Flight Delay Error Analysis for Amsterdam Schiphol Airport Using a Deep Neural Network (DNN)

V. Balsingh G. Bloem M. de Keijzer L. Haagh B. Harrison-Galvez
R. Lunenburg C. Overtveld N. van der Hijden T. Vleming

Predicting flight delays has become an important method to reduce airline economical losses and to keep costumers satisfied. In this research, a model is developed to predict errors in delay estimations using a deep learning neural network, since airlines cannot accurately predict flight delays. This model predicts errors in flight delays for flights both arriving at and departing from Schiphol Airport. For these predictions, flight data from Schiphol Airport covering six months in the years 2012/2013 is used. From this data, relevant features are selected (airline, time between message and expected arrival or departure, and arrival or departure country). Then, a model with the best predictive capabilities possible, relative to other trialled models, is trained and validated. The created model predicts the delay errors with an accuracy (defined as coefficient of determination or R^2) of 0.64 for arriving flights and an accuracy of 0.95 for departing flights. This means delay errors for departing flights are better predictable than for arriving flights, with the latter having an accuracy comparable to non-automated processes. This model can then be used by Schiphol Airport by creating an app for the customer to increase airline customer experience.

Key words: aircraft delay, machine learning, deep neural network, ReLU, Adam algorithm

I. Introduction

Over the past couple of years, the number of people travelling all around the world has increased drastically. The main form of transport is flight [1]. Due to the increase in aircraft arriving at and departing from airports every day, airports have become busier than ever, hence the possibility of a flight being delayed is highly present. These delays have a large economical impact and result in negative passenger experience. The cost of these delays for all flights in the USA in 2007 was 31.2 billion USD [2]. In the Netherlands, these delays have a large impact on Schiphol Airport, which transported 70 million people in 2016 [3]. Therefore, it is important to reduce flight delays to a minimum, in order not to lose money and keep the passengers satisfied.

To reduce flight delays, errors in flight time updates have to be predicted. Although each airline makes their own delay predictions, these are not always accurate [4]. For this reason, making a model to predict errors in delays is essential since it will help to improve the models used by airlines, which will also improve passenger experience. Even when the airline's prediction is given early in the flight, this model will be able to predict the error of the arrival time already. This is beneficial for Schiphol Airport, since Schiphol can now act in advance and coordinate gate availability and maintenance. Passengers will benefit by having more accurate information about their flights. For these predicting models, previous researches have been conducted. However, these researches have not led to significant improvements in predicting flight delays.

For example, Meyn has tried applying probabilistic models [5]. Mueller and Chatterji have made use of standard distributions to compute the probability of delays [4]. Tu, Ball and Jank take yet another approach using the Expectation-Maximisation algorithm whereas Ding uses a more concise method, namely multiple linear regression [6][7]. Finally, there are multiple researchers who have tried applying machine learning to this problem before. Based on the aforementioned researches, it can be concluded that a neural network leads to the best accuracy on complex data. Furthermore, it is easy for the user to apply the model after it has been trained.

A neural network is an application of machine learning which learns to recognise patterns in a data set. These patterns are used to build a predicting model. The neural network is made up of different layers, each layer contains weights which are automatically adjusted according to the recognised patterns [8].

This knowledge gap leads to the following research question: how can flight delay estimates be used to improve customer flight experience at Schiphol Airport? The goal of this research is to create a neural network model to predict and analyse flight delay errors for Schiphol Airport, with an accuracy (R^2) in the range of 0.50 to 0.65. This is an expected range for non-automated processes [9].

The following structure is used for this paper. Firstly, this paper explains the data handling methods. Secondly, in the method section, the working principle of neural networks, the method of training, the performance criteria and optimisation methods are discussed. Thirdly, the results are shown in the subsequent section. Fourthly, the execution of the method and the results is reflected upon in the discussion. Finally, a conclusion is drawn, and future recommendations are given.

II. Data input

The model developed in this research is based on data from Flight Information Royal Dutch Airlines (FIRDA). This is a system which records messages transmitted between airlines and airports. The data received from FIRDA consists of delay predictions, made by the arriving or departing aircraft. From two periods of three months, data is obtained from this system and collected in one file [10]. The raw data is shown in Table 1.

Table 1: Raw data (first five data points)

OrgFltDate	SDep	BDep	SArr	BArr	Dep	Arr	MsgTime	NewMsg	FltNbr	NewTime
23NOV2012	06:15	06:21	07:30	07:38	CDG	AMS	22NOV12 16:50	C05	UA 1816	-
23NOV2012	06:15	06:21	07:30	07:38	CDG	AMS	23NOV12 06:20	CDQWS	UA 1816	-
23NOV2012	06:15	06:21	07:30	07:38	CDG	AMS	23NOV12 06:22	0822C	UA 1816	07:22
23NOV2012	06:15	06:21	07:30	07:38	CDG	AMS	23NOV12 06:33	0823C	UA 1816	07:23
23NOV2012	06:15	06:21	07:30	07:38	CDG	AMS	23NOV12 06:56	0828E	UA 1816	07:28

The data consists of multiple message lines per flight. Each line is treated by the neural network as a separate data point. To clarify the variables that are used in the table; 'BDep/BArr' is the actual departure/arrival time, as opposed to the scheduled time ('SDep' and 'SArr'). This means that the difference between the scheduled and the block time is the delay time. 'Dep' and 'Arr' are the departure and arrival airport, respectively. Next to that, 'MsgTime' is the date and moment in time when the delay prediction message was sent. Lastly, 'NewTime' is the new prediction of departure/arrival time.

First of all, the data set is cleaned up. A small number of these lines miss data for the actual time of departure and/or the actual time of arrival. For arriving flights data points without arrival time are excluded from the data set, and vice versa for departing flights. These data points have no value for the neural network, since they do not provide a reference time from which the neural network can learn.

Then, the variables in each data point are split into either categorical or continuous variables. Categorising data allows adding non-continuous and non-numerical data, such as 'airline', to a neural network, by translating it to discrete numerical data [11]. Doing this helps the neural network recognise patterns more easily, allowing variables such as the time difference between message and arrival/departure to be added as input explicitly [8].

This represents the initial data input for the model. However, it is found that separating the arrival and departure data yields better accuracy in predictions in error for each; this is explained in further detail in section III, subsection C.

III. Method

In order to predict the aircraft delay error, a deep neural network is used. This network is built using the data mentioned in the previous section. This section provides a detailed view on the model, showing how it works and explaining why this specific type of network was chosen. The optimisation process and performance criteria are described as well.

A. Network classification

In this research, a deep neural network is built in order to predict the difference between the final delay and the delay estimated by the pilot. It is built using Keras, a deep learning toolkit for Python which is based on the TensorFlow library [12][13].

A deep neural network is a neural network with more than one hidden layer. Apart from hidden layers, neural networks also make use of an input and output layer. The input layer takes the standardised feature columns from the processed data frame, and the output layer gives a predicted error on the expected delay. The hidden layers, each consisting of a number of nodes, generate the output.

The neural network built in this research is a feed-forward neural network. Feed-forward is a type of neural network in which the data is not recycled in-between hidden layers, like in a recurrent neural network. A recurrent neural network has the ability to fulfil more complex tasks, such as speech analysis, using fewer hidden layers [8][14]. This comes at the cost of higher required computational power. Since the given flight data is not as complex as speech, a feed-forward type neural network will suffice.

B. Network run-through

In this section, the full neural network is explained. The section is structured in the same way that the data flows through the neural network. All six steps are handled chronologically and the used functions are explained. Even though two different networks are built, one for departure and one for arrival, both of them have the same structure and therefore the explanation below is applicable for both.

Firstly, the neural network reads the data mentioned in the data input section. In order for the data to be used by the program, the input data has to be standardised first. Standardising the data is a mean to decrease the magnitudes of the values in the input data by converting the data using a standard distribution. Using standardised data will improve the neural network performance, since the program will have to handle a smaller range of numbers [15].

Secondly, the data is split into a training and testing set. The training set contains the data points that are used in the construction of the model. This data set uses 70 % of the total standardised input data. The remaining 30 % makes up the testing set, as this is a commonly used division [8]. The testing set is used to find the accuracy of the model that is build from the training data. Since the input data is chronological, the data points are randomly assigned to the sets. This is done in order to prevent the influence of unseen patterns with respect to time. Having for example all the winter months in either the training or testing set, may result in a less accurate model.

Thirdly, the model itself is built. Construction of the model requires the setting of multiple parameters. First of all, the layers have to be set. For each layer, the number of nodes and activation function are determined. The purpose of an activation function is to determine whether a given node in a hidden layer is activated or outputted, and what that output value is. This is determined using the input to that node, in a feed-forward network this input is the sum of all weighted input data with corresponding biases. In simple terms the activation function can be seen as a way in which nodes communicate with one another. Keras allows for the use of multiple activation functions, out of which the rectified linear unit activation function, or ReLU, was selected for the model [16]. ReLU is currently the leading activation function in the world and is the most commonly used activation function for deep neural networks [17].

Next, two more parameters need to be set when building the model: the loss function and the optimiser. The loss function is required to enable the neural network to make accurate predictions. The loss is a measure that shows the difference between the prediction of the model and the actual results of the training set. Decreasing the loss of the network will result in more accurate predictions later on. Keras has multiple loss functions to choose from [18]. The program mentioned in this paper uses the mean squared error, or MSE, in order to train the model. The mean error is the difference between the actual error and the predicted error in arrival or departure time by the model. The square of this is the mean squared error. The mathematical representation is given in Equation 1, used in a neural network as the loss function.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{1}$$

For the reduction of the loss of a neural network, an optimisation algorithm is required. Keras has several options for the optimisation algorithms [19], of which multiple are tried in order to find the optimiser that fits

the model best. This turns out to be adaptive momentum estimation, or Adam. Adam is a form of gradient descent, which means that it adapts the model coefficients, used in the hidden layer(s) along a gradient towards a stage of minimum error. An explanation for how gradient descent optimisation techniques are used is presented by Ruder [20].

Essentially, gradient descent aims to minimise the error in a linear regression line fit to the data. By adjusting the weight and bias associated to said linear fit in a number of iterations, the error is decreased [8]. This is a commonly used method for machine learning [21]. The Adam algorithm has been shown to perform better than any other optimisation algorithm when deep neural networks are considered [22][23].

In the fourth step, the model is set up in order for it to run the data. In this step, three parameters are set: the number of epochs, the batch size and the validation split. An epoch is one run through the training data set. An extra line of code is added in order to stop the model from running if the loss stops decreasing. This will benefit the running speed, since no unnecessary epochs are run. The batch size is the number of data points that is taken into the program before updating the weights of the model. For the model used in this paper, the mini-batch approach is used. This method only uses a small part of the data set which decreases processing time but is still accurate, compared to using the entire data set for training [21].

Validation sets are taken from the training set during the training of the model in order to measure how the model performs on new data. It is a method commonly used to quickly measure the over-fitting of the model and ensure generalisation. It is essential to make sure that no over-fitting occurs when designing the model. Over-fitting occurs when the model resembles the training set too much. As a result, the accuracy when testing the training set is extremely high, but the testing set is not modelled accurately at all. Therefore, the trade-off is between a model that closely resembles the training set, and also gives the most accurate general result when tested on the validation set [24]. The validation split used for the training set is 20 % for this model.

For the fifth step, the model is run, saving the results from each epoch for later examination. The training and validation loss are plotted after running the model to visualise if over-fitting has occurred. A separate plot is made showing the validation mean absolute error, or MAE, in order to pick the number of epochs used for later predictions. The epoch at which the MAE is smallest is used for making the predictions.

Finally, the prediction is made. The model built in the previous steps is used to estimate the error made by the pilot for the testing data set. After these predictions are made, the real prediction errors are compared to the estimated errors found by the model in order to determine the coefficient of determination, or R^2-value. The R^2-value is the most important indicator for accuracy of the neural network since it shows how well the model fits the data. The mathematical representation is given in Equation 2. The R^2-value is always between 0 and 1, where 0 resembles a bad fit and 1 a good fit; and with this the accuracy of the neural network is evaluated.

$$R^2 = 1 - \frac{\sum_i (Y_i - f_i)^2}{\sum_i (Y_i - \hat{Y}_i)^2} \tag{2}$$

C. Optimisation

After the initial model is built, it has to be optimised to achieve a sufficient accuracy. In this optimisation process many of the aforementioned parameters are altered in an iterative process in order to create the optimal flight delay prediction error model for Schiphol airport. These parameters include the number of layers, the nodes per layer and the batch size.

Apart from the iteration of the parameters mentioned, several methods are used in an attempt to improve the performance of the neural network. The first of these methods is the use of data splitting. For the neural network, the data is split into departing and arriving flights. This is done to create two separate neural networks, one for the departing and one for the arriving flights. Two separate neural networks have a better expected accuracy than a single neural network, since the data output (error) for departure and arrival differ greatly. Due to this difference, one single model is not able to predict both arrival and departure as accurately as two separate models. In practice, splitting the data increases the accuracy of the neural network and therefore it is decided to keep this change.

The features used by the neural network have a significant impact on the model performance. The features used by both the arrival and departure model are: the airline, the scheduled departure time, the

scheduled arrival time and the time left until departure/arrival. Apart from these features, the arrival model uses the actual departure time and the departure model uses the arriving country name. The reason that the departure model does not use block departure time is that for real life applications the block departure time will be unknown, since the plane has not yet departed at the time the prediction is made. Finally, the desired output for both models is the delay estimation error. This is also imported from the data into the neural network, but no weights will be assigned to this parameter. The delay estimation error provided is used to check the loss of the model.

Improving a given model not only entails increasing its accuracy, but also ensuring that the model is able to perform well on new data; this is called regularisation. This is seen by use of a validation set, a subset of the training data used in the training of the model. When a large amount of data is present it may be sufficient to simply split a percentage of the training data before the model is run. K-fold cross-validation is a method most commonly used when there is little data to work with in order to avoid over-fitting of the model. This is because it partitions the training into a number of smaller data sets called 'folds', each of these folds is used as validation once during training so that the validation data changes with each epoch and the results are averaged to generate one estimation [25]. As discussed in section IV, there is very little over-fitting seen in both models, evident by the constant negative gradient of the validation loss, as the loss of the validation set continues to decrease with an increased number of epochs. Therefore, the use of K-fold cross-validation does not lead to an improvement of the neural network and it is decided not to use K-fold in the final model.

Finally, more direct regularisation techniques have been explored, similar to the use of cross-validation. These techniques prevent over-fitting to the training set of the model. The difference lies in the manipulation of the training input itself in each layer of the network, rather than splitting the training data and selecting multiple validation sets to be used in training. One such manipulation is termed weight regularisation, whereby large weights are punished by the loss function. Another method used was dropout, which zeros a certain percentage of the data input to each layer forcing the network to work with a smaller amount of data and therefore not form strong patterns seen only in that data. Both techniques have been tried, however, with a model that does not appear to be over-fitting drastically, the drop in accuracy does not outweigh the gain in regularisation. Therefore, these techniques are disregarded for the final run.

IV. Results

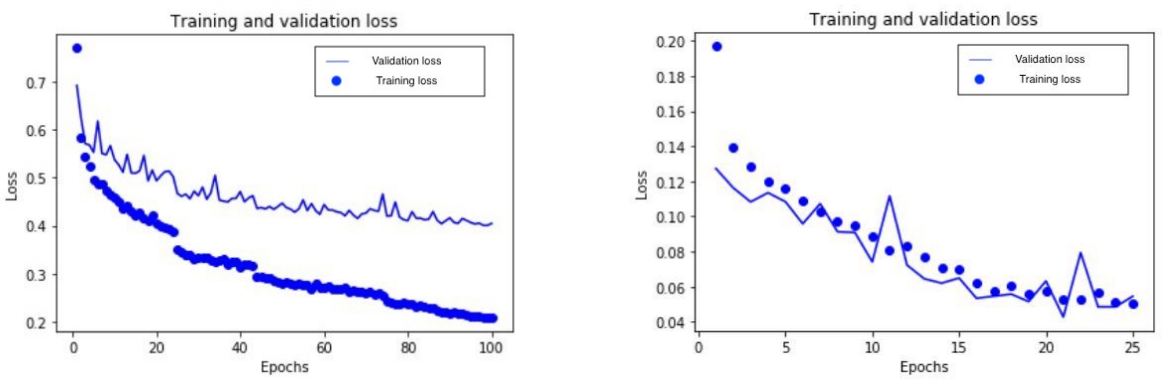
In this section, a description is given on what results are achieved from the aforementioned method. The characteristics of both the arriving flights and departing flights model are presented in Table 2. The layers presented in this table are the hidden layers constructed for the network. The input layers for both models are structured the same, as the input data shape is unchanged for each model. Therefore, the output also has the same shape.

Table 2: Results

Parameter	**Arrival**	**Departure**
Architecture		
Number of layers	3	3
Number of nodes	Layer 1: 256 Layer 2: 256 Layer 3: 128	Layer 1: 128 Layer 2: 128 Layer 3: 256
Batch size	128	128
Optimiser function used	Adam	Adam
Use of clusters	yes (arrival and departure)	yes (arrival and departure)
Features used (input shape)	6	6
Performance		
R^2 (accuracy)	0.64	0.96
MSE (loss)	0.43	0.05

Table 2 shows a difference only in the number of nodes per layer of the model, aside from this and the data input the models performed best with identical architectures.

Figure 1a and Figure 1b display the number of epochs with the corresponding losses. Each time one epoch is added until the validation loss (continuous line) plateaus. Once the line plateaus, a loss minimum is reached and the amount of epochs is chosen at this minimum.



(a) Arriving flights.

(b) Departing flights.

Figure 1: Loss per epoch for training and validation data sets for arriving/departing flights.

It is evident that the validation loss plateaus and separates from the training loss earlier for the arriving flights than for the departing flights. This is analogous of over-fitting of the model, i.e. the arriving flight model does not perform as well on new data for a given number of epochs while the departing flight does not excibit this problem.

The results achieved after running the arrival model show that after 85 epochs, no better prediction can be made on new, unseen data. Setting this epoch number for the final training of the model yields an R^2 of 0.64 and a loss (mean squared error) of 0.43. Similarly Figure 1b shows prediction stops improving after only 24 epochs; yielding an R^2 of 0.96 and loss of just 0.05. Evidently, the departure flight delay error can be far better predicted when compared to the arriving flight delay error.

V. Discussion

Here, the aforementioned results are evaluated on their reliability and usability. Firstly, it can be observed that the accuracy of the departing flights is a lot higher than the accuracy of the arriving flights. This can be explained by the fact that the time until departure is usually shorter than the time until arrival. Therefore it is easier to predict at what time the plane departs than to predict at what time the plane arrives.

Secondly, the departing planes are still on the ground when the prediction is made. In that case, the plane is not as prone to weather changes or other, random or environmental causes of delay.

Thirdly, the operations done on the ground are more automated. This ensures there is a smaller variation in the data due to the consistency of machines. If humans would do the same work, the delay could be dependable on the employee doing the job. Therefore the difference between the departing and arriving accuracy are expected.

As mentioned before, a reliable R^2 for non-automated processes lies between 0.50 and 0.65. As the arriving flights are automated to a lesser extent, the value of 0.64 can be considered reliable. As opposed to arrivals, departures may be more automated; additionally there are not as many exterior influences which can decrease accuracy of the results. This could therefore account for the higher R^2 value of 0.96 which, together with the high performance of the validation data, can be considered as a reliable result.

Finally, a flaw in the data handling is the fact that flights which take place overnight at the end of a month, are disregarded due to difficulties in computing time differences. As the number of these flights is very low compared to the total number of data points, the impact on the model accuracy is assumed negligible.

VI. Conclusion

In this research, a neural network to predict the errors in flight delay is made, as the impact of delay on society is large. This is done using a data set with flights arriving and departing at Schiphol Airport in the period of six months. This data set is cleaned by adding and removing features to increase the accuracy of the model. The data set is split into a training and testing set and trained using a feed-forward deep neural network. This first model is then optimised by increasing the number of hidden layers. These optimised neural networks can predict delay errors with an accuracy (R^2) of 0.96 for departing flights and an accuracy of 0.64 for arriving flights.

The research goal has been met with a higher then expected accuracy for the departing flights, and an expected accuracy for arriving flights.

For future applications, a number of changes and additions can be made to the program in order to increase accuracy. One of them is to include more relevant variables to the data set, for example weather data at Schiphol Airport. Another means to improve accuracy is to use a larger data set from a longer time frame. To make the model predictions more useful in the short term, instantaneous delay predictions could be made, using the data from FIRDA directly.

After applying these recommendations, the model can be used in real life by Schiphol airport, but also for other airports by changing the input data. Adding live data to the model makes the predictions instantaneous, making it more useful for airports to use operationally. Using the model, aircraft delays can be predicted a lot more accurately than the predictions that are currently used. This can decrease costs rising from delays for airports and airlines.

The most optimal way to implement the model in practice is with the use of an app as this can be made easily accessible to all travellers. This app will be user-friendly, since the user only needs to insert their flight number in order to get an accurate prediction of the departure/arrival delay. In this way, airline customers will be able to know their delay in advance with little to no worry in the time changing again. Since the customers can now act accordingly to the delay, their airline experience is improved drastically with help from machine learning.

Acknowledgements

The authors of this article thank Dr.ir. Bruno Lopes dos Santos for his help and guidance, and Dr Jesper Verhoef for his feedback on this article.

References

[1] Dobruszkes, F., "High-speed rail and air transport competition in Western Europe: A supply-oriented perspective," *Transport policy*, Vol. 18, No. 6, 2011, pp. 870–879.

[2] Ball, M. et al., "Total delay impact study: a comprehensive assessment of the costs and impacts of flight delay in the United States," *University of California, Berkeley. Institute of Transportation Studies*, 2010.

[3] Royal Schiphol Group, "Facts and Figures 2016," *Schiphol Group*, Jan 2017.

[4] Mueller, E. and Chatterji, G., "Analysis of aircraft arrival and departure delay characteristics," *AIAA's Aircraft Technology, Integration, and Operations (ATIO) 2002 Technical Forum*, 2002, pp. 58–66.

[5] Meyn, L., "Probabilistic Methods for Air Traffic Demand Forecasting," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, May 2002.

[6] Tu, Y., Ball, M. O., and Jank, W. S., "Estimating flight departure delay distributions A statistical approach with long-term trend and short-term pattern," *Journal of the American Statistical Association*, Vol. 103, No. 481, 2008, pp. 112–125.

[7] Ding, Y., "Predicting flight delay based on multiple linear regression," *IOP Conference Series: Earth and Environmental Science*, Vol. 81, IOP Publishing, 2017, p. 012198.

[8] Mueller, A. C. and Guido, S., *Introduction to Machine Learning with Python*, OReilly Media, 2016.

[9] Wooldridge, J. M., *Introductory Econometrics: A modern approach*, South-Western Cengage Learning, 2009.

[10] Santos, B. F., "Theme 4 Project: Test, Analysis and Simulation," Project manual, TU Delft, Delft, The Netherlands, 2018.

[11] TensorFlow, "Feature Columns," `https://www.tensorflow.org/get_started/feature_columns`, Apr 2018.

[12] Chollet, F. et al., "Keras," `https://keras.io`, 2015.

[13] Dean, J., Viegas, F., and Abadi, M., "TensorFlow," `https://www.tensorflow.org/`, Mar 2018.

[14] Sak, H., Senior, A., and Beaufays, F., "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," *Fifteenth annual conference of the international speech communication association*, 2014.

[15] Shanker, M., Hu, M., and Hung, M., "Effect of data standardization on neural network training," *Omega*, Vol. 24, 1996.

[16] Keras-Documentation, "Activations," `https://keras.io/activations/`, 2018.

[17] Sharma, S., "Activation Functions: Neural Networks Towards Data Science," `https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6`, Sep 2017.

[18] Keras-Documentation, "Losses," `https://keras.io/losses/`, 2018.

[19] Keras-Documentation, "Optimizers," `https://keras.io/optimizers/`, 2018.

[20] Ruder, S., "An overview of gradient descent optimization algorithms," *ArXiv e-prints*, 2016.

[21] Brownlee, J., "A Gentle Introduction to Mini-Batch Gradient Descent and How to Configure Batch Size, https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/," Jul 2017.

[22] Walia, A. S., "Types of Optimization Algorithms used in Neural Networks and Ways to Optimize Gradient Descent," Jun 2017.

[23] Kingma, D. P. and Ba, J. L., "Adam: A Method for Stochastic Optimization," *3rd International Conference for Learning Representations*, San Diego, California, 2015.

[24] Hawkins, D. M., "The Problem of Overfitting," *ChemInform*, Vol. 35, No. 19, Nov 2004.

[25] Francois, C., *Deep learning with Python*, Manning Publications Co., 2018.