800 and a precision of 0.571 while the CNN got a recall of 1.000 and precision of 0.769. An explanation for the CNNs better performance is that it's designed to recognize patterns in in large sets of data. The recall score is a measurement of how well the model classifies positively labeled curves (stars with an exoplanet in orbit) while precision score measures how well the model classifies negatively labeled curves (stars with no exoplanet in orbit). This shows that since the SVM got a higher precision than recall it is better on classifying the negatively labeled curves than the positive ones while for the CNN it is contrariwise. The F_β score is the harmonic mean of the recall and precision score and with $\beta = 1$ the two scores are weighted equally. The CNN got a F_1 score of 0.870 while the SVM got a score of 0.667. The accuracy for both of the models are close to 1 though one have to remember that the accuracy score alone may not be representative when dealing with highly imbalanced data sets. The precision recall curve is a graphical representation of the precision-recall trade off and the area under precision-recall curve is a calculation of the its integral. For CNN the precision-recall curve has been helpful for choosing the classification threshold value in order to get a large precision for a recall of 1.000.

5.3.1 Features

Sadly, the feature based classification for the SVM did not perform very well, as seen in figure 16. This may be because the *catch22* feature set was not able to capture the properties which distinguishes differences of positive and negative samples, which means the method has a high bias. Indications of this can be seen in figure 16, where the models perform bad on the training data as well as on the test data and thus can not separate negative from positive samples. The 22 features used were selected by testing the *hctsa* features on 93 different time series, but it is possible that the time series in this study differ too much from them for the features to be useful. The bigger feature sets in *hctsa* or *tsfresh* might perform better. Another alternative is to use features more specifically designed for this problem, which has been relatively successful in another exoplanet study [20]. Since the fourier transformed data yielded pretty good results, frequency based features could probably be an option.

5.4 Further studies

When it comes to preprocessing of data the possibilities are endless. For this study only a few alternatives of preprocessing where investigated and maybe further work would have improved the performance of the models. Since the model is looking for dips in the light curves one possible further step in preprocessing could be removing all upper outliers since they could be irrelevant.

For the convolutional neural network further investigation of the network could have been made by removing or adding layers. When building a neural network it can be tempting to add more layers than necessary. For this thesis a quite deep network was used and further studies in removal of hidden layers could be relevant. Another idea for modification of the model would be to use recurrent neural networks, and more specifically long short term memory networks also known as LSTM networks. After a meeting with Philip Harrison, a PhD student at Uppsala University at the Department of Pharmaceutical Biosciences using machine learning for his research, who also has tried using machine learning for exoplanet hunting suggested that the LSTM networks could be suitable for this project. LSTM networks differs from traditional neural networks by remembering previous events in a way that traditional neural networks are incapable of [27].

For the SVM, bigger feature sets or self-designed features might be able to distinguish the characteristics of the postitive curves better, which *catch22* could not, and thus could be explored further. More attempts with different filters and other pre-processing might also give better results. The model configuration with *GridSearchCV* could also be done with even more parameters, to examine how they affect the results.

One thing that could also be useful is to create a model which can handle time series of different lengths as input, which neither the CNN, nor the best performing SVM can. This means that light intensity data measured over a longer/shorter time or sampled at another frequency can not be classified. To develop a model capable of this would be useful for analysis of data from different campaigns or telescopes and analyzing such data would also give better insight to how the models perform in other settings.

6 Conclusion

Two methods for finding transiting exoplanets where presented, convolutional neural networks and support vector machine. The CNN managed to classify all transiting exoplanets correctly for the test set but mislabeled a fraction of the light fluxes belonging to stars with no orbiting exoplanets. The SVM performed equally as the CNN in finding the negatively labeled curves but didn't predict the positively labeled curves as well. For all the parameters that was used for validation the CNN had a higher score than the SVM for the test data. Therefore the most well performing model on this data set containing time-series of light intensity from extrasolar stars with the chosen configurations turned out to be CNN. The reason for this result is thought to be the good pattern recognition ability of CNN.

Working with such imbalanced data sets can be hard and it is an issue that is handled multiple times. With only a fraction of positive labeled curves creativity is needed for generating more positive examples and validation of the models are needed to be done with certain measurements.

Within machine learning and designing of algorithms there is an infinite number of configurations for a certain model. Both of the methods have endless possibilities of how to tune variables and multiple choices of how to preprocess the data. With that said, from this thesis one can not make a general conclusion on which method that will perform the best on this data set, only conclusions for the investigated configurations.

Although this thesis did not present a perfect solution it is shown that impressive results can be achieved without extensive knowledge and hopefully this study can be used for further improvements within the field.

7 References

References

[1] Chen, Rick. “NASA’s First Planet Hunter, the Kepler Space Telescope: 2009-2018.” Text. NASA, October 22, 2018. http://www.nasa.gov/kepler/missiontimeline.

[2] Barentsen, Geert. “NASA - Kepler/K2 Guest Observer Program.” Kepler & K2. Accessed May 20, 2019. ./pages/k2-campaign-fields.html.

[3] Lindholm, Andreas, Niklas Wahlström, Fredrik Lindsten, och Thomas B Schön. ”Supervised Machine Learning”. Department of Information Technology - Uppsala University, 19 mars 2019, 83.

[4] “Exploring Exoplanets with Kepler Activity.” NASA/JPL Edu. Accessed April 10, 2019. https://www.jpl.nasa.gov/edu/teach/activity/exploring-exoplanets-with-kepler/.

[5] “5 Ways to Find a Planet.” Accessed April 11, 2019. https://exoplanets.nasa.gov/5-ways-to-find-a-planet/index.html#/2.

[6] “Exoplanet Exploration: Planets Beyond Our Solar System.” Exoplanet Exploration: Planets Beyond our Solar System. Accessed April 11, 2019. https://exoplanets.nasa.gov/.

[7] “K2.” STScI. Accessed April 10, 2019. http://archive1.stsci.edu/k2.

[8] “Exoplanet Hunting in Deep Space.” Accessed April 10, 2019. https://kaggle.com/keplersmachines/kepler-labelled-time-series-data.

[9] Keenan, Tyler. “Neural Networks: Machine Learning Inspired by the Brain.” Hiring | Upwork, April 10, 2017. https://www.upwork.com/hiring/data/neural-networks-demystified/.

[10] Goodfellow, Ian, Yoshua Benigo, and Aaron Courville. Deep Learning. MIT Press. Accessed May 27, 2019. https://www.deeplearningbook.org/.

[11] Kingma, Diederik P., och Jimmy Ba. ”Adam: A Method for Stochastic Optimization”. ArXiv E-Prints, december 2014, arXiv:1412.6980.

[12] “1.4. Support Vector Machines — Scikit-Learn 0.20.3 Documentation.” Accessed April 3, 2019. https://scikit-learn.org/stable/modules/svm.html#svm-kernels.

[13] “Introduction — Tsfresh Documentation.” Accessed May 27, 2019. https://tsfresh.readthedocs.io/en/latest/text/introduction.html.

[14] “In-Depth: Support Vector Machines | Python Data Science Handbook.” Accessed May 22, 2019. https://jakevdp.github.io/PythonDataScienceHandbook/05.07-support-vector-machines.html.

[15] Barber, David. Bayesian Reasoning and Machine Learning. Cambridge: Cambridge University Press, 2011. https://doi.org/10.1017/CBO9780511804779.

[16] Batuwita, Rukshan, and Vasile Palade. “Class Imbalance Learning Methods for Support Vector Machines.” In Imbalanced Learning, edited by Haibo He and Yunqian Ma, 83–99. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2013. https://doi.org/10.1002/9781118646106.ch5.

[17] Deisenroth, Marc Peter, A Aldo Faisal, and Cheng Soon Ong. “Mathematics for Machine Learning,” n.d., 421.

[18] Fulcher, B. D., and N. S. Jones. “Highly Comparative Feature-Based Time-Series Classification.” IEEE Transactions on Knowledge and Data Engineering 26, no. 12 (December 2014): 3026–37. https://doi.org/10.1109/TKDE.2014.2316504.

[19] Lubba, Carl H., Sarab S. Sethi, Philip Knaute, Simon R. Schultz, Ben D. Fulcher, and Nick S. Jones. “Catch22: CAnonical Time-Series CHaracteristics.” ArXiv:1901.10200 [Cs, Stat], January 29, 2019. http://arxiv.org/abs/1901.10200.

[20] Bugueno, Margarita, Francisco Mena, and Mauricio Araya. “Rening Exoplanet Detection Using Supervised Learning and Feature Engineering,” n.d., 10.

[21] Garza, Gabriel. “Exoplanet Hunting with Machine Learning and Kepler Data -> Recall 100

[22] “Sklearn.Metrics.Fbeta_score — Scikit-Learn 0.21.1 Documentation.” Accessed May 17, 2019. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.fbeta_score.html.

[23] Schön, Thomas. “Lecture 3 – Classification, Logistic Regression,” n.d., 27.

[24] “Sklearn.Model_selection.GridSearchCV — Scikit-Learn 0.21.1 Documentation.” Accessed May 23, 2019. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.

[25] “Transit Photometry.” Accessed April 24, 2019. http://www.planetary.org/explore/space-topics/exoplanets/transit-photometry.html.

[26] Loff, Sarah. “NASA’s NExSS Coalition to Lead Search for Life on Distant Worlds.” Text. NASA, April 21, 2015. http://www.nasa.gov/feature/nasa-s-nexss-coalition-to-lead-search-for-life-on-distant-worlds.

[27] Olah, Christopher. “Understanding LSTM Networks – Colah’s Blog.” Accessed May 24, 2019. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

8 Appendix

All code can be found in this GitHub-repository: `https://github.com/precisit/kex-exoplanet`