ISSN 2278-3091
Volume 9, No.5, September - October 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse287952020.pdf
https://doi.org/10.30534/ijatcse/2020/287952020

Improving Performance of Convolutional Neural Network in Movie Rating Prediction Using Hyperparameter Tuning

Rudy Aditya Abarja[1], Antoni Wibowo[2], Dewi Retno Sari Saputro[3], Norhaslinda Zainal Abidin[4], Rudolf Jason Kwaria[5]
[1,2] Computer Science Department,
BINUS Graduate Program - Master of Computer Science,
Bina Nusantara University, Jakarta, Indonesia 11480
[1]rudy.abarja@binus.ac.id, [2]anwibowo@binus.edu
[3] Program Studi Matematika FMIPA UNS,
Universitas Sebelas Maret, Indonesia 57126
dewiretnoss@staff.uns.ac.id
[4] Institute of Strategic Industrial Decision Modelling, School of Quantitative Sciences,
Universiti Utara Malaysia, 06010 UUM Sintok, Kedah, Malaysia
nhaslinda@uum.edu.my
[5] Department of Materials Science and Engineering,
School of Materials and Chemical Technology,
Tokyo Institute of Technology, Yokohama, Kanagawa 226-8502, Japan
jason@mi-6.ac.jp

ABSTRACT

Predicting movie rating before the movie is released become unsolved problem for movie prediction case. Many existing researches about movie rating prediction failed to address this problem because they usedthe post-release elements such as social media comments to predict movie rating. Our previous study addressed this problem by using historical features and other features that obtained before movie is released. In this study we used the features that we found in our previous. This study focused on optimizing the one-dimension CNN that we used on our previous study. Several studies were conducted to proposed technique to optimize DNN. We combined several optimization techniques such as regularization, activation function, optimizer, loss function and batch normalization to optimized CNN. The implementation of several techniques such as ReLU, dropout, Adam, L2 weight regularizer and batch normalization proved to give performance boost in our CNN model. After hyperparameter tuning is completed, our CNN model proved to give promising result by having the best performance compared to other machine learning models such as SVR and ANN.

Key words: Movie rating, rating prediction, historical values, CNN, 1DNN, dropout, hyperparameter tuning

1. INTRODUCTION

Movie is one of the most popular entertainment for modern day people as a source of relaxation and entertainment[1].It has become the most important source of source of entertainment for people throughout the world. Moreover, movie can be considered a work of art that makes people gone crazy about it [2].

In the past five years, United States and Canada has released 765 movies per year in average. In 2019, 835 movies were released in the US and Canada, this number has increased by 70 movies compared to 2018 [3]. Because there are a large number of movies released, people would need a guide to judge whether a movie is good or not so that they won't spend their money watching bad movies[4]. Numerical rating for movie is easier to understand than movie reviews [5]. Therefore, rating has acritical role in determining whether a movie would be worth to be watched or not.

Usually ratings and reviews come out after the movie is released. However, there is a chance that newly generated ratings after a new movie is released to be biased because it comes from a small group of audience watching before the general public watch [6]. One of the possible factors that might cause this bias is the possibility that the initial audience consists of paid viewers [7]. Given these problems, an objective rating is needed before the movie is released. The challenge of previous studies so far is to predict movie ratings before they are released in cinemas or even before they are produced.

Our previous study found that the historical features had the best performance compared to other features [8]. Therefore, it is possible to predict the movie rating before the movie is released.One of the unique features that being used in our previous study is historical features. These features were aggregate attribute that created from relationship between a movie and previously released movies. We assumed that the predicted rating based on these historical features is more

objective than the rating that generated by early audience. Ning et al. call this method a cohort rating prediction [9].

Many previous studies used basic machine learning model or statistical model such as linear regression [10, 11, 12, 13], support vector regression (SVR), and k-NN [14, 15]. However, Ning et al. used generative convolutional neural network. In that study they compared CNN with several popular machine learning models. Based on Ning et al. experiment, deep learning models such as CNN and LSTM had better performance than basic machine learning models [9].In our previous study, CNN was also used as machine learning model and it had promising movie rating prediction performance by beating several machine learning models[8]. Moreover, CNN proved to have good performance for cases beside movie rating prediction such as face recognition [16] and video spam detection [17].

Previous study has suggested that CNN have promising performance [8]. However, it was also discovered that the hyperparameter needs to be tuned based on our further experiment result after we conducted previous study. Our optimized CNN in previous study had lower performance compared to popular regression model such as SVR and linear regression. This study focused on CNN optimization process using hyperparameter tuning. For the features, we used the features that we extracted in our previous study [8]. This study tried to improve the one-dimension CNN that we used on the previous study. We did hyperparameter tuning for layer architecture, activation function, optimizer, regularizer, loss function and normalizer. Nevertheless, the optimizations were implementedin this study proved to give performance boost compared to optimized CNN in our previous study and other machine learning models such as SVR and ANN.

2. PREVIOUS WORKS

Previous studies [18, 15, 19] used a collaborative filtering approach. These studies utilized user data to make predictions, because of that the predicted rating is made for specific user. Some studies used social media as dataset[10, 12]. Besides that, there were studies that used video as data source [20, 21]. Based on the data source, these studies did not focus on rating prediction before the movie was released. There were studies that used mixed data source from several website and social media[13, 22, 23, 24, 9]. However, only studies [9, 24]focused on using data that generated before the movie is released.

The machine learning models used in the previous researches [15, 18, 19, 10, 12, 20, 21, 13, 25, 23] were mostly still using non-deep learning models, such as: SVD, k-NN, regression tree, linear regression, HMM, SVR, Factorization Machine, and simple neural network. There were only two studies that use deep learning models [9, 22]. Previous study by Ning et al. proved that deep learning model performance is better than other models [9].

Our previous study found that the historical features had the best performance compared to metadata, categorical, topical and social features [8]. Moreover, the combination of historical, metadata, categorical and topical features as feature set got the best performance compared to other feature sets. Historical, metadata, categorical and topical features were obtained from movie attributes which are available before the movie is released. One-dimension CNN (1DCNN) was used in our previous study as machine learning model [8]. We did basic optimization for the CNN by modifying layers and implementing dropout. The implementation of dropout proved to give performance boost to baseline CNN. Our optimized CNN in the previous study got the best performance compared to several machine learning models. Although it had promising performance, it had many rooms to improve.

One of important problem in machine learning is how to make sure that model performs well not only on training data, but also performs well on new data [26]. Regularization is the strategies used in machine learning are explicitly designed to reduce errors in the test process, perhaps at the expense of increasing errors in the training process [26]. In our previous study we used dropout as regularization method and it gave positive impact to our CNN model. Dropout is a technique to reduce overfitting in deep neural network model [27].

Besidesdropout, there are different regularization techniques to reduce overfitting. DropConnect is technique that generalizes dropout by randomly dropping the weights rather than the activations on layer [28].Tompson et al. proposed different dropout method that drops entire feature maps instead of individual elements [29]. This method is called spatial dropout, this method help promote independence between feature maps and should be used instead.

Activation function was also used to improve deep learning model performance. Agarap, Abien Fred M. conducted study to implement Rectified Linear Units (ReLU) in deep neural network (DNN) [30]. That study shown that ReLU had promising performance. Zang et al. used dropout and ReLU activation function to optimize multiple sclerosis identification using CNN [31]. Dahl et al. also used dropout and ReLU to improve DNN performance for large vocabulary continuous speech recognition case [32]. The combination of dropout and ReLU proved to have promising result in previous studies [31, 32].

Optimizer is a method used to change the attributes of neural network such as weights and learning rate in order to reduce the losses. There are several studies about optimizer. Sutskever et al. successfully implemented stochastic gradient descent (SGD) with momentum [33]. SGD uses a well-designed random initialization and a particular type of slowly increasing schedule for the momentum parameter. Kingma et al. proposed an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments called Adam [34].Dozat, Timothy proposed a method that implemented Nesterov momentum into Adam, this method is called Nadam[35].

Zeiler, Matthew D. proposed method called Adadelta which is a stochastic gradient descent method that is based on

adaptive learning rate per dimension to address two drawbacks: (1) the continual decay of learning rates throughout training and (2) the need for a manually selected global learning rate [36]. Duchi et al. proposed optimizer method called Adagrad[37]. Adagrad is an optimizer with parameter-specific learning rates, which are adapted relative to how frequently a parameter gets updated during training. The more updates a parameter receives, the smaller the updates. McMahan et al. implemented FTRL algorithm as optimizer to improve in the context of traditional supervised learning for ad click through rates (CTR) prediction case [38].

Another approach to improve neural network model is using normalization. Training DNN is because the distribution of each layer's inputs changes during training, as the parameters of the previous layers change. This causing the slowdown of the training process. This phenomenon is called internal covariate shift and could be solved by normalizing layer inputs. Ioffe, S., & Szegedy, C. proposed method called batch normalization[39]. Training deep neural networks with tens of layers is challenging as they can be sensitive to the initial random weights and configuration of the learning algorithm. Batch normalization is a technique for training DNN that standardizes the inputs of a layer for each mini-batch. This method stabilizes the learning process and dramatically accelerating the training of deep networks.

Based on the literature study, our research used hyperparameter tuning to improve performance of 1DCNN. We tried several available approaches such as regularization, activation function, optimizer, loss function and batch normalization. We treated each techniques category as a hyperparameter. This study would try several hyperparameter settings for each category to optimize the performance of 1DCNN movie rating.

3.PROPOSED METHOD

3.1 Dataset

This study used the combined IMDb and TMDb datasets obtained from Kaggle. The movies were released from 1916 until 2013will be used as training dataset and the movie were released from 2014 until 2017 will be used as training dataset. Only movies which has English as primary spoken language are included in the dataset.

3.2Feature Selection

The evaluation of selected feature is needed to make sure we use the best feature set as input. Based on our previous study, we found that historical features had the best performance compared to metadata, categorical, topical and social features. We also found that the combination of historical, metadata, categorical, and topical features had the best performance compared to others feature set [8]. In our previous study, we used baseline CNN to compare the performance of feature and feature set. In current study, linear regression will be used to evaluate feature set performance. Linear regression was used because neural network models tend to have variance in performance for each model training process, while linear regression has a fixed performance. The variance in CNN is caused by random weight feature when we use deep learning library from cuDNN. Mean square error (MSE) and mean absolute error (MAE) will be used to measure the performance.

3.3CNN Optimization

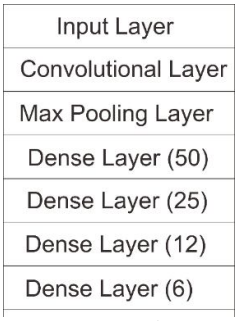


Figure 1: Architecture of baseline CNN

CNN optimization process is needed to improve movie rating prediction accuracy of CNN. We used one-dimensional CNN (1DCNN) employed in our previous study[8]. Figure 1 is the architecture of the baseline CNN. The CNN optimization process consists of hyperparameter tuning. The optimization process is conducted to find the best hyperparameter of several attributes and divided into six part.

These are the optimization process: (1) find best convolutional layer - max pooling layer pair number, (2) find best node number in dense layer, (3) find best activation function, (4)find best optimizer for CNN with dropout, (5) find best regularization method, (6) find best loss function and (7) find the effect of implementing batch normalization in CNN. MSE and MAE are used to measure the effect of the hyperparameter tuning. Model with the best performance in each part will be used for the next part of optimization process.

3.3.1 Convolutional Layer – Max Pooling Layer

Finding the best number for convolutional layer – max pooling layer pair is the first step of the first optimization part. There were three models with different number for convolutional layer – max pooling layer configuration(2 pair, 3 pair and 4 pair) to be tested.

3.3.2 Dense Layer Node

The next step is to find the best number of each dense layer node.

Table 1: Dense layer number configuration

Model	1st Layer	2nd Layer	3rd Layer	4th Layer
Baseline	50	25	12	6
1	200	100	50	25
2	100	50	25	12
3	87	44	22	11
4	64	32	16	8
5	32	16	8	4

The CNN models that were used in this study consists of four dense layers. Table 1 shows models with different combination of node number in dense layers.

3.3.3 Activation Function

Optimization of activation function used model with the best convolutional layer – max pooling layer pair number and combination of best node number in dense layers which is found in the previous two optimization process part. We experimented with these activation functions: (1) Tanh, (2) Rectified linear unit (ReLU), (3) Scaled exponential linear unit (SELU) and (4) Exponential linear unit (ELU).

3.3.4Optimizer

The fourth process used the best convolutional layer – max pooling layer pair number, node number in dense layer and activation function which is found in the previous optimization processes. The difference is the addition of dropout layer in the model, with the probability of each dropout layer is 50% (0.5). We usedthese optimizers: (1) RMSprop, (2) Adam, (3) Nadam,(4) Stochastic gradient descent (SGD), (5) Adadelta, (6) Adagrad and (7) Adamax.

3.3.5 Regularization

After finding the best hyperparameter in the previous four parts, we optimized regularization method of the CNN. There are two processes in this part: (1) find the best regularization layer configuration and (2) find the best weight regularization configuration.

For the first process, there were three different regularization layers. Dropout (DO),spatial dropout 1D (SD1D)and dropconnect (DC)were used in the first process. Spatial dropout 1D can be only used in the max pooling layer because its need tensor matrix input, while dense layer only has 2-dimension matrix because it is received input from flatten layer. Table 2 shows models with different regularization layer configuration.

Table 2: Regularization layer configuration list

Model	Convolutional – Max Pooling Layer	Dense Layer
1	-	-
2	DO(0.1)	DO(0.1)
3	DO(0.2)	DO(0.2)
4	DO(0.5)	DO(0.5)
5	DO(0.1)	-
6	DO(0.2)	-
7	DO(0.1)	DO(0.2)
8	DO(0.2)	DO(0.1)
9	SD1D(0.1)	-
10	SD1D(0.2)	-
11	SD1D(0.5)	-
12	SD1D(0.2)	DO(0.1)
13	SD1D(0.2)	DO(0.2)
14	DC (0.1)	DC (0.1)
15	DC (0.2)	DC(0.2)
16	DC (0.5)	DC (0.5)

The second process was conducted to find best hyperparameter for weight regularizer for 2nd and 4th dense layer. L1 regularizer, L2 regularizer and L1 L2 regularizer are used in this process. The best performance model from the first process is used in the second process as base model. Table 3 shows models with different weight regularization configuration.

Table 3: Weight regularization layer configuration list

No	Regularizer	Weight penalty				
		1	2	3	4	5
1	L1	0.1	0.5	0.01	0.05	0.001
2	L2	0.1	0.5	0.01	0.05	0.001
3	L1 L2	0.1	0.5	0.01	0.05	0.001

3.3.6 Loss Function

This process is conducted to find the best loss function. Model with best performance in the previous part is used as base model. These are the lost functions that will be experimented: (1) Mean squared error (MSE), (2) Mean absolute error (MAE), (3) Huber loss (0.5), (4) Huber loss (1.0), (5) Huber loss (1.5), (6) MeanSquaredLogarithmicError and (7) MeanAbsolutePercentageError.

3.3.7 Batch Normalization

This part was conducted to analyze the effect of batch normalization for CNN model. Model from the previous part was used as base model. Batch normalization wasimplementedinto convolutional layer of the model.

We then compared the performance of CNN model which was implemented batch normalization to the model without batch normalization. Each model is trained and evaluated 10 times. After training process, each model average MSE and MAE is compared to find the best model.

3.4 Evaluation

Evaluation process is needed to show the result ofthe optimization process of CNN model. Each model in this processwas trained and evaluated by training dataset. Afterwards, the evaluation processwas conducted by using test dataset. Mean squared error (MSE) and mean absolute error (MAE) were used as the performance metrics to evaluate the model. MSE and MSE show the average of magnitude error for the predicted ratings. MAE formula is shown in Eq (3) and MSE is shown in Eq (4).

$$Mean\ absolute\ error\ = \frac{1}{N}\sum_{i=1}^{N}|f_i - \tilde{y}_i| \quad (3)$$

$$Mean\ square\ error\ = \frac{1}{N}\sum_{i=1}^{N}(f_i - \tilde{y}_i)^2 \quad (4)$$

After the CNN optimization was finished, we conducted evaluation process to compare CNN performance with other machine learning models. Linear regression (LR), support vector regression (SVR), Gaussian process regression (GPR), stochastic gradient descent regressor (SGDR), decision tree regression (DTR), k-Nearest neighbor regression (kNNR) and artificial neural network (ANN) were used as comparison models.

The next step after finding two best performer models were independent t-test. Independent t-test was conducted to find out how significant the difference in movie rating prediction accuracy between the two-best model. The formula to find t value is shown in Eq (5).

$$t = \frac{m_A - m_B}{\sqrt{\frac{S^2}{n_A} + \frac{S^2}{n_B}}} \quad (5)$$

4. EXPERIMENTS

There were three experiments that would be used in this study. The first experiment purpose would be to compare the performance of several feature sets using linear regression. The second experiment was the CNN optimization process. The third experiment would be evaluation process to compare optimized CNN with other machine learning models. We used keras, TensorFlow and cuDNN for deep learning library. While scikit-learn was used as library for comparison models.

4.1 Dataset

The dataset used in these experiments was the pre-processed dataset we used from previous study [8]. The dataset contained4.317 movies which were released from 1916 until 2016. Movies released from 2000 until 2013 were used as training dataset (3.819 movies) and movies released after 2013 were used as test dataset (498 movies). The dataset already transformed into feature vector for the model input usage.

4.2 Feature Selection

In our previous study we measured the feature performance using baseline CNN as model [8]. In this study we used linear regression as model, the reason is already mentioned in Chapter 3.2. Table 4 shows the performance of each features.

Table 4: Feature performance

Features	MSE	MAE
Social features	1.43394	0.93363
Topical features	1.47	0.94
Social features (MinMax)	1.43394	0.93363
Categorical features	1.35	0.88
Metadata features (MinMax)	1.26406	0.87379
Metadata features	1.26406	0.81193
Historical features	1.11064	0.87379
Historical features (MinMax)	**1.11064**	**0.81193**

The result was similar with our previous study, historical features had the best performance and the MinMax scaler gave positive impact to the performance. Therefore, MinMax scaler would be used for the features with numerical type.

For the feature set experiment, we also used same feature set as our previous study [8]. Table 5 shows the performance of each features set. The result of feature set experiment was similar with our previous study, the combination of metadata, historical, categorical and topical features had the best performance. Therefore, dataset with attributes from this feature set was used for feature dataset. There were 87 attributes in total for the feature dataset.

Table 5: Feature set performance

Features Set	MSE	MAE
Raw dataset	1.66	2.29
Categorical + topical	1.35	0.88
Categorical + topical + social	1.32	0.87
Metadata + categorical + topical + social	1.14	0.82
Metadata + categorical + topical	1.13	0.82
Metadata + historical	1.11	0.79
Metadata + historical + social	1.11	0.8
Historical + categorical + topical + social	1.09	0.8
All features (without minmax)	1.072	0.79
All features	1.07	0.79
Metadata + historical + categorical + topical (without minmax)	1.064	0.786
Metadata + historical + categorical + topical	**1.062**	**0.785**

4.3 CNN Optimization

As mentioned before in Chapter 3.1, CNN optimization process consisted from seven processes of hyperparameter tuning. Each model was trained and evaluated using training dataset. After that, model was evaluated by test dataset. Each training process was consisted of 500 epochs. The best performing model from each optimization part was used as base model for the next part of optimization process. The baseline model for the experiments is the baseline CNN from our previous study, not the optimized one [8].

4.3.1 Convolutional layer- Max Pooling Layer Number

Table 6 shows the result of the experiment to find best number of convolutional– max pooling layer pair. Model 2 that consisted of two pair convolutional – max pooling layer pair had the best result in this experiment with MSE 1.45 and MAE 0.94. This result showed that the greater number of convolutional – max pooling layer pair does not always give increase in performance.

Table 6: Convolutional – max pooling layer result

Layer pair	MSE	MAE
1	1.47	0.94
2	**1.45**	**0.94**
3	1.64	0.97
4	1.59	0.95

4.3.2 Node in Dense Layer

Table 7 shows the result of the experiment to find best number of nodes in each dense layer. There are four dense layers in the CNN model used. Model 3 with configuration of 87, 44, 22 and 11 nodes for1st, 2nd, 3rd and 4th dense layer had the best performance by obtaining MSE 1.44 and MAE 0.92. This result showed that the greater number of nodes number in dense layer always give increase in performance.

4.3.3 Activation Function

Table 8 shows the result of the experiment to find best activation function for the convolutional layers and dense layers. Model 4 that used ReLU as activation function has the best performance with MSE 1.38 and MAE 0.84.

Table 7: Dense layer node number experiment result

Model	Layer				MSE	MAE
	1st	2nd	3rd	4th		
Baseline	50	25	12	6	1.47	0.94
1	200	100	50	25	1.46	0.94
2	100	50	25	12	1.56	0.95
3	**87**	**44**	**22**	**11**	**1.44**	**0.92**
4	64	32	16	8	1.54	0.92
5	32	16	8	4	1.47	0.94

Table 8: Activation function experiment result

No	Activation Function	MSE	MAE
1	tanh	1.67	0.97
2	SELU	1.53	0.93
3	ELU	1.61	0.97
4	**ReLU**	**1.38**	**0.90**

4.3.4 Optimizer

Table 9 shows the result of the experiment to find best optimizer for CNN model with regularizer layer. Model 2that used Adam as optimizer has the best performance with MSE 1.14 and MAE 0.84.

Table 9: Optimizer experiment result

No	Optimizer	MSE	MAE
1	RMSprop	1.24	0.86
2	**Adam**	**1.14**	**0.84**
3	Nadam	1.21	0.84
4	SGD	1.30	0.91
5	Adadelta	8.75	2.75
6	Adagrad	2.90	1.47
7	Adamax	1.17	0.085
8	Ftrl	1.48	1.00

4.3.5 Regularizer

There were two part in regularizer experiment, first was conducted to find best regularization hyperparameter and second was to find best weight regularization hyperparameter.Table 10 shows the result of the experiment to find best regularization layer hyperparameter. Model 7 that implemented dropout with 10% chance in convolutional layer and dropout with 20% chance in dense layer had the best performance. This model had MSE 1.11 and MAE 0.80. This experiment showed that dropout had better performance compared to spatial dropout 1D and dropconnect. In addition, we also found that all regularization layer with 50% chance had worse performance compared to regularization layer with 10% and 20% chance.

The second part was conducted to find best weight regularizer hyperparameter in 2nd and 4th dense layer. Table 11 shows the result of the second part of regularization experiment. Model 9 that implemented L2 weight regularizer with 0.01 penalty value had the best performance. This model obtained MSE 1.04 and MAE 0.78. This experiment result showed that L2 weight regularizer had better overall performance compared to L1 weight regularizer and L1 L2 weight regularizer. In addition, the result also showed that weight penalty with value 0.5 had the worst performance in all weight regularizer method. This can be concluded that for CNN model that used in this study, optimal value range for weight penalty is lesser than 0.5.

Table 10: Regularizer layer experiment result list

No	Convolutional Layer	Dense Layer	MSE	MAE
1	-	-	1.14	0.84
2	DO(0.1)	DO (0.1)	1.16	0.83
3	DO (0.2)	DO (0.2)	1.16	0.81
4	DO (0.5)	DO (0.5)	1.39	0.90
5	DO (0.1)	-	1.42	0.86
6	DO (0.2)	-	1.17	0.81
7	DO **(0.1)**	DO **(0.2)**	**1.11**	**0.80**
8	DO (0.2)	DO (0.1)	1.16	0.81
9	SD1D (0.1)	-	1.42	0.86
10	SD1D(0.2)	-	1.17	0.81
11	SD1D(0.5)	-	1.23	0.84
12	SD1D(0.2)	DO (0.1)	1.20	0.83
13	SD1D(0.2)	DO (0.2)	1.12	0.80
14	DC (0.1)	DC (0.1)	1.37	0.87
15	DC (0.2)	DC (0.2)	1.77	1.00
16	DC (0.5)	DC (0.5)	5.38	1.69

Table 11: Weightregularizerexperiment result

No	Weight Regularizer	MSE	MAE
1	-	1.11	0.80
2	L1 (0.1)	1.48	0.95
3	L1 (0.5)	1.52	0.95
4	L1 (0.01)	1.17	0.82
5	L1 (0.05)	1.47	0.97
6	L1 (0.001)	1.07	0.81
7	L2 (0.1)	1.05	0.79
8	L2 (0.5)	1.11	0.80
9	**L2 (0.01)**	**1.04**	**0.78**
10	L2 (0.05)	1.05	0.78
11	L2 (0.001)	1.10	0.80
12	L1 L2 (0.1)	1.48	0.95
13	L1 L2 (0.5)	1.49	0.95
14	L1 L2 (0.01)	1.13	0.80
15	L1 L2 (0.05)	1.47	0.94
16	L1 L2 (0.001)	1.07	0.81

4.3.5 Loss Function

Table 12 shows the result of the experiment to find best loss function for the used CNN model. Model 4 that used MSE as loss function had the best performance with MSE 1.04 and MAE 0.78.

4.3.6 Batch Normalization

Table 13 shows the result of the experiment to find the impact from implementing batch normalization in CNN model. In this experiment we compared the performance CNN model with batch normalization and without batch normalization. Each model was trained and evaluated 10 times. After that, we compare the average MSE and MAE for the two models. Model 4 that used batch normalization had

better performance with MSE 1.051 and MAE 0.778 compared to model that did not use batch normalization.

Table 12: Regularization layer configuration list

No	Loss Function	MSE	MAE
1	Huber Loss (delta = 0.5)	1.07	0.79
2	Huber Loss (delta = 1.0)	1.09	0.82
3	Huber Loss (delta = 1.5)	1.12	0.79
4	**MSE**	**1.04**	**0.78**
5	MAE	1.12	0.79
6	MeanSquaredLogarithmic	1.10	0.80
7	MeanAbsolutePercentageError	1.12	0.82

Table 13: Batch normalization experiment result

No	Normalization Layer	Avg MSE	Avg MAE
1	Batch normalization	**1.051**	**0.778**
2	Without batch normalization	1.085	0.78

After finishing CNN optimization process, we found the best hyperparameter for our CNN model. Table 14 shows the hyperparameters for optimized CNN model.

Table 14: Optimized CNN's hyperparameter

Hyperparameter	Value
Convolutional layer – max pooling layer pair	2
Nodes in dense layer	87, 44, 22, 11
Activation function	ReLU
Optimizer	Adam
Regularizer layer	Dropout (0.1 & 0.2)
Weight regularizer	L2 (0.01)
Loss function	MSE
Normalization	Batch normalization in convolutional layer

Figure 2 shows the architecture for the optimized CNN model.

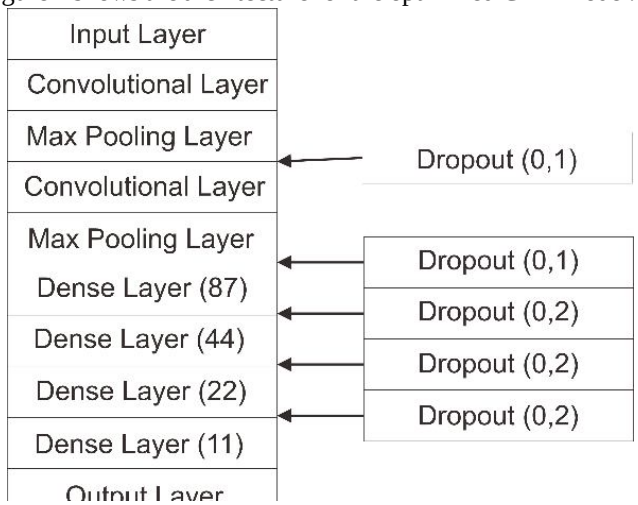


Figure 2: Optimized CNN architecture

4.4 Evaluation

After we finished the CNN optimization process, we conducted evaluation process to compare optimized CNN performance with other machine learning models. Table 15 shows the result of the experiment to compare optimized CNN models with other machine learning models.

Our optimized CNN had the best performance with MSE 1.036 and MAE 0.759. SVR had second place with MSE 1.061 and MAE 0.786. The experiment result also shown that our optimization process was giving positive impact to the CNN performance. The optimization process gave performance increase by 0.294 in MSE and by 0.121 in MAE compared to baseline CNN. In addition, our optimized CNN in this study beat optimized CNN in our previous study by 0.164 in MSE and by 0.071 in MAE.

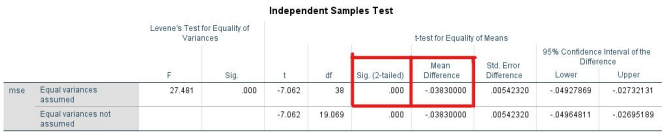
Table 15: Evaluation experiment result

Model	MSE	MAE
Optimized CNN	**1.036**	**0.759**
SVR	1.061	0.786
LR	1.062	0.786
SGDR	1.07	0.79
Optimized CNN [8]	1.20	0.83
Baseline CNN	1.33	0.88
kNNR	1.34	0.89
ANN	1.49	0.95
DTR	1.85	1.04
GPR	2.32	1.25

Table 16: Evaluation experiment result

Run	CNN		SVR	
	MSE	MAE	MSE	MAE
1	1.012	0.759	1.061	0.784
2	1.089	0.776	1.062	0.785
3	1.008	0.747	1.063	0.786
4	1.044	0.772	1.061	0.785
5	1.019	0.749	1.062	0.787
6	1.025	0.756	1.06	0.786
7	1.042	0.756	1.062	0.784
8	1.052	0.761	1.06	0.785
9	1.025	0.757	1.061	0.786
10	1.044	0.76	1.062	0.787
11	0.996	0.746	1.06	0.784
12	1.006	0.742	1.062	0.785
13	1.011	0.75	1.061	0.784
14	0.992	0.748	1.061	0.784
15	1.046	0.759	1.063	0.785
16	0.995	0.746	1.061	0.784
17	0.994	0.748	1.062	0.785
18	1.025	0.752	1.061	0.786
19	1.019	0.747	1.062	0.785
20	1.016	0.75	1.059	0.784
Avg	**1.023**	**0.754**	**1.061**	**0.785**

After we got two best performer models in evaluation experiment. Independent t Test was conducted to prove that the two models has significant difference in movie rating prediction performance. Table 16 shows the sample data for optimized CNN and SVR. We used IBM SPSS Statistics 26 to conduct the independent t Test. After we did normality test, we found that only MSE sample data for both models had normal distribution. Therefore, only MSE data was used for independent t Test.



Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means					
								95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
mse	Equal variances assumed	27.481	.000	-7.062	38	.000	-.03830000	.00542320	-.04927869	-.02732131
	Equal variances not assumed			-7.062	19.069	.000	-.03830000	.00542320	-.04964811	-.02695189

Figure 4: Independent t Test result

Figure 4 shows the independent t Test result. Based on the result, the Sig (2-tailed) value is 0.000, which mean its smaller than α value (0.05). In addition, the mean difference value also shows that optimized CNN has -0.0383 difference compared to SVR. Therefore, we conclude that optimized CNN had significant difference in movie rating predication performance compared to SVR.

5. CONCLUSION

Our study proved that our optimization process by using hyperparameter tuning for one dimensional CNN (1DCNN)had promising result. Our optimized CNN's performance in predicting movie rating beat other machine learning models. The optimized CNN in this study also beat the performance of optimized CNN in our previous study [8].Moreover, features from feature extraction process from the previous study proved again to give positive impact to the movie rating prediction.

We also found several findings during the experiments. Firstly, greater pair number of convolutional layer – max pooling layer does not always give performance boost. This behavior was also found for the number of nodes in dense layer, greater node number in each dense layer does not always give performance boost. For the activation function, ReLU proved to have better performance than other activation functions for regression case like this study.

Dropout as regularization layer proved to have better performance than DropConnect and Spatial Dropout. Therefore, Dropout might be used to reduce overfitting in other regression case that use tabular dataset. Moreover, we found that the most suitable optimizer for model with dropout was Adam. Adam beat other optimizers when we implemented Dropout with 50% probability in our CNN model. For weight regularizer, L2 weight regularizerwith 0.01 penalty value proved to have the best performance in our study.

For loss function, MSE proved to have the best performance in our study. Lastly, the batch normalization experiment proved that the batch normalization implementation on the convolutional layer gave performance boost. Our hyperparameter tuning process might be suitable for other prediction cases which use deep learning model and use tabular dataset as input.

Although CNNoptimization in this study gave promising result, it still has rooms for improvement. The hyperparameter tuning process could be optimized by implementing method such as GridSearchCV, ParameterGrid, ParameterSampler and RandomizedSearchCV. Moreover, the implementation of genetic algorithm or particle swarm also might optimize the hyperparameter tuning process.

REFERENCES

1. R. Parimi and D. Caragea, "Pre-release Box-Office Success Prediction for Motion Pictures," in *International Workshop on Machine Learning and Data Mining in Pattern Recognition. , , , .*, Berlin, Heidelberg, 2013.
2. S. Mundra, A. Dhingra, A. Kapur and D. Joshi, "Prediction of a movie's success using data mining techniques," in *Information and Communication Technology for Intelligent Systems*, Singapore, 2019.
3. "2019 THEME Report," Motion Picture Association of America (MPAA), Washington, D.C., 2019.
4. M. H. Latif and H. Afzal, "Prediction of Movies popularity Using Machine Learning," *IJCSNS International Journal of Computer Science and Network Security, VOL.16 No.8*, pp. 127-131, 2016.
5. S. M. Mudambi and D. Schuff, "What makes a helpful review? A study of customer reviews on Amazon.com," *MIS Quarterly, 34*, pp. 185-200, 2010.
6. S. S. Khopkar and A. G. Nikolaev, "Predicting long-term product ratings based on few early ratings and user," *Electronic Commerce Research and Applications 21*, pp. 38-49, 2017.
7. J. Lorenz, "Universality in movie rating distributions," *The European Physical Journal B 71.2*, pp. 251-258, 2009.
8. R. A. Abarja and A. Wibowo, "Movie Rating Prediction Using Convolutional Neural Network Based on Historical Values," *International Journal of Emerging Trends in Engineering Research*, vol. 8, no. 5, pp. 2156-2164, 2020.
9. X. Ning, L. Yac, X. Wang, B. Benatallah, M. Dong and S. Zhang, "Rating prediction via generative convolutional neural networks based," *Pattern Recognition Letters*, pp. 1-9, 2018.
10. A. Oghina, M. Breuss, M. Tsagkias and M. de Rijke, "Predicting IMDB Movie Ratings Using Social Media," in *Advances in Information Retrieval*, Barcelona, 2012.
11. P.-Y. Hsu, Y.-H. Shen and X.-A. Xie, "Predicting movies user ratings with imdb attributes," in *International Conference on Rough Sets and Knowledge Technology*, Cham, 2014.
12. W. Schmit and S. Wubben, "Predicting ratings for new movie releases from twitter content," in *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, Lisboa, 2015.
13. J. Zhu, Y. Guo, J. Hao, J. Li and D. Chen, "Gaussian Mixture Model based prediction method of movie rating," in *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, Chengdu, 2016.
14. S. Kabinsingha, S. Chindasorn and C. Chantrapornchai, "A movie rating approach and application based on data mining," *International Journal of Engineering and Innovative Technology (IJEIT) Volume 2*, pp. 77-83, 2012.
15. O. B. Fikir, İ. O. Yaz and T. Özyer, "A Movie Rating

Prediction Algorithm with Collaborative Filtering," in *2010 International Conference on Advances in Social Networks Analysis and Mining*, Odense, 2010.

16. J. R. B. D. Rosario, "Development of a Face Recognition System Using Deep Convolutional Neural Network in a Multi-view Vision Environment," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 8, no. 3, pp. 369-374, 2019.
17. F. E. Mendili and Y. E. B. E. Idrissi, "Detection of Video Spam In Social Network Based Neural Network Convolution," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 8, no. 4, pp. 1372-1381, 2019.
18. Y. J. Lim and Y. W. Teh, "Variational Bayesian approach to movie rating prediction," *Proceedings of KDD cup and workshop*, pp. 15-21, 2007.
19. M. Marović, M. Mihoković, M. Mikša, S. Pribil and A. Tus, "Automatic movie ratings prediction using machine learning," in *MIPRO 2011*, Opatija, 2011.
20. D. S. Tan, S. See and T. J. Tiam-Lee, "Automatic rating of movies using an arousal curve extracted from video features," in *International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*, Puerto Princesa, Palawan, 2014.
21. R. Navarathna, P. Lucey, P. Carr, E. Carter, S. Sridharan and I. Matthews, "Predicting movie ratings from audience behaviors," in *IEEE Winter Conference on Applications of Computer Vision*, Steamboat Springs, CO, 2014.
22. S. Basu, "Movie Rating Prediction System Based on Opinion Mining and Artificial Neural Networks," in *International Conference on Advanced Computing Networking and Informatics*, Singapore, 2019.
23. W. R. Bristi, Z. Zaman and N. Sultana, "Predicting IMDb Rating of Movies by Machine Learning Techniques," in *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-5). IEEE.*, Kanpur, 2019.
24. B. Çizmeci and Ş. G. Ögüdücü, "Predicting IMDb ratings of pre-release movies with factorization machines using social media.," in *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, Sarajevo, 2018.
25. S. Basuroy, S. Chatterjee and S. A. Ravid, "How critical are critical reviews? The box office effects of film critics, star power, and budgets," *Journal of marketing, 67(4)*, pp. 103-117, 2003.
26. I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, MIT Press, 2016.
27. 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 15* , pp. 1929-1958, 2014.
28. L. Wan, M. Zeiler, M. Zeiler, Y. LeCun and R. Fergus, "Regularization of Neural Networks using DropConnect," *Proceedings of the 30th International Conference on Machine Learning*, vol. 28, pp. 1058-1066, 2013.
29. J. Tompson, R. Goroshin, A. Jain, Y. LeCun and C. Bregler, "Efficient Object Localization Using Convolutional Networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, Boston, 2015.
30. A. F. M. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.
31. Y.-D. Zhang, C. Pan, J. Sun and C. Tang, "Multiple sclerosis identification by convolutional neural network with dropout and parametric ReLU," *Journal of computational science*, vol. 28, pp. 1-10, 2018.
32. G. E. Dahl, T. N. Sainathy and G. E. Hinton, "IMPROVING DEEP NEURAL NETWORKS FOR LVCSR USING RECTIFIED LINEAR UNITS," in *2013 IEEE international conference on acoustics, speech and signal processing*, 2013.
33. I. Sutskever, J. Martens, G. Dahl and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, Atlanta, 2013.
34. D. P. Kingma and J. L. Ba, "ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION," *arXiv preprint arXiv:1412.6980*, 2014.
35. T. Dozat, "Incorporating nesterov momentum into adam," 2016.
36. M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.
37. J. Duchi, E. Hazan and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of machine learning research*, vol. 12, no. 7, pp. 2121-2159, 2011.
38. H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, S. Chikkerur, D. Liu, M. Wattenberg, A. M. Hrafnkelsson, T. Boulos and J. Kubica, "Ad click prediction: a view from the trenches," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, Chicago, 2013.
39. S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.