Performance of Deep Neural Network for Tabular Data — A Case Study of Loss Cost Prediction in Fire Insurance

Dian Maharani, Hendri Murfi, and Yudi Satria

***Abstract*—The factors that influence fire insurance continue to grow and head to the problem of big data. It is necessary to develop a model to predict the loss cost due to fires by examining the state-of-art models which are adaptable to the big data. One of the models is deep learning, which is an extension of the neural network. This model shows good performances for unstructured data such as image and text. In this paper, we examine the deep learning for loss cost prediction in fire insurance whose training data is tabular or structured data. We use one of the deep learning architectures called deep neural network (DNN), which consists of two or more hidden layers. Our simulation shows that DNN gives quite a similar accuracy to the standard shallow learning of the neural network. It means that deep learning does not improve the performance of the standard shallow learning of neural network for the structured or tabular data of loss cost prediction in fire insurance.**

***Index Terms*—Deep learning, deep neural network, tabular data, structured data, fire insurance, loss cost prediction.**

I. Introduction

Insurance is a service provided by an insurance company to assure the risk of financial losses experienced by a person or group who pay premiums based on an agreement [1]. One of the insurance products is fire insurance, which is an insurance product that covers loss or damage, which is directly caused by fire. In fire insurance, the low frequency of fires and the high severity requires the insurance company to make modeling or analysis of predicted costs of losses due to fire. This aims to identify the risks of policyholders and the amount of insurance coverage that can be borne to replace fire losses more accurately.

Moreover, many factors that cause fires is also challenging for fire insurance companies. Doerr and Cristina stated that climate change and global warming are predicted to increase the frequency and severity of fires in several regions of the world. These factors certainly have an impact on health, community socio-economic activities, and infrastructure [2]. Besides, Kelly, Kleffner, Halek, & Nickerson, stated that the level of population density, crime rate, unemployment rate, age, education level, climate and weather conditions, and demographics of a region can influence the frequency and severity of fires [3]. Moreover, Ye, Wang, Guo, & Li, also state that materials that cause fire, topography, and human activities can influence the frequency and severity of fires [4]. These factors are very diverse and constantly changing. In this case, the amount of data continues to grow and change that brings to the problem of big data.

Big data is a collection of data that has the characteristics of high volume, high velocity, and a wide variety [5]. In addition, L'Heureux, Grolinger, Elyamany, & Capretz also explain that there are several challenges in analyzing big data, including the volume of data (many, size, and scale of data), variety data (structure of data variables, data types, and data interpretation), data velocity (data speed), and data veracity [6]. The challenge of analyzing big data is certainly also a problem for fire insurance companies. On the other hand, it is very important for fire insurance companies to predict the loss cost due to fire for insurance policies. Therefore, it is necessary to develop a model to predict the loss cost due to fires by examining the state-of-art models which are adaptable to the big data.

Deep learning is a state-of-art model in machine learning for prediction. Deep learning is an extension of the standard neural network in term of the number of hidden layers. These hidden layers aim to incorporate the feature selection process into the model. Some hidden layer architectures have been proposed and show better performance for unstructured data. For example, a convolution neural network is a deep learning model which is popular for image data [7]. This model also has applied for other unstructured data types such as textual data and shows a promising performance [8]. Moreover, the online learning of the neural network makes the deep learning adaptable for big data.

In this paper, we examine the deep learning for loss cost prediction in fire insurance whose training data is a tabular data or structured data. We use one of the deep learning architectures called deep neural network (DNN), which consists of two or more hidden layers. This DNN architecture has some parameters and hence called hyperparameter to be optimized. Firstly, we tune the hyperparameter to show the sensitivity of the hyperparameter to the performance of DNN. Therefore, we get the best setting for the hyperparameters. Next, we compare the performance of DNN with that of the standard shallow learning of neural network in term of accuracy. Our simulation shows that DNN gives quite a similar accuracy to the standard shallow learning of the neural network. It means that deep learning does not improve the performance of the standard shallow learning of neural network for the structured or tabular data of loss cost prediction in fire insurance.

The rest of the paper is organized as follows: In Section II, the reviews of related works are presented. Section III

Manuscript received July 22, 2019; revised September 29, 2019. This work was supported in part by This work was supported by the University of Indonesia under grant PIT9 2019.

The authors are with Universitas Indonesia, Indonesia (Corresponding author: Dian maharani; e-mail: dian.maharani71@sci.ui.ac.id, hendri@sci.ui.ac.id, ysatria@sci.ui.ac.id).

describes the research method. Section IV describes the simulation. In Section V, we discuss the implementation program, results, and discussion of the simulations. Finally, we give a conclusion in Section VI.

II. Related Work

DNN is the development of the NN model. In this case, there is more than one hidden layer on the DNN model. Besides, the DNN model is a new model that combines the feature selection process as part of the model. DNN is a very flexible model with a very large number of parameters. This is because of the many hidden layers and neurons in each layer [9]. DNN, as a deep learning model, has an important role in analyzing big data problems. This is because the ability to manage the problem of the data is labeled or not labeled with a very large volume [10]. Moreover, Zhang, Yang, Chen, & Li, also stated that deep learning can handle large data problems, heterogeneous data, real-time data, and can provide good accuracy for processing data with many features [11].

In its application, the DNN model has been widely used in pattern recognition problems for unstructured data such as sound, images, computer vision, and robotics [12]. The use of DNN on structured data, especially problems related to insurance is still rare. One of the DNN studies was conducted by Kuo to predict loss reserves for property insurance and accidents and compare them with stochastic models. The results showed that the DNN model was more accurate than the stochastic model [13]. Moreover, Saputro, Murfi, and Nurrohmah applied the DNN model for classification problem in automobile insurance. The results showed that the accuracy of DNN is slightly better than the standard neural networks in term of normalized-gini [14].

Several studies on the application of the DNN model above did not test the DNN model for analyzing a big data problem related to the prediction of the loss cost due to fire in the insurance sector. Therefore, in this study, we focus on examining DNN for loss cost prediction due to fires in the insurance sector whose training data is a big tabular or structured and comparing the performance of DNN with standard shallow learning of neural network in term of the accuracy.

III. Research Method

In this study, the DNN and NN models were used to predict the loss cost due to fire in the insurance sector.

A. Deep Neural Network (DNN)

Deep Neural Network (DNN) is one of the deep learning models that are part of artificial intelligence inspired by the workings of the human brain. The purpose of this DNN model is to analyze and solve problems as humans do base on their thinking. The DNN model is the development of the NN model, where the selection of features in the DNN model is part of the model and is not run separately. Also, DNN architecture consists of more than one hidden layer. The number of hidden layers defines the depth of the DNN architecture [15]. Mathematically the data flow and learning algorithm of the DNN model is developed from the NN model.

According to Bishop, data processed by the NN model is run with the following algorithm [16];

The data enter to neurons z_j:

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{jo}^{(1)} \quad j = 1 \dots M \tag{1}$$

The data output from neurons z_j:

$$z_j = h(a_j) \tag{2}$$

The data enter to neurons y_k:

$$a_k = \sum_{i=1}^{D} w_{kj}^{(1)} z_j + w_{ko}^{(2)} \quad j = 1 \dots M \tag{3}$$

The data output from neurons y_k:

$$y_k = l(a_k) \tag{4}$$

General form:

$$y_k(x) = l\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{jo}^{(1)}\right) + w_{ko}^{(2)}\right) \tag{5}$$

$$y_k(x) = l\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i\right)\right) \tag{6}$$

$w_{ji}^{(1)}$ and $w_{kj}^{(2)}$: weight parameters,
$w_{jo}^{(1)}$ and $w_{ko}^{(2)}$: bias parameters,
$h(.)$,$I(.)$: activation functions.

In the NN model, the most important problem of learning this model is how to determine the parameters of weight and bias so that the minimum error value is obtained. For example, given training data (training) $\{x_n, t_n\}, n = 1, \dots, N$, where $t_n \in \{-1,1\}$ then the number of errors for the training data can be expressed as a sum of squared error (SSE)

$$E(W) = \sum_{n=1}^{N} E_n(W) = \frac{1}{2}\sum_{n=1}^{N} |y_k(x_n) - t_n|^2 \tag{7}$$

In this case, the objective function of the NN model defined by $E(W)$ above is a non-linear function where $\nabla E(W) = 0$ is not closed. So, the method that can be used to solve the optimization problem is an iterative method such as Stochastic Gradient Descent (SGD), Adam, or Adagrad as a variation of SGD.

In this case, the renewal of the weight value in the output layer can be determined by the following formula;

$$w_{kj}^{(2)(\tau+1)} = w_{kj}^{(2)(\tau)} - \eta \delta_k z_j \tag{8}$$

Whereas for renewal the weight of the hidden layer can be determined by;

$$w_{ji}^{(1)(\tau+1)} = w_{ji}^{(1)(\tau)} - \eta \delta_j x_i \tag{9}$$

where, η is the learning rate parameter, which is a positive scalar that determines the size of the learning stages used to reach the local minimum.

In this study, the input layer DNN architecture is suitable with 350 dimensions. This is because of 350 features affecting the extent of the cost of losses caused by fire. The DNN model architecture used in this study adopts DNN architecture introduced by Korzinkin *et al*., where the DNN model consists of four hidden layers with each neuron 2000,1500,1000,500 and one output neuron. Also, a dropout rate (0.2) layer is applied to each layer dense interlude [17].

The Adagrad optimizer method is also used in this study. The activation function that is used in the hidden layer is 'PReLu' and the linear activation function is used in the output layer. The linear activation function is used in the output layer because the problem in this study is a regression. Furthermore, the hyperparameters used in the Korzinkin *et al*. study will be re-optimized to obtain a higher level of accuracy in predicting the cost of losses in fire insurance. DNN architecture is illustrated in Fig. 1.

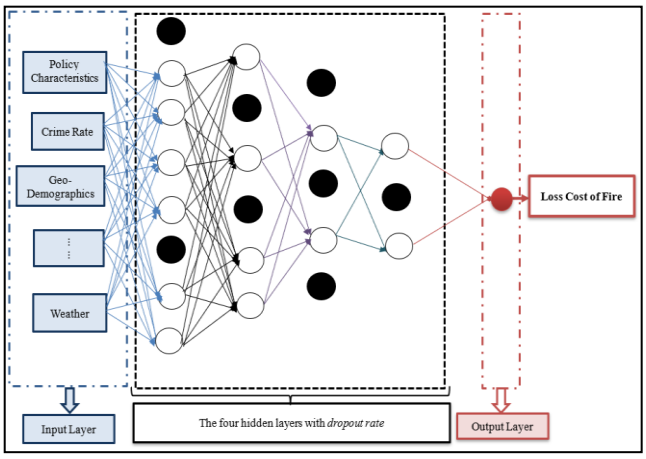

Fig. 1. Architecture Deep Neural Network (DNN) model.

While the standard shallow learning of the NN model initialization architecture that will be optimized in this study is the NN model, which consists of one input layer, one hidden layer, and one output layer. In this case, many neurons in the hidden layer are 100. The activation function that is used in the hidden layer is 'ReLu'. While the linear activation function is used in the output layer. The linear activation function is used in the output layer because the problem in this study is a regression. The method of the optimizer that will be used is Adam. Also, L2 Regularization, dropout rate, and batch normalization will be applied to the hidden layer to overcome overfitting. Furthermore, the NN model will be optimized to get the best NN model in predicting the cost of losses in fire insurance. NN architecture is illustrated in Fig. 2.

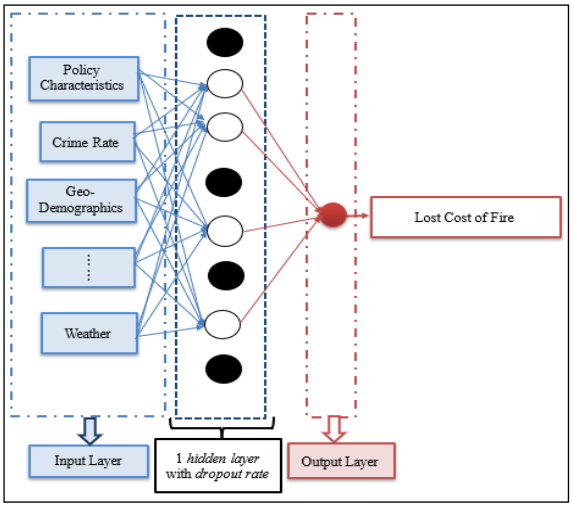

Fig. 2. Architecture Neural Network (NN) model.

B. Network Optimization

In the building the DNN and NN models, it is very important to optimize some parameters that used. This is done to reduce prediction errors from the model. These parameters will be described as follows.

1) *Activation functions*. The activation function is used in DNN and NN models, including linear functions, sigmoid functions, softmax functions, tanh functions, PReLu, and ReLu functions [16]. The activation function is a function that making the layer active and mapping neurons from the input layer to neurons in the output layer.

2) *Regularization weight (L2)*. Based on Goodfellow, *et al*., regularization is a technique to reduce errors in general but not reduce training errors during algorithm learning. In the problem of linear regression, the learning criteria can be modified with "weight decay". In this case, a sum consisting of the average squared error in the training and the criterion $E(W)$ representing the weight that has the norm L^2 *s*maller squares can be minimized [15]. Let $E(W)$be defined as follows;

$$E(W) = \frac{1}{2}\sum_{n=1}^{N}|y_k(x_n) - t_n|^2 + \alpha\|w\|^2 \qquad (10)$$

where α is a previously selected value to control the learning process to use relatively small weights.

3) *Batch-Normalization*. Batch normalization or batch-normalization is a technique to normalize the results of the activation process of the weight renewal value in the hidden layers of the NN and DNN models. This batch-normalization technique standardizes the output values of the calculation process of the activation function to mean, and the unit standard deviation is close to zero. In this case, the batch-normalization technique is carried out to increase the accuracy and speed of the model training process. Also, according to Bjorck, Gomes, Selman, & Weinberger, some of the benefits of applying batch-normalization techniques are the convergent learning process and the generalization of the results of the models for the better. In its application, for a layer in the NN and DNN models with d-dimensions of input $x = (x^{(1)} \ldots x^{(d)})$, each dimension will be normalized by the following formula [18].

$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{Var[x^{(k)}]}} \qquad (11)$$

where expectation and variance values are calculated based on training data.

4) *Dropout*. Based on Hinton, *et al.*, dropout is a technique to overcome the occurrence of overfitting and become one of the strategies to combine exponentially many different neural network architectures efficiently. The dropout technique was carried out by removing noisy neurons randomly during the training model. One of the advantages of the dropout technique is that the dropout can improve the performance of the NN model for image recognition, sound, and document classification problems [19].

In the simplest case, each unit is maintained with the same probability and depends on another unit. Abdallah *et al*.,) recommends a dropout rate between 20%-50% [20]. On the other hand, Hinton *et al*. recommend optimizing dropout rates in the range of 20%-80% [19].

In this research, we tune the hyperparameter to show the sensitivity of the hyperparameter to the performance of the

model. In this case, the process of tunning hyperparameter is carried out such as the number of neuron units used, activation function, percentage dropout rate, α parameter or penalty value of regularization, learning rate for each NN model optimizer method and DNN is carried out in the following order:

1) Initialization of the Architecture of the NN and DNN models
2) Optimizing the NN and DNN model neurons, the best number of the neuron is applied at the next optimization stage.
3) Optimizing the NN and DNN model dropout rate parameters, the best dropout rate is applied at the next optimization stage.
4) Optimizing the parameter α or the penalty value of the regularization of weights in the NN and DNN models, the best α value is applied at the next optimization stage.
5) Optimizing the activation function of the NN and DNN models, the best activation function is used for the optimization of the next stage.
6) Optimizing the learning rate for each optimizer method used in the NN and DNN models.

IV. Simulation

In this study, the simulations for pre-processing data, processing, learning process, and final simulations for evaluation of the models was carried out. In each simulation also carried out several processes that explained below.

A. Pre-processing Data

The data used in this study is secondary data on a liberty mutual insurance company. The data in this study has a data capacity of 1.46 GB consisting of 452061 samples and 300 different variables (features). In general, these variables are features of insurance policy policies, weather, crime rates, and geo-demography. The data used consists of categorical data and numerical data. In this case, the data feature is expressed as var1-var17, which represents the policy variable, crimevar1-crimevar9 represents the crime rate variable, geodemvar1-geodemvar37 as a geo-demography variable, and weathervar1-weathervar236 is a weather variable.

Based on the analysis of the data, it is also known that there are incomplete data (missing value). In this case, the policy feature contains missing data, with a percentage of 61.19%. In the crime feature, there is a missing value with a percentage of 25%. Geodemographic features include incomplete data (missing value) 44.50%. Weather feature there are incomplete data (missing value) of 24.71% of all of the weather features available. In this study, three steps of data processing were carried out.

1) *Handle missing value*. Based on the data distribution, it is known that there are many data, "Not a Number (NaN)" on the data features that loss cost due to fire. Missing values are imputed with the data mode in the corresponding column so that complete data is obtained. In this case, the mode method was chosen to overcome incomplete data because the data in this study consisted of categorical and numerical variables. In the categorical variable, there are data with ordinal and nominal variable types. Therefore, so that the data used remains representative and does not change the meaning of the data, the mode method is used.
2) *Handle categorical data*. In this study, there are data with categorical variables. While data processing with DNN and NN models can be done if the data used is numerical. That way, the first pre-processing stage of the data, the categorical variables in the data are converted into an integer.

Based on Potdar, the NN model that uses the one-hot encoding technique to cope with categorical data has an accuracy of 90%. Therefore, the one-hot encoding technique was used in this study [21]. In this study, the one-hot encoding technique caused an increase in the observation feature column from 300 to 350.

3) *Data standardization*. In this study, the second step of pre-processing data is standardizing data. Based on Jayalaksmi, T., & Santhakumaran, the process of standardizing input data in the NN model is very important to prepare the appropriate data during the training model process. This has effects on the speed of the model training process because each feature is on the same scale the calculation of standardized research data with Z-scores [22].
4) *Splitting data*. The third step of pre-processing data is splitting data. It has become the training and testing data, with a proportion of 80%: 20%. In this case, the training data is separated into two separate data sets. Around 80% of the training data is used for model selection (model selection), and 20% is used as test validation data. This test validation data used to estimate generalizations during or after training in evaluating the performance of DNN models and NN.

B. Processing Data

Processing data in this study used the NN and DNN models. In this case, simulations for the DNN and NN models use the *package of Keras*. The stages of learning include compiling, training, and evaluating the DNN and NN models whose architecture and initialization parameters have been explained in the research method section. In the model evaluation process, the function used is the *Mean Square Error (MSE)*, with calculations:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(c_i - \hat{c}_i)^2 \quad (12)$$

where, c_i = true values ; $\hat{c}_i$ = predict values

In this study, the MSE function used because the problem to be solved is a regression problem.

V. Implementation Program, Results, and Discussion

In this study, several processes of DNN and NN optimization was carried out. The first step is the process of tunning the hyperparameter to show the sensitivity of the hyperparameter to the performance of the model and get the optimal DNN; the second step is the process of tunning the hyperparameter to show the sensitivity of the hyperparameter to the performance of the model and get the optimal NN. The last step is comparing the DNN and NN model based on the lost value of models to predict loss cost in fire insurance.

A. Implementation Program, Result, and Discussion about Optimization DNN Model

In the learning process of the DNN model, optimization of several hyper-parameters models was carried out to obtain the optimal DNN model. Hyper-parameters optimized include the number of neurons, the percentage dropout rate used, the penalty (α) parameter in the regularization technique of weight, the activation function, and the parameter learning rate (η) optimizer method used in the DNN model. Implementation DNN optimization program and the accuracy of the DNN model as a result of optimizing the hyper-parameters of the DNN model are explained below.

1) The accuracy of the DNN model as a result of neuron optimization

The accuracy of the DNN model as a result of the neuron optimization of the DNN model can be seen in the diagram in Fig. 3.

Fig. 3. Accuracy of the DNN model of neuron optimization results.

Based on Fig. 3, it can be seen DNN architecture with type II neurons gives the smallest MSE value. Also, the more neurons used, the more parameters used in the DNN model and the longer the computing time of the DNN model.

2) The accuracy of the DNN model as a result of dropout rate optimization

The accuracy of the DNN model as a result of the dropout rate optimization of the DNN model can be seen in the diagram in Fig. 4.

Fig. 4. Accuracy of the DNN model of dropout rate optimization results.

Based on Fig. 4, it can be seen DNN architecture by applying the dropout rate (0.2) on each hidden layer produces the smallest MSE value. Also, there is no problem of overfitting or underfitting on the DNN model for each percentage of the optimized dropout rate.

3) The accuracy of the DNN model as a result of the penalty (α) parameter of regularization weight optimization

In the implementation, to obtain the optimal DNN model, the parameter penalty (α) of the regularization technique is optimized in the range 10^{-3} to 10^{3}. The accuracy of the DNN model as a result of the parameter penalty (α) of Regularization Weight optimization of the DNN model can be seen in the diagram in Fig. 5.

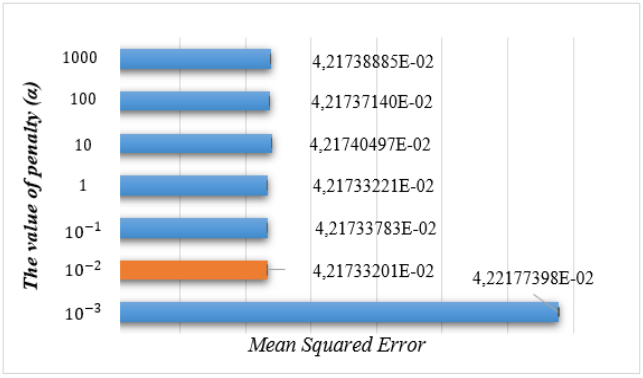

Fig. 5. Accuracy of the DNN model of parameter penalty (α) of regularization weight optimization results.

Based on the diagram in Fig. 5, it can be concluded that the penalty value $\alpha = 10^{-2}$ gives the smallest MSE value for the DNN model while the penalty value $\alpha = 10^{-3}$ gives the largest MSE value for the DNN model.

4) The accuracy of the DNN model as a result of the activation function optimization

The accuracy (MSE) of the DNN model as a result of the optimization of the activation function can be seen in Fig. 6.

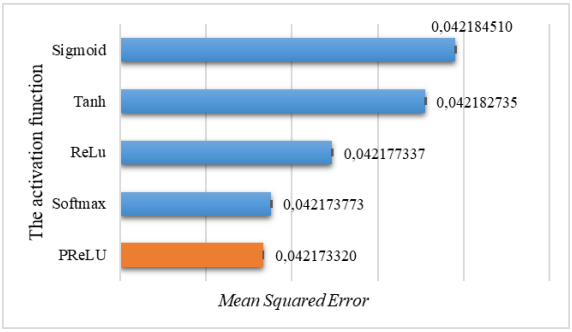

Fig. 6. Accuracy of the DNN of optimization activation function results.

Based on Fig. 6, it can be seen DNN architecture with the PReLU activation function produces the smallest MSE value when compared to other activation functions. Also, the DNN model experiences overfitting when using the softmax activation function. Therefore, the process of optimizing the DNN model will then apply the PReLU activation function.

5) The accuracy (MSE) of the DNN model in the learning rate η Parameter optimization process in the DNN Method optimizers

In this study, the learning rate parameter in the optimizer method used in the DNN model was optimized to obtain the optimal model. Some of the optimizer methods include SGD, Adagrad, and Adam methods. After learning rate optimization, the best optimizer method to optimize the DNN model is so that the MSE value is the smallest, namely the Adam method with the learning rate 10^{-5}. The results of the implementation of Adam's learning rate optimization program can be seen in Fig. 7.

Based on Fig. 7, it can be seen the learning rate value for

Adam gives the smallest MSE value for the DNN model, which is 10^{-5}. Besides that, the greater the learning rate value of Adam method results in the greater the MSE variance value of the DNN model.

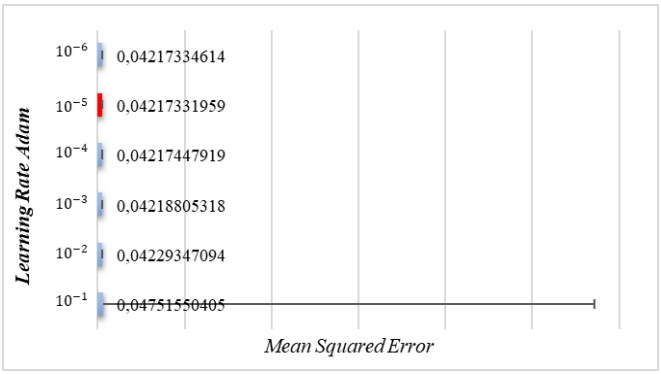

Fig. 7. Accuracy of the DNN model of results of the implementation of Adam's learning rate optimization program.

B. Implementation, Result, and Discussion about Optimization NN Model

In this study, in addition to optimizing several hyper-parameters DNN model, hyper-parameter NN model optimization was also carried out. Several stages in optimizing the NN hyperparameter are carried out with the same steps when optimizing hyperparameters DNN model. The following is an explanation of each step of the NN model hyper-parameter optimization.

1) The accuracy of the NN model as a result of neuron optimization

The accuracy of the NN model as a result of the neuron optimization of the NN model can be seen in the diagram in Fig. 8.

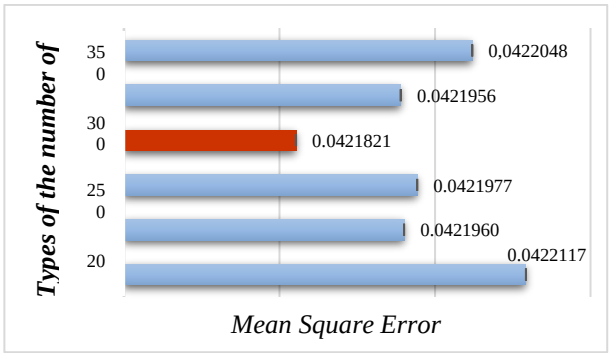

Fig. 8. Accuracy of the NN model of neuron optimization results.

Based on Fig. 8, it can be seen NN architecture with type IV neurons gives the smallest MSE value. Therefore, the optimization of the next NN model will be used in the pattern of type IV (250) neurons.

2) The accuracy of the NN model as a result of dropout rate optimization

The accuracy of the NN model as a result of the dropout rate optimization of the NN model can be seen in the diagram in Fig. 9.

Based on the figure, it can be seen the NN architecture by applying the dropout rate (0.5) on the hidden layer produces the smallest MSE value. Therefore, the optimization of the next NN model will apply a dropout rate (0.5) to its hidden layer.

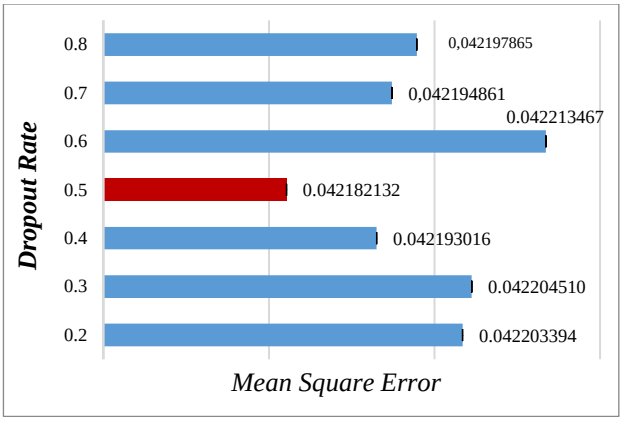

Fig. 9. Accuracy of the NN model of dropout rate optimization results.

3) The accuracy of the NN model as a result of parameter penalty (α) of regularization weight optimization

In the implementation, to obtain the optimal NN model, parameter penalty (α) of the regularization technique is optimized in the range 10^{-3} to 10^{3}. The accuracy of the NN model as a result of parameter penalty (α) of Regularization weight optimization of the NN model can be seen in Fig. 10.

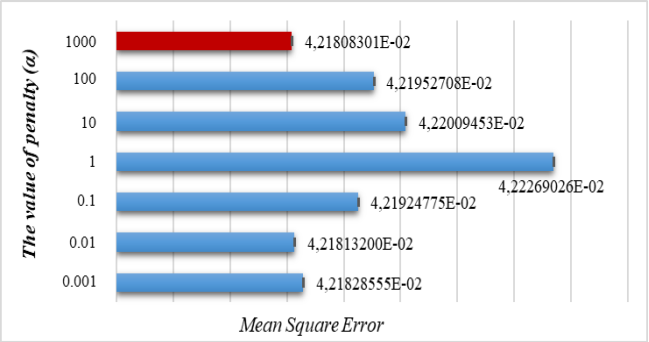

Fig. 10. Accuracy of the NN model of parameter penalty (α) of regularization weight optimization results.

Based on the diagram in Fig. 10, it can be concluded that the penalty value $\alpha = 10^3$ gives the smallest MSE value for the NN model. While the penalty value α=1 gives the largest MSE value for the NN model. Therefore, the optimization of the next NN model uses a penalty (α) for the regularization of 10^3.

4) The accuracy of the NN model as a result of the activation function optimization

The accuracy (MSE) of the NN model as a result of the optimization activation function can be seen in Fig. 11.

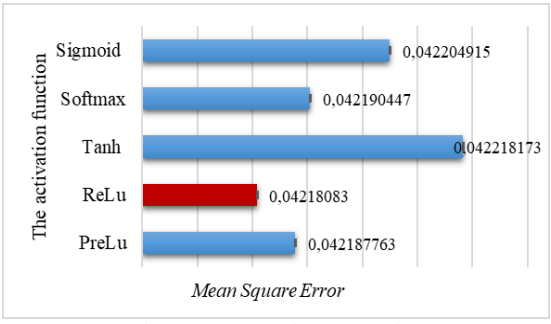

Fig 11. Accuracy of the NN model of activation function optimization results.

Based on Fig. 11, it can be seen the NN architecture with the ReLU activation function produces the smallest MSE value when compared to other activation functions. Therefore, the process of optimizing the NN model will then apply the ReLU activation function.

5) The accuracy (MSE) of the NN model in the learning rate η parameter optimization process in the NN method optimizers

In this study, the learning rate parameter in the optimizer method used in the NN model was optimized to obtain the optimal model. Some of the optimizer methods include SGD, Adagrad, and Adam methods. After learning rate optimization, the best optimizer method to optimize the NN model is so that the MSE value is the smallest, namely the SGD method with the learning rate 10^{-5}. The results of the implementation of the SDG's learning rate optimization program can be seen in Fig. 12.

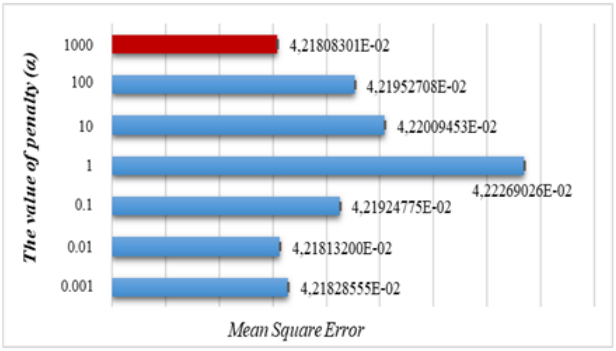

Fig. 12. Accuracy of the NN model as a results of the implementation of SGD's learning rate optimization program.

On the SGD method, the learning rate of 10^{-1} and 10^{-2} cannot know the value of the MSE. This is because the learning rate used for SGD, in this case, skips local minima. Also, by applying the learning rate 10^{-5} to the SGD method, the NN learning process before the tenth iteration has converged and stopped at the sixty-ninth iteration.

C. Optimal DNN Architecture

Based on the results, the optimal DNN is an architecture consisting of one input layer, four hidden layers with many neuron units 2250, 1750,1250, and 750, and one output layer. The PReLU used as an activation function. Applying a dropout rate (0.2) and a penalty regularization of 10^{-2} in each hidden layer DNN model. Applying batch normalization after the second hidden layer DNN model. Using Adam (learning rate of 10^{-5}) as optimizer methods.

From the results of the optimization hyperparameters of the DNN model, there are several changes in hyperparameters DNN optimal with the initial DNN architecture. The differences hyperparameter DNN before and after optimization include the number of neurons in the hidden layer, and the method of optimizers used. The accuracy of the DNN model before the optimization is 0.04217333666 ± 0.704457323e-15. After being optimized, the level of accuracy of the optimal DNN architecture in predicting the cost of losses in fire insurance is reviewed based on its MSE value, which is equal to 0.04217331959 ± 0.63924424e-15. In this case, the results of the implementation process are illustrated in Fig. 13.

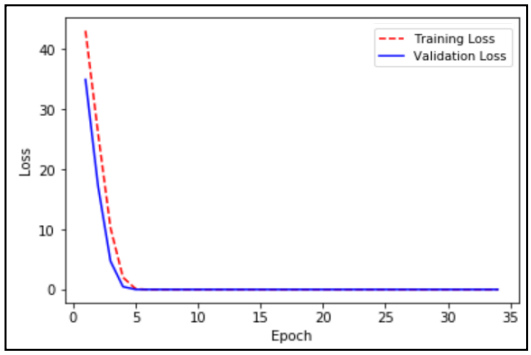

Fig. 13. Graph of optimal DNN model.

Based on the graph in Fig. 13, it can be seen that the curve generated from the learning process of the DNN model shows that the value of loss (MSE) in the training data as well as the validation test decrease to a stable point. In this case, although the value of loss (MSE) of the training data is smaller than one of the validation data, the gap between the two-loss values is very small. This indicates that there are no problems with underfitting or overfitting on the DNN model. Also, in the sixth to 35th iteration, the DNN model has converged, and the loss value on the testing data does not decrease again so that by applying early- stopping the 35th iteration of the learning process has stopped.

From the implementation of the DNN model, we can know the numbers of parameters that used and which are not used on the DNN model. The number of parameters used in the first layer is obtained by multiplying the number of neuron units used in the first hidden layer by the many data features used and then summing with the number of neuron units. The number of batch normalization parameters can be obtained by multiplying many units of neurons in the hidden layer that applies batch normalization with four parameters in batch normalization (gamma weights, beta weights, moving mean, moving variance). In this case, 7000 parameters were obtained consisting of 3500 weighting parameters updated in the backpropagation process, and 3500 weighting parameters that were not updated during the backpropagation process. While many parameters for the second, third, fourth hidden layer, n = 2,3,4 and the output layer can be obtained with the following formula:

$$\text{Number of Parameters}_n = \text{Neuron}_n \times (\text{Neuron}_{(n-1)} + 1), \quad n = 2,3,4,\&(\text{output}) \tag{13}$$

D. Optimal NN Architecture

Based on the results, the optimal NN is an architecture consisting of one input layer, one hidden layer with 250 neuron units, and one output layer. The ReLU is used as an activation function; dropout rate is 0.5 and a penalty regularization of 10^{-3}in each hidden layer NN model. We apply batch normalization after the hidden layer NN model. SGD with a learning rate of 10^{-5}) is as optimizer method.

From the results of the NN model architecture hyperparameter optimization, there are several changes in with the hyperparameters of initial NN architecture. The hyperparameter differences include the number of neurons in the hidden layer, the value of the penalty regularization weight, and the method of optimizers used. The accuracy of the NN model before the optimization is 0.0422116953948 ± 25.8386293113e-10. After being optimized, the level of

accuracy of the NN architecture in predicting the cost of losses in fire insurance is reviewed based on its MSE value, which is equal to 0.0421733518359 ± 0.64079999e-15. In this case, the results of the implementation process are illustrated in Fig. 14.

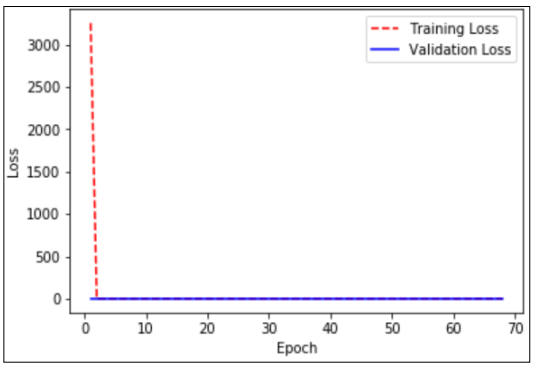

Fig. 14. Graph of optimal NN model.

Based on the graph in Fig. 14, it can be seen that the curves generated from the learning process of the NN model show that in the first iteration, the value of loss (MSE) in the training data is very high, then in the second iteration to 60, the loss value (MSE) in the data training and test validation data decreased to a stable point. In this case, even though the loss (MSE) value in the training data is smaller than the test validation data, but the gap between the two-loss values is very small. This indicates that there are no problems with underfitting or overfitting on the NN model. Also, in the sixth to the 60th iteration, the NN model is convergent, and the loss (MSE) value in the testing data does not decrease again so that by applying early- stopping, in the 60th iteration the learning process has stopped.

From the implementation of the NN model, we can know the numbers of parameters that used and which are not used in the NN model. The number of parameters used in the first hidden layer is obtained by multiplying the number of neuron units used in the first hidden layer by the many data features used and then summing with the number of neuron units. The number of batch normalization parameters can be obtained by multiplying many units of neurons in the hidden layer that applies batch normalization with four parameters in batch normalization (gamma weights, beta weights, moving_mean, moving_variance). In this case, 1000 parameters are obtained, consisting of 500 weighting parameters that are updated in the backpropagation process, and 500 weighting parameters that are not updated during the backpropagation process while many parameters for the output layer can be obtained by formulas as in (13).

E. The Performance of DNN and NN Model

The last step in this research compared the architecture of the DNN model and NN model. The simulation results are presented in Table I.

TABLE I: THE PERFORMANCE OF DNN AND NN MODEL

Model	Akurasi (MSE)	Running Time Program	
		Training	Testing
DNN	0.04217331959 ± 0.63924424e-15	567.842 s	6.163 s
NN	0.04217335183 ± 0. 64079999e-15	330.947 s	4.824 s

Based on Table I above, it can be seen the Deep Neural Network (DNN) model is comparable to the Neural Network (NN) model. Also, the running time program for the Neural Network (NN) model is faster than the Deep Neural Network (DNN) model.

VI. CONCLUSION

Based on the accuracy of the DNN and NN above, it can be seen the DNN model that has been optimized is reviewed based on its MSE value, which is equal to 0.04217331959 ± 0.63924424e-15. The accuracy of the NN model that has been optimized is reviewed based on its MSE value of 0.04217335183 ± 0.64079999e-15. This shows that DNN still gives similar accuracy to the standard shallow learning of the neural network. It means that deep learning can be used to solve big data problems, but does not improve the performance of the standard shallow learning of neural network for the structured or tabular data of loss cost prediction in fire insurance.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

The first author is tasked with collecting data, analyzing data, managing research data, and analyzing results until conclusions are obtained. While the second and third author directs, guides, and provides suggestions for each stage of research. All authors had approved the final version.

ACKNOWLEDGMENT

This work was supported by the University of Indonesia under grant PIT9 2019. Any opinions, findings, and conclusions or recommendations are the authors' and do not necessarily reflect those of the sponsor.

REFERENCES

[1] J. F. Outreville, *Theory and Practice of Insurance*, First ed., Boston: Springer, US., 1998.
[2] S. H. Doerr and C. Santin, "Global trends in wildfire and its impacts: Perceptions versus realities in a changing world," *Phil. Trans. R. Soc. B.*, vol. 371, 2016.
[3] M. Kelly, A. Kleffner, M. Halek, and D. Nickerson, "The role of insurance in reducing the frequency and severity of fire losses," The University of Fraser Valey: Center for Public Safety and Criminal Justice Research, ch. 2, pp. 22-40, 2017.
[4] T. Ye, Y. Wang, Z. Guo, and Y. Li, "Factor contribution to fire occurrence, size, & burn probability in a subtropical coniferous forest in East China," *PLoS One*, vol. 12, no. 2, pp. 1–18, 2017.
[5] M. Beyer and D. Laney, *The Importance of 'Bigdata': A Definition*, Gartner Research: Stamford, CT, USA, Tech. Rep. G00235055, 2012.
[6] A. L'Heureux, K. Grolinger, H. F. Elyamany, and M. A. M. Capretz, "Machine learning with big data: Challenges and approaches," *IEEE Access*, vol. 5, pp. 7776–7797, 2017.
[7] F. Chollet, *Deep Learning with Python*, United States of America: Manning Publication, 2018.
[8] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *Journal of Big Data*, vol. 2, pp. 1–21, 2015.
[9] G. Hinton, L. Deng, D. Yu *et al*., "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, pp. 1–27, 2012.
[10] N. M. Elaraby and M. Elmogy, "Deep learning: Effective tool for big data analytics," *Int. J. Comput. Sci. Eng*., vol. 5, no. 5, pp. 254–262, 2016.

[11] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "A survey on deep learning for big data," *Inf. Fusion*, vol. 42, pp. 146–157, 2018.
[12] V. Sze, S. Member, Y. Chen, S. Member, and T. Yang, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, pp. 2295–2329, 2017.
[13] K. Kuo, "Deep triangle: A deep learning approach to loss reserving," presented at Casualty Loss Reserve Seminar, 2018.
[14] A. R. Saputro, H. Murfi, and S. Nurrohmah, "Analysis of deep neural networks for automobile insurance claim prediction," *Communications in Computer and Information Science*, vol. 1071, pp. 114–123, 2018.
[15] I. Goodfellow, I. B. Yoshua, and C. Aaron, *Deep Learning*, MIT Press, 2016.
[16] C. H. Bishop, *Pattern Recognition and Machine Learning*, Berkeley: Springer, 2006.
[17] M. Korzinkin, A. Aliper, E. Putin *et al.*, "Deep biomarkers of human aging: Application of deep neural networks to biomarker development," *Aging (Albany. NY).*, vol. 8, no. 5, pp. 1021–1033, 2016.
[18] J. Bjorck, C. Gomes, B. Selman, and K. Q. Weinberger, "Understanding batch normalization," in *Proc. 32nd Conference on Neural Information Processing Systems*, no. NeurIPS, 2018.
[19] G. Hinton, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.
[20] Z. S. Abdallah, J. Kamruzzaman, and B. Srinivasan. Effect of Hyper-Parameter Optimization on the Deep Learning Model Proposed for Distributed Attack Detection in Internet of Things Environment. [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1806/1806.07057.pdf
[21] K. Potdar, "A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers," *International Journal of Computer Applications*, pp. 10–13, 2017.
[22] T. Jayalakshmi and A. Santhakumaran, "Statistical Normalization and Back Propagation for Classification," *International Journal of Computer Theory and Engineering*, vol. 3, no. 1, pp. 1–5, 2011.

Dian Maharani received her bachelor degree in mathematics education from Universitas Lampung majoring in 2011. She received the masters' degree in mathematics from Universitas Indonesia in 2019. Her researches are applying machine learning and deep learning.

Hendri Murfi received his bachelor degree in mathematics from Universitas Indonesia, masters in computer science from Universitas Indonesia, and Dr. rer. nat. from TU Berlin, Germany. Currently, he is serving as a lecturer and researcher at Data Science Group at the Department of Mathematics, Universitas Indonesia. His researches are machine learning with applications in topic modeling, sentiment analysis, recommender system, and insurance.

Yudi Satria received his bachelor degree in mathematics from Universitas Indonesia, masters in informatics from Institut Teknologi Bandung, and doctor from Universitas Indonesia. Currently, he is serving as a lecturer and researcher at Data Science Group at the Department of Mathematics, Universitas Indonesia. His researches are machine learning with applications in image processing and text analysis.