Speculating Success:
An Experimental Study on Improving Neural Network Efficiency

Word Count: 5,379

ABSTRACT

A neural network is a program that inputs a series of input variables to predict an output variable. It is commonly used for image recognition as well as other general pattern recognition tasks in numerical data. While the use of neural networks continue to increase in academic papers, many underutilize the development of new techniques that could increase the efficiency of the neural network. This paper attempts to determine which techniques can be useful in increasing the efficiency of a neural network using a dataset of students in Portuguese secondary schools. Using an experimental methodology, the paper finds Dropout and the Adam optimizer to increase the accuracy in the neural network's predictions and doesn't find PCA to decrease the training time of the neural network. Testing 8 different variations, one neural network was able to operate at a consistent 70% accuracy which was an improvement compared to other prominent papers in the field.

INTRODUCTION

Due to the recent growth in the field of big data, data mining—the study of finding trends in big data—is becoming more prevalent. One commonly used and effective data mining model is a neural network, a computer program designed to predict an output variable using other input variables. In a neural network there are hidden layers between the input and output with an adjustable number of nodes that all have specific "weights". These weights are "trained," or modified to produce the most accurate predictions for the corresponding input.

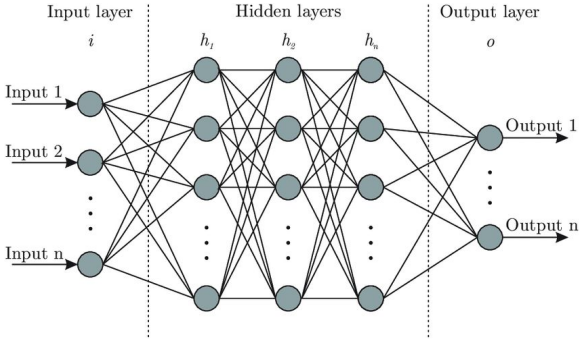

Figure 1: Visual representation of a Neural Network
Source: [1]

The data is split into a training set and a test set, where the weights are trained according to the training set, and then the ability to generalize that training to the test set is tested. Using a mathematical "loss" function that penalizes incorrect predictions and rewards correct predictions, the weights are changed using a group of algorithms that perform gradient descent. Gradient descent is the mathematical calculation that looks at the loss function and the error between the prediction of the neural network and the actual value and what modifications to the weights would decrease the loss value the most. Gradient descent is able to change the weights to decrease the loss function at the highest rate. Gradient descent has many different specific algorithms that can be modified to change the way the neural network computes the weights. Each step of gradient descent is called an epoch. After many epochs, the neural network has completed its training and accurately predicts a value corresponding to the input data.

The accuracy of a neural network is the percentage of correct outputs the neural network computes. The training time of a neural network is the time it takes for a neural network to operate at its maximum accuracy. These two metrics contribute to the efficiency of a neural network, where a higher accuracy and lower training time is more desirable.

The accuracy and training times mostly depend on the quality and size of the dataset, but techniques such as condensing the dataset or using different algorithms to tune the neural network can be introduced to improve these measures of efficiency. These techniques have been proven to be effective in increasing the efficiency of neural networks, and the use of many of these techniques have been integrated into many modern AI programming systems. These techniques can be implemented before training, modifying the structure of the data, or during training, modifying the way the neural network acts to make the most efficient neural network.

Since neural networks are currently utilized in many fields, such as predicting stocks or reading handwritten letters, it is strongly desirable to improve the efficiency of these systems. In the field of education, neural networks are therefore able to find potential at-risk students as they can use data on these students to predict the final grade they will most likely receive. While this has been tested and validated, researchers in this field have mostly neglected the techniques that can be used to make their system more accurate and quick at training. Seeing the desirability of improving neural network efficiency and the lack of improvements in the field, this paper aims to fill the gap of improving the accuracy and training times of neural networks for

predicting student performance using various techniques that haven't been considered for this purpose in combination.

LITERATURE REVIEW

Ever since the early development of neural networks in the 1940s, these complex structures have been utilized for predicting anything from objects in an image to the value of a certain stock in five years. Neural networks are a classification method in the field of data mining, and have been used in prediction tasks in fields outside of computer science. Currently, many techniques have been developed to increase the efficiency of neural networks for predictive tasks, however, these haven't been tested in combination. In this study, student performance will be used as the prediction task.

Approach

Shaleena and Paul, researchers at the Jyothi Engineering College, examined the potential methodologies around using data mining to predict student performance. They conclude that data mining can provide the tools to conduct the prediction task if an appropriate approach is used. The approach they recommend consists of first "cleaning" the data to decrease statistical noise and clutter. This may involve decreasing the dimensionality of the data, removing outliers, or processing the data using an algorithm. The process of modifying the data, called feature engineering, is done to simplify the prediction task for the neural network. Often, feature engineering is not implemented in computer science research papers for predicting student performance. The researchers then recommend implementing a data mining method, such as neural networks, to find trends within the data to make accurate predictions [2].

Shahiri et al. in their meta-analysis reviewed 13 years of literature on the use of five data mining methods (decision trees, naive bayes classifiers, k-nearest neighbor classifiers, and support vector machines, and neural networks) for the purpose of predicting student performance. A decisions tree is a linear data mining method that determines a prediction based on nested conditions. As a result, it has trouble dealing with complex classification tasks as they rely on features specifically defining linear relationships between variables and did not work well for this classification task. A naive bayes classifier is a classifier that assumes independence between features to predict the likelihood of a prediction being true. Once again, the simplicity of the prediction isn't well suited for the complex classification for predicting student performance. A k-nearest neighbor classifier groups similar data points and extrapolates data points that are put into the classifier. Since k-nearest neighbour classifiers don't perform generalizations as they find similar data points when making predictions rather than creating general assumptions of the dataset, the classifier may work well for a dataset but cannot be used effectively for a new batch of data. Support Vector Machines aim to predict variables by grouping points with the same output together and separating different points. Support Vector Machines are able to work well with small datasets but not with large ones, which is a major drawback when many students need to be examined for the predictive task. Since these data mining methods proved to be unfit for this complex predictive task, neural networks were found to be most effective due to the non-linear nature of the method and its ability to handle many inputs [3].

Studies on Predicting Student Performance using Neural Networks

To this day, there has been a large array of studies that attempted to predict student performance through the use of neural networks, using purely grades. Tekin uses a small sample of 127 students, using 49 grades from previous years, to predict student performance. He was able to achieve a high accuracy of 85% using a neural network even without using the background features of the students [4]. The large accuracy can be attributed to Tekin's use of data management, as recommended by Shaleena and Paul, but it could also be attributed to an overfitted model due to the smaller sample size as the author didn't include the validation set results.

Lykourentzou et al. used a dataset of 57 students to predict multiple choice scores without examining background features such as age and sex. Their validation error and test error varied greatly, meaning that the neural network overfit the data. Overfitting is when a neural network is able to accurately predict the training set of data, but is unable to make those generalizations to the test set. Furthermore, the researchers don't state an exact number for their accuracy, but claim that their predictions were off in their results, meaning that their lack of background features, small dataset size, and lack of data preprocessing lead to their overfit and inaccurate neural network [5].

Osmanbegovic and Suljic compared a Naives Bayes classifier, J48 algorithm, and a neural network to predict student performance. Using some background features for the 257 students they examined and proper data preprocessing, they were able to achieve an accuracy of 71% for a two level classification (predicting either pass or fail), which is not extremely high. The low accuracy can be the result of a small dataset and similar to all other studies that have been conducted around this research topic, lacks the implementation of various neural network techniques [6].

Silva and Cortez were able to fill a lot of gaps other researchers had left out, which allowed them to create the most proficient neural network compared to the mentioned research. Using a larger sample compared to others (1000+ students), the researchers compared neural networks, decision trees, and support vector machines to predict student performance. They included many background features such as previous grades, socioeconomic features, and school activities, which left out many of the previously mentioned researchers. With their neural network, they were able to achieve 88.3% accuracy for a two level classification (pass/fail) and 60.3% for a five level classification (A-F) primarily due to their large sample and many features [3], [7].

Techniques

There are three techniques that have recently been developed to increase the efficiency of neural networks. These aim to increase accuracy and decrease training time during feature engineering and the training phase.

Dropout

As neural networks have become more commonplace in data mining, researchers have been developing techniques to improve the accuracy and training times of these programs. Srivastava et al. developed a method called dropout which is implemented during the training of a neural network, where in every forward pass of the neural network, every neuron in the input and hidden layers has a set probability to be deactivated for that pass, ensuring all neurons are fully trained and there isn't an overreliance on specific neurons. This technique also helps prevent overfitting. While the implementation of this technique increases training time as it increases the amount of passes required to maximize accuracy, when this technique was used on various neural networks and datasets, the errors of these networks halved on average, showing significant improvements in the accuracies [8].

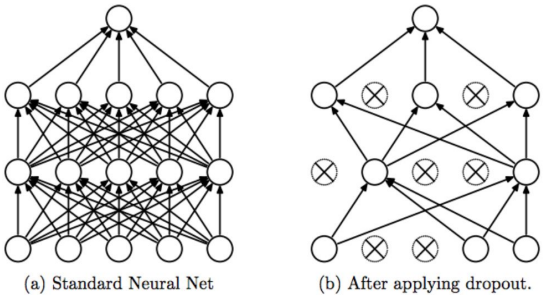

Figure 2: A visual representation of the Dropout Technique
Source: Adapted from [8]

A clear indication of overfitting can be found in the loss graphs. The loss for the training set and validation set are measured each epoch and plotted on the same axes. The neural network aims to decrease the training loss, so if the validation loss decreases too, it means that the neural network is correctly able to generalize the trends to accurately predict all parts of the data.

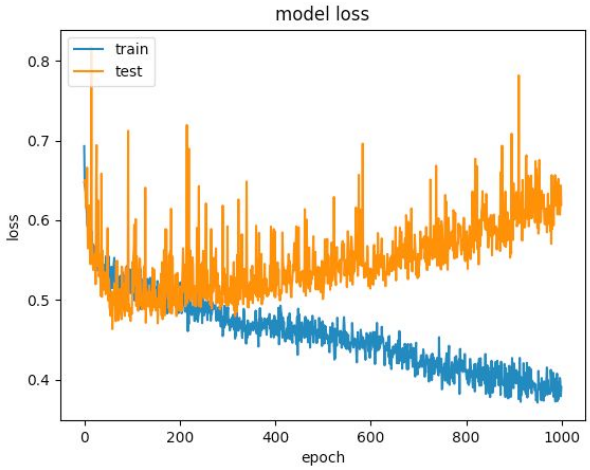

Figure 3. An overfitting sample Neural Network
Source: [9]
Notice the validation (test) loss increasing while the training loss decreases

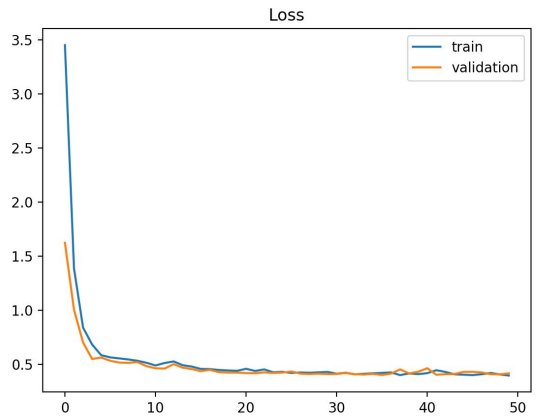

Figure 4. An optimal sample Neural Network
Source: [10]
Notice both losses decreasing and stagnating at similar values

PCA

Dr. Jason Zhang, a Senior Threat Researcher at Sophos, implemented Principal Component Analysis (PCA) into a neural network that determined whether or not a virus was present in a PDF document. PCA is a method of feature engineering that serves to decrease the amount of columns of a dataset while keeping X% of the variance in the data. When data modified by PCA is used for training in a neural network, there may be a small decrease in accuracy due to the decreased amount of input features, however, it has been proven to improve training time significantly for the same reason.

Zhang concluded that PCA decreased the training time of the neural network by 22%, but kept accuracy almost constant (<2% difference). The goal of PCA is to "clean" data, reducing the dimensions without losing the trends in the data, and proves to be an important part of data mining, as outlined in Shaleena and Paul's analysis of implementing data mining techniques [11].

Gradient Descent

Ruder's metaanalysis of various gradient descent methods also reflects the increase in techniques that can improve the accuracy and training times of neural networks. Ruder explains that by using an appropriate learning rate, mini-batch gradient descent (using a small amount of data rows for every forward pass) is the best method of passing data.

Stochastic Gradient Descent (SGD) is a gradient descent variant that maintains a static learning rate and manipulates one parameter of the neural network at a time. Randomness is introduced when parameters are updated on each pass, resulting in a relatively quick training period, but due to the randomness, this method generates mixed results, making it somewhat unreliable.

The Adam optimizer is a gradient descent variant that computes learning rates for every parameter on each pass. It has been proved to be more efficient and effective due to this adaptive methodology, and for the past few years has held onto the title of the most effective and efficient gradient descent method. The adaptiveness of the Adam optimizer can be attributed to the use of momentum, which encourages the weights to change at a faster rate towards the beginning of the training process, but slow down towards the end to speed up the training process while producing optimal results. While there have been new developments in gradient descent technology with new methods such as AdaBound, these developments have not been tested sufficiently to be considered consistently superior.[1]

Adam is the best optimizer for gradient descent as it incorporates ideas of momentum and other algorithms. Lastly, Ruder also claims that other techniques such as early stopping (where the training process is stopped early to prevent the neural network from overfitting) and batch normalization (where all the data is configured to the same scale) can be used to improve the efficiency and effectiveness of the gradient descent process [12].

Research Question and Gap

A common thread that can be drawn from all of these past papers is the lack of implementation of data cleaning, specific techniques such as dropout, and improved gradient descent methods and optimizers, which brings about uncertainties in the full potential of these neural networks. Seeing this, the new question arises: "How can neural network techniques be implemented to improve the accuracy and training times of neural networks for the purpose of predicting student performance?" This paper attempts to answer this question in order to increase the validity of the use of this technology in the real world and hypothesizes that the implementation of these methods will improve training time and accuracy.

[1] Seeing this, Adam will be used as the "new" tested gradient descent method

METHOD

Model

Zhang's quantitative experiment comparing neural networks using a combination of various techniques will be modeled. Zhang preprocesses the data using PCA, training identical networks on either the unprocessed data or the processed data. He compares the accuracy and training time of each, noting improvements in both networks. Similarly, in this paper, neural networks with small variations will be tested to see if the efficiency increases. A method similar to Zhang, therefore, is able to isolate techniques to determine a cause and effect relationship between the techniques that were utilized and the resulting efficiency of the neural network, allowing the gap to be addressed effectively.

Here, a total of eight different combinations of techniques (PCA, dropout, Adam) will be tested, including one control network that mirrors the neural network used in Silva and Cortez's research. This is done to see if the usage of these techniques does increase accuracy and decrease training time. Each combination of techniques will be tested five times each in order to remove variability from the initial random weights. This approach is most appropriate for the task of comparing the effect of various techniques as other variables are controlled and so a cause and effect relationship can be determined between the techniques and efficiency (accuracy and training time).

Data

The data is from the Portugese public schools Gabriel Pereira Secondary School and Mousinho da Silveira Secondary School during the 2005-2006 school year. It is published on the UCI machine learning repository, and I will specifically be using data from the Portugese class, with 649 students and 33 features for each student. This is the most comprehensive dataset that was used in a previous research paper on the topic, with the variety of relevant features and large amount of students. The dataset was used in Silva and Cortez's research paper, so the control network trained on this data is representative of the neural network they constructed in their paper.

TABLE I
Features included in the Dataset

Feature name	Feature type	Feature Explanation
school	binary	student's school ('GP' - Gabriel Pereira or 'MS' - Mousinho da Silveira)
sex	binary	student's sex ('F' - female or 'M' - male)
age	numeric	student's age (from 15 to 22)
address	binary	student's home address type ('U' - urban or 'R' - rural)
famsize	binary	Family size ('LE3' - less or equal to 3 or 'GT3' - greater than 3)
Pstatus	binary	parent's cohabitation status ('T' - living together or 'A' - apart)
Medu	numeric	mother's education (0 - none, 1 - primary education (4th grade), 2 – 5th to 9th grade, 3 – secondary education or 4 – higher education)
Fedu	numeric	father's education (0 - none, 1 - primary education (4th grade), 2 – 5th to 9th grade, 3 – secondary education or 4 – higher education)
Mjob	nominal	Mother's job ('teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')
Fjob	nominal	Father's job ('teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')
reason	nominal	reason to choose this school (close to 'home', school 'reputation', 'course' preference or 'other')
guardian	nominal	student's guardian ('mother', 'father' or 'other')
traveltime	numeric	home to school travel time (1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour)
studytime	numeric	weekly study time (1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)
failures	numeric	number of past class failures (n if 1<=n<3, else 4)
schoolsup	binary	extra educational support (yes or no)
famsup	binary	family educational support (yes or no)
paid	binary	extra paid classes within the course subject (yes or no)

activities	binary	extra-curricular activities (yes or no)
nursery	binary	attended nursery school (yes or no)
higher	binary	wants to take higher education (yes or no)
internet	binary	Internet access at home (yes or no)
romantic	binary	In a romantic relationship (yes or no)
famrel	numeric	quality of family relationships (from 1 - very bad to 5 - excellent)
freetime	numeric	free time after school (from 1 - very low to 5 - very high)
goout	numeric	going out with friends (from 1 - very low to 5 - very high)
Dalc	numeric	workday alcohol consumption (from 1 - very low to 5 - very high)
Walc	numeric	from 1 - very low to 5 - very high
health	numeric	current health status (from 1 - very bad to 5 - very good)
absences	numeric	number of school absences (from 0 to 93)
G1	numeric	first period grade (from 0 to 20)
G2	numeric	second period grade (from 0 to 20)
G3*	numeric	second period grade (from 0 to 20)

* *Output of the neural network*

While the Portugese grading system is on the scale from 0-20, the values for columns G1, G2, and G3 will be modified to be on a five-level scale as the specificity of a 20 point scale is not necessary to identify the level of the student and a 20 point scale would make training extremely difficult for the neural network.

Data Preprocessing

The data will be preprocessed using the Pandas library also in the Python programming language. Pandas is a library in Python that is used for efficient data preprocessing. Python will be used due to the wide varieties of associated programs that allow the creation of neural networks in a simple and effective manner.

The features G1, G2, and G3 are modified using Pandas on the scale found below as done by Silva and Cortez.

TABLE II
Conversion of 20 point grading scale to 5 point

Score on 20 point scale	Score on the 5 point scale
0-9	5
10-11	4
12-13	3
14-15	2
16-20	1

The numeric input data will be normalized with a mean of 0 and a standard deviation of 1 through the fastai library to keep all the data on the same scale while preserving the variance of the data. This makes sure all features have the same weight on each particular node of the neural network.

Creating the Neural Networks

The neural networks will be created using the Fastai library in Python. Fastai implements the Pytorch library to construct machine learning programs, but Fastai is able to implement neural networks for tabular data more concisely compared to Pytorch, and therefore will be used.

All 8 variations will be tested on a neural network with an input layer with 33 input nodes (one for each feature listed above), one hidden layer with 20 nodes, and an output layer with 5 nodes.[2,3] The model outputs its certainty of the particular student achieving a specific grade on the 1-5 scale, and the most certain grade is considered the program's "guess". For example, if the program outputs its certainty of the student achieving a 1 is 10%, and a 2 at 90%, the program's "guess" would be a 2.

Concerning the categorical variables, the variables will be transformed into embeddings, which are a numerical representation that takes into account how similar each categorical variable is by analyzing the relationship between the data. Embeddings will be

[2] Silva and Cortez use an internal grid search to find the ideal amount of hidden nodes in the layer between 0 and 8, however when using dropout, more nodes must be implemented to reduce the probability of all nodes being turned off during a forward pass. Seeing this, a large number of nodes will be used.

[3] After initial testing, an error in the program occurred when using only one hidden layer, so another layer with a large amount of nodes was added to debug it.

used to describe categorical variables as the numerical representation of the data preserves the trends in the data and can be input into a neural network even though it does increase the dimensionality of the data.

The loss function that will be used is the Mean Squared Error function (MSE) rather than the Root Mean Squared Error function (RMSE) which was used in Silva and Cortez's paper. MSE is incredibly similar to RMSE as the only difference is that while MSE doesn't take square root of the error, while RMSE does. They are both commonly used in machine learning, however MSE will be used as the fastai library does not include the RMSE loss function.

The learning rate of the neural network will be determined using the fastai built-in lr.find() method that runs multiple passes through the neural network to determine the most optimal learning rate of the model. This will be run once for every variation to optimize the performance of the program.

The training data will be examined in mini-batches of size 4 as neural networks using mini-batches have consistently outperformed those utilizing larger batches, but speeds up the training process compared to networks that use no batches as stated in Ruder's paper.

After the training process is over using fastai's fit() method for 75 epochs (similar to the amount of epochs used in Silva and Cortez's paper) the training loss, validation loss, accuracy, and time taken per epoch will be written into a CSV file, which will then be analyzed.

Control Neural Network

The control neural network will be implementing Stochastic Gradient Descent (SGD), no dropout, and no feature engineering. This neural network will be used to simulate the neural network used in Silva and Cortez's, a neural network configuration that is currently used in academia, research to compare the use of the techniques in combination.

Tested Techniques

Seeing the effectiveness of PCA, Dropout, and the Adam optimizer in other fields of neural network research and how these techniques were unimplemented in the current literature surrounding predicting student performance, these techniques will be tested in combination. Therefore, there will be eight different tested combinations.

PCA will be implemented using the Scikit-learn library, but since the method of feature engineering can only be applied to quantitative data, after the numeric variables are configured to have a mean of 0 and a standard deviation of 1 these 15 numerical columns will be decreased by 27%, resulting in 11 numerical columns. This decreases the total number of columns to 29 from 33, resulting in a total 13% dimensionality reduction.

TABLE III
Neural Network Configurations and Utilized Techniques

Configuration	Dropout	Optimizer	Feature Engineering
1*	No	SGD	None
2	No	SGD	PCA
3	No	Adam	None
4	No	Adam	PCA
5	Yes	SGD	None
6	Yes	SGD	PCA
7	Yes	Adam	None
8	Yes	Adam	PCA

* *Control Neural Network*

Limitations

As Silva and Cortez did not make their code public, it's difficult to know for sure what specific parameters were used and therefore to replicate their exact program. While the control network attempts to recreate the neural network used in their paper, it most likely will not be exact due to this lack of knowledge. Consequently, results may be inconsistent when comparing the efficiency of the two networks. Furthermore, Silva and Cortez did not look at the training time of their neural network, so comparing results with their network cannot be done.

FINDINGS

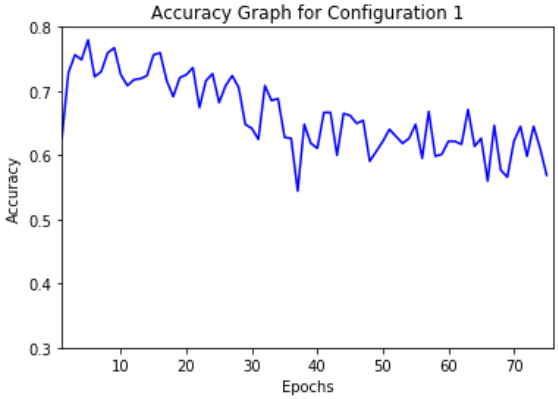

Figure 5: The accuracy graph for configuration 1

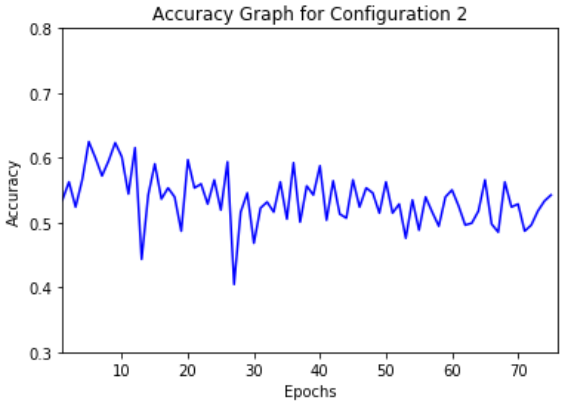

Figure 6: The accuracy graph for configuration 2

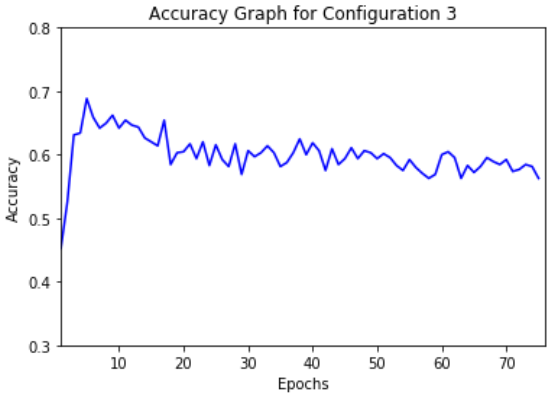

Figure 7: The accuracy graph for configuration 3

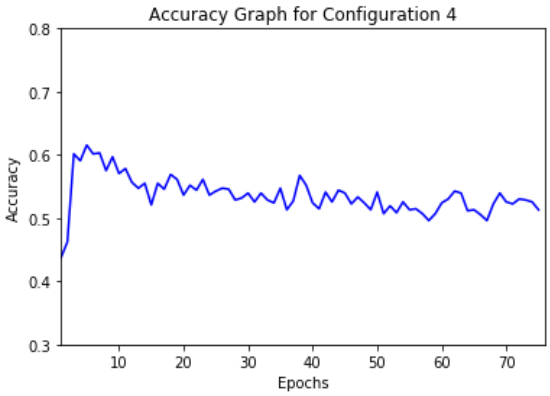

Figure 8. The accuracy graph for configuration 4

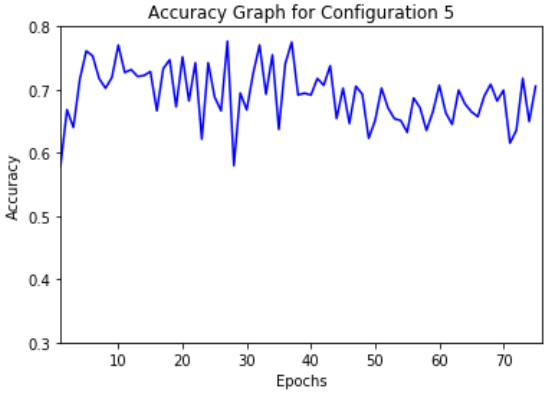

Figure 9: The accuracy graph for configuration 5

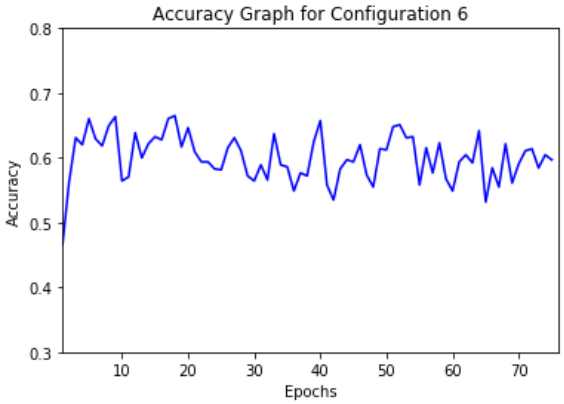

Figure 10: The accuracy graph for configuration 6

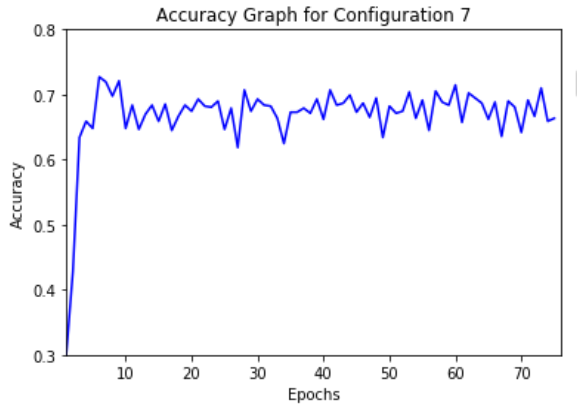

Figure 11: The accuracy graph for configuration 7

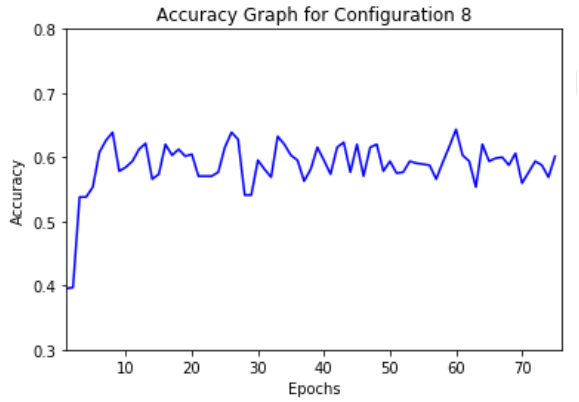

Figure 12. The accuracy graph for configuration 8

For all configurations 1-4, the neural network accuracies tended to decrease over epochs rather than increase. Over the 75 epochs, configuration 1 decreased from 76% to 56% fluctuating ±5% and configuration 2 (configuration 1 with PCA) decreased from 62% to 52% with large fluctuations of ±8%. Configuration 3 saw its accuracy decrease from 68% to 57% fluctuating ±2.5% and configuration 4 (configuration 3 with PCA) decreased from 62% to 54% with fluctuations of ±3%.

Configuration 5 saw a somewhat consistent accuracy of ~70%, but the accuracy fluctuates maximum of ± 10%. Configuration 6 saw an accuracy of ~58%, and the accuracy fluctuated about 5%.

Configuration 7 saw the best results, with a consistent accuracy of ~70% and only fluctuated ±3%. Configuration 8 (configuration 7 with PCA) had a consistent accuracy of ~60% and fluctuated ±5%.

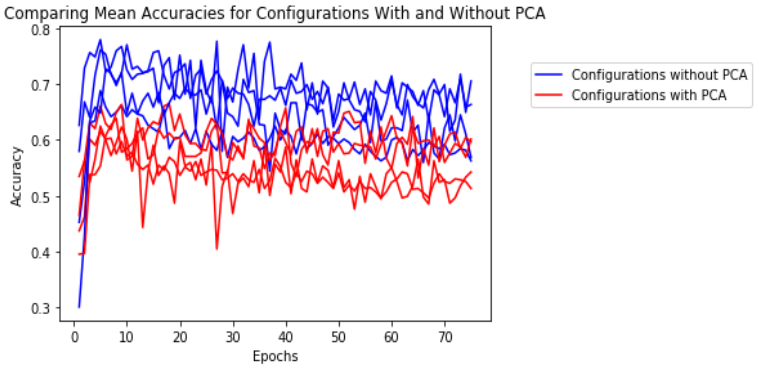

Figure 13: The accuracy graph for all configurations, color coded by the use of PCA

A noticeable trend was the significantly lower accuracies in PCA neural networks as noted in figure 13, comparing neural networks with PCA (configurations 2,4,6,8) and without PCA (configurations 1,3,5,7). This is due to the dimensionality reduction of the technique, however the difference is significant to the performance of the neural networks.

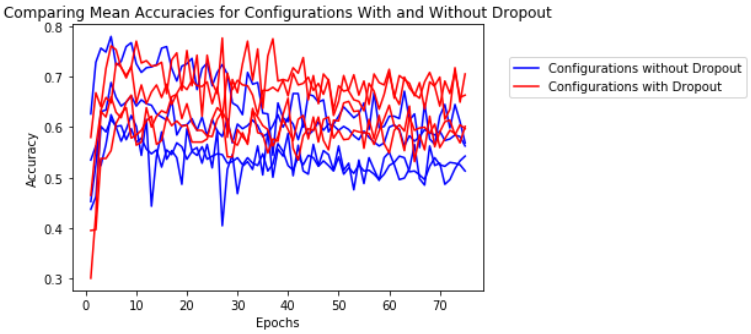

Figure 14: The accuracy graph for all configurations, color coded by the use of dropout

Using dropout increases the overall accuracy of the network, as seen in figure 14. This is also evident in the increasing accuracies in configurations 5-8, where dropout was used, compared to configurations 1-4, where it wasn't.

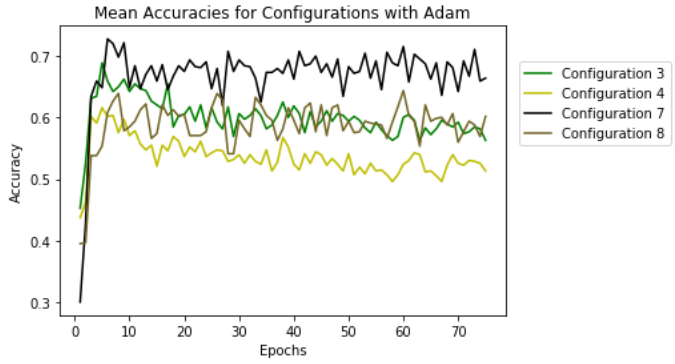

Figure 15: The accuracy graph for configurations that utilize Adam

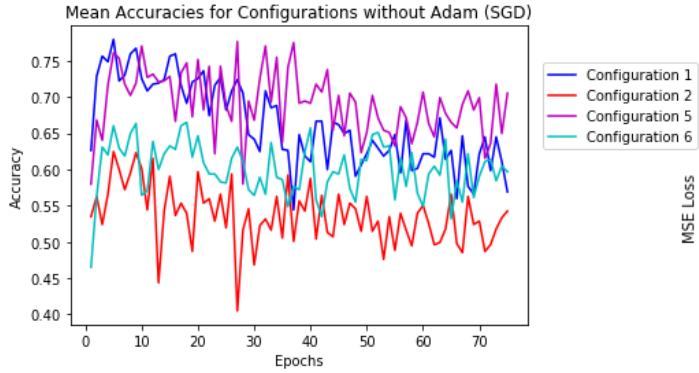

Figure 16: The accuracy graph for configurations that utilize SGD

The usage of Adam seemed to have little effect on the overall accuracies, as seen in figure 15, however did have a significant effect on the fluctuations of the accuracies, averaging a 3.38% accuracy fluctuation when used, whereas SGD had an average of 7% accuracy fluctuation.

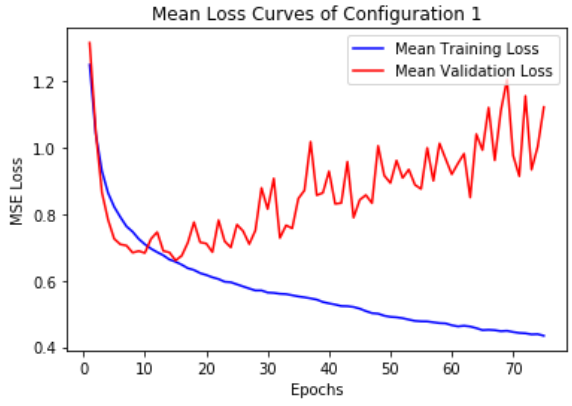

Figure 17: Mean loss curves of configuration 1

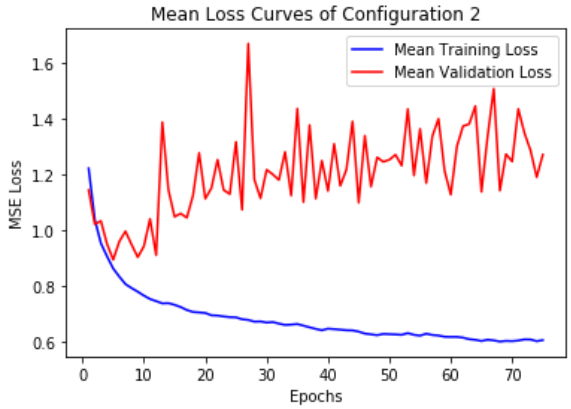

Figure 18: Mean loss curves of configuration 2

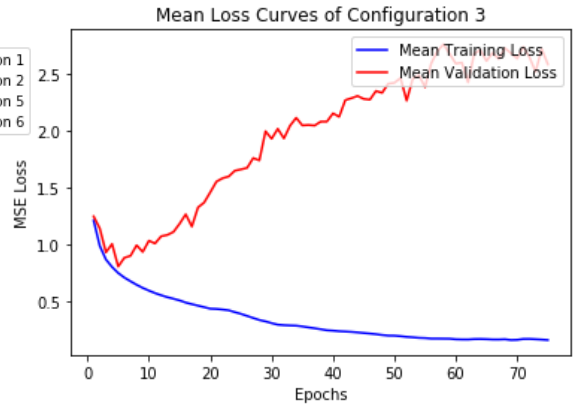

Figure 19: Mean loss curves of configuration 3

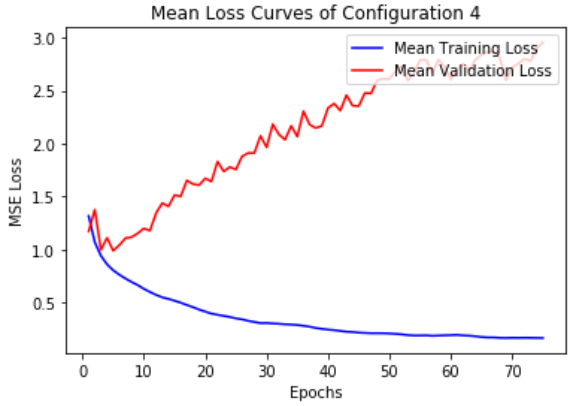

Figure 20: Mean loss curves of configuration 4

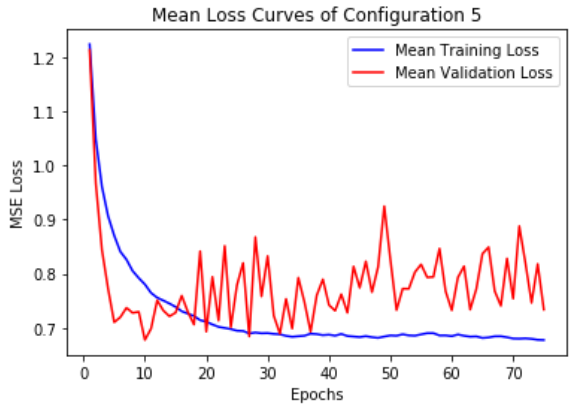

Figure 21: Mean loss curves of configuration 5

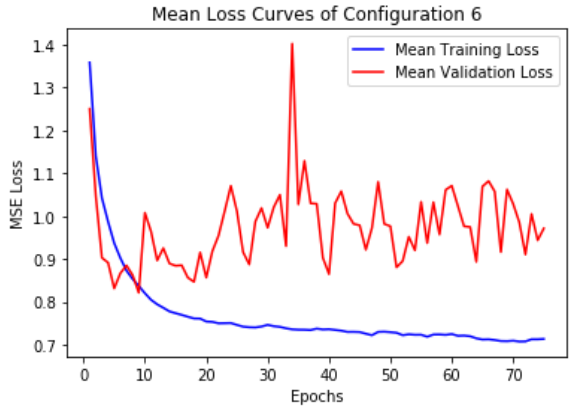

Figure 22: Mean loss curves of configuration 6

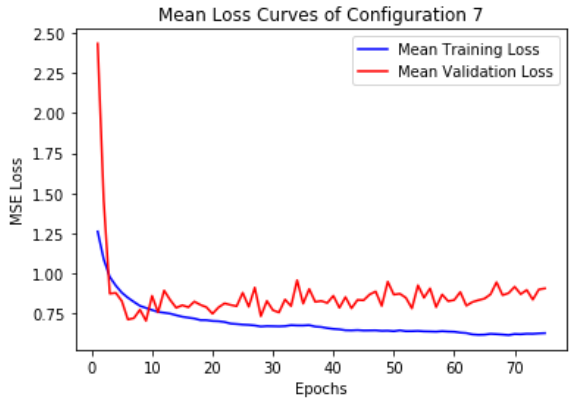

Figure 23: Mean loss curves of configuration 7

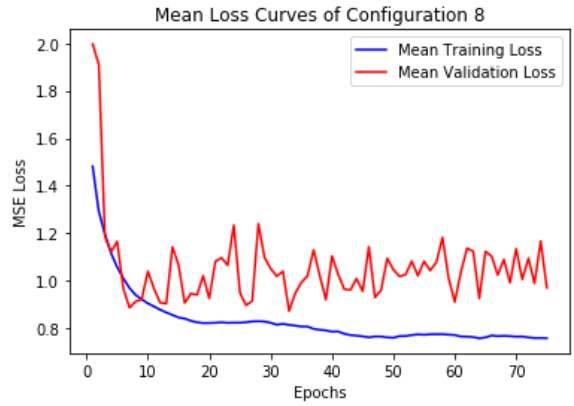

Figure 24: Mean loss curves of configuration 8

The usage of Dropout prevented overfitting in the neural networks, which can be seen in the loss graphs of the neural networks. Configurations 1-4 (no dropout) consistently had overfitting issues, with the training loss increasing and validation loss stagnating whereas configurations 5-8 (with dropout) saw decreasing training and validation losses without a consistent increase in either.

TABLE IV
Corresponding Training time for each Configuration

Configuration	Training Time (min)	Dropout	Optimizer	Feature Engineering
1*	89.99	No	SGD	None
2	89.78	No	SGD	PCA
3	93.69	No	Adam	None
4	92.36	No	Adam	PCA
5	91.23	Yes	SGD	None
6	90.70	Yes	SGD	PCA
7	89.62	Yes	Adam	None
8	92.77	Yes	Adam	PCA

* *Control Neural Network*

Training times with PCA were inconsistent. For configurations 1-6, the PCA counterpart to its original neural network had a 0.7% decrease (38 seconds) on training time, however when adding configurations 7 and 8 to this calculation, there was a

0.3% (16 seconds) mean increase when adding PCA to the configuration. There is no practically significant difference in training times when training for upwards of an hour and a half when using PCA.

Discussion

The combination of numerical and categorical data that described the students were effective in creating a well-performing neural network, seeing how the large number of inputs were able to accurately predict student performance. This is inline with Shaleena and Paul's findings. They state that neural networks are one of the best data mining methods for predicting student performance due to its nonlinear nature. Seeing the 70% accuracy of some of the neural networks discussed in this paper, this paper finds evidence to support Shaleena and Paul's statements.

70% accuracy for a 5 level classification was higher than the 65.1% accuracy in Silva and Cortez's paper which may seem small, but is a significant leap and proof that the inclusion of techniques in neural networks can increase the accuracy of the neural networks, but more research has to be done to improve the training time.

Dropout

The tendency for the models to overfit was a large issue for many of the configurations. Configurations that didn't utilize dropout (configurations 1-4) consistently had increasing validation losses and accuracies and stagnating training losses. This shows that networks with configurations 1-4 overly relied on certain nodes that were tuned well to the training set and couldn't generalize the data to the test set. Srivastava et al., who first developed dropout as a means to reduce overfitting, noted that over-tuning specific neurons can lead to large overfitting, contributing to a lower accuracy and validation loss, which was seen in the data.

Adam

Out of configurations 5-8, configurations 5 and 6, which used Dropout but not the Adam optimizer, took more epochs for the validation loss to stagnate and saw larger fluctuations in the accuracy and loss graphs compared to configurations 7 and 8, contributing to less consistent results in the accuracy graphs. In configurations 7 and 8, which utilized both Dropout and Adam, there were small changes in the vertical movements in the graph. Looking at the accuracy graphs of configurations 5-8, the most apparent differences are the vertical movements of the graph. Configurations 5 and 6 fluctuated around 10% and 5% respectively while configurations 7 and 8 fluctuated at around 3% and 5%. This is likely due to momentum, which is incorporated into the Adam algorithm but not SGD. As the losses decrease, momentum decreases the rate at which the weights change value, leading the accuracy to change at a smaller rate and fluctuating less. Less fluctuations in accuracy are beneficial to the neural networks since it leads to a more consistent performance of the neural network, improving the amount of cases the neural network can accurately predict.

In addition to decreasing fluctuations, the inclusion of Adam allowed the neural network to reach its peak performance faster. In the loss curve for configuration 7 and 8 (where Adam is used), the large decrease in loss is in the first 15 epochs and stagnates afterwards, showing a high rate of convergence. However, in configurations 5 and 6 (where SGD is used), it took around 25 epochs for the loss to decrease and begin stagnating, a lower rate of convergence. This is also likely due to momentum as the other key feature of it is that momentum speeds up the learning process before the loss begins to stagnate and learning slows down, leading to a faster peak compared to other gradient descent methods.

This is in line with Ruder's metaanalysis of gradient descent methods as he notes the inclusion of momentum in Adam to be a leading factor in aiding the neural network to reach optimal performance. Since other variables (technique, dataset, random weight initialization) were accounted for, we can see that the two effects of Adam and momentum in this study was the increased rate of convergence and decrease in accuracy fluctuation.

PCA

As Zhang mentioned in his paper, in theory, the dimensionality reduction of PCA logically should result in fewer calculations being made per epoch, leading to a lower training time, however this wasn't seen in this study. There were inconsistent results when applying PCA to neural networks, with an average increase in training time when PCA was applied. When training time did decrease in configurations 2, 4, and 6, the results proved to be practically insignificant as a 30 second decrease to a 90 minute training time isn't a large difference.

Furthermore, the networks that utilize PCA are seen to perform worse on average compared to those that don't. This was expected as Zhang, in his paper, also saw a decrease in accuracy similar to these results,

however, combined with the lack of practical significance in the decrease in training time, PCA did not have an overall positive impact on the neural networks it was implemented in.

The most efficient results (best combination of high accuracy and low training time) came from Configuration 7, which used Dropout and the Adam optimizer. Being able to predict 5% better than the neural network in Silva and Cortez's paper while using the same dataset, it is clear that the techniques of dropout and using more effective optimizers (such as Adam) have a statistically significant effect on the efficiency of a neural network system.

Limitations and Future Studies

While the possibility of improving the accuracy of neural network technology around predicting student performance has been validated, the extent to which it can be improved is still researchable. Since the neural networks' weights are initialized to be random at first, while low, there may be a slight chance that the results were impacted by this randomness. In addition, the training times saw little to no change with the inclusion of PCA, calling for more research into techniques that would improve the training time and thereby the overall efficiency of neural networks.

As this study used embeddings for categorical variables, the dimensionality of each categorical variable was low, but the overall number of features was most likely higher than anticipated. This would mean when PCA was applied, it reduced the dimensionality of the dataset more than what was discussed in the findings section, which would likely mean there was a larger decrease in accuracy than a true dimensionality reduction of 13%.

Since the field of data mining continues to see constant developments, new techniques in the realm of feature engineering and gradient descent algorithms should continue to be tested to improve the efficiency of neural networks. For instance, as Ruder described in his paper, algorithms such as Nadam (Nesterov-accelerated Adaptive Moment Estimation) which combines Nesterov accelerated gradient and Adam, builds on previous techniques to improve the gradient descent process for the highest efficiency, which should be tested and be used to explore the true potential of data mining methods.

CONCLUSION

This paper finds the accuracy of neural networks to be improved by the use of various techniques. Dropout led to a higher accuracy and prevented overfitting in the dataset, resulting in a well tuned neural network that was able to accurately predict new data and create generalizations rather than repredict previous training data. Using the Adam optimizer over SGD leads to a lower variability in the accuracy between epochs of training, allowing for a more constant increase in accuracy. Combined, these two techniques were able to create a neural network that had a 70% accuracy for predicting student performance on a 5 level scale, which is higher than what was seen in the current literature surrounding the same topic of predicting student performance. The use of Principal Component Analysis was tested too. While it had the intent of reducing training time by decreasing the size of the dataset, there was too great of a decrease in accuracy and no significant decrease in training time for it to be viable in this context, which calls for more research into the development of techniques to improve the training time of neural networks.

BIBLIOGRAPHY

[1] F. Bre, "Artificial Neural Network Architecture." Nov. 2017, [Online]. Available: https://www.researchgate.net/profile/Facundo_Bre/publication/321259051/figure/fig1/AS:614329250496529@1523478915726/Artificial-neural-network-architecture-ANN-i-h-1-h-2-h-n-o.png.

[2] K. P. Shaleena and S. Paul, "Data mining techniques for predicting student performance," *2015 IEEE International Conference on Engineering and Technology (ICETECH)*. 2015, doi: 10.1109/icetech.2015.7275025.

[3] A. M. Shahiri, W. Husain, and N. A. Rashid, "A Review on Predicting Student's Performance Using Data Mining Techniques," *Procedia Comput. Sci.*, vol. 72, pp. 414–422, Jan. 2015.

[4] A. Tekin, "Early Prediction of Students' Grade Point Averages at Graduation: A Data Mining Approach," *Eurasian Journal of Educational Research*, no. 54, pp. 207–226, 2014.

[5] I. Lykourentzou, I. Giannoukos, G. Mpardis, V. Nikolopoulos, and V. Loumos, "Early and dynamic student achievement prediction in e-learning courses using neural networks," *J. Am. Soc. Inf. Sci.*, vol. 60, no. 2, pp. 372–380, Feb. 2009.

[6] E. Osmanbegovic and M. Suljic, "Data Mining

Approach for Predicting Student Performance," *Economic Review: Journal of Economics and Business*, vol. 10, no. 1, pp. 3–12, 2012.

[7] A. Silva and P. Cortez, "Using Data Mining To Predict Secondary School Student Performance," *EUROSIS*, Jan. 2008.

[8] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, Jun. 2014.

[9] S. Schwartz, "An overfitting sample Neural Network." Jul. 2017, [Online]. Available: https://i.stack.imgur.com/m0NAK.png.

[10] J. Brownlee, "Example of Train and Validation Learning Curves Showing a Good Fit." Feb. 2019, [Online]. Available: https://3qeqpr26caki16dnhd19sv6by6v-wpengine.netdna-ssl.com/wp-content/uploads/2018/12/Example-of-Train-and-Validation-Learning-Curves-Showing-A-Good-Fit.png.

[11] J. Zhang, "Machine Learning With Feature Selection Using Principal Component Analysis for Malware Detection: A Case Study," *arXiv [cs.CR]*, Feb. 10, 2019.

[12] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv [cs.LG]*, Sep. 15, 2016.