Capstone Project

Junil Park

Machine Learning Engineer Nanodegree

Sep. 26, 2020

Definition

Project Overview

Various hyperparameters, such as the size of the kernel constituting the convolutional neural network (CNN), the number of channels, the stride, the number of convolution and max pooling layers, the number of fully connected layers, the number of epochs, the batch size, and the dropout ratio, determine the structure of the CNN. It has a decisive influence on not only determining but also extracting features. In this report, in addition to research on performance optimization based on kernel size, number of channels and stride structure, which have been proven in existing CNN-related papers, changes in CNN structure, max pooling size, dropout ratio, and fully connected network structure in the abstraction stage of CNN. The purpose of this report was to examine how this convolutional neural network affects the final performance. After basic configuration of the CNN, kernel, channel, and stride hyperparameter to have a certain performance through the implementation of a simple CNN, the CNN structure, max pooling size, dropout ratio, and fully connected layer to be observed are set as observation variables to learn. The trend and correlation of test performance changes were observed. The CNN model used in this report was trained and tested using the CIFAR-10 Dataset.

Problem Statement

Multi-layer neural network is a kind of artificial neural network structure, which is a machine learning algorithm implemented by imitating human learning method and is used in various fields such as functional reasoning, pattern recognition, and clustering. A multilayer neural network is a supervised learning-based learning model that has a structure with one or more hidden layers between an input layer and an output layer, and learns using training data consisting of input data and target output. The multilayer neural network learns by adjusting the weight to minimize the error between the output value of the neural network and the target output for the input data using a predetermined rule [1]. Multilayer neural networks have the advantage of being able to solve nonlinear problems and to create desired approximation functions from training

data. However, since it directly operates on the original data without considering the characteristics of the data, it requires a lot of learning data, and accordingly, it takes a lot of learning time and there are disadvantages of overfitting. A new neural network structure called convolutional neural network (CNN) has been proposed to compensate for the disadvantages of such multilayer neural networks. CNN is a machine learning model based on a supervised learning technique with a neural network structure proposed in the 1990s. The CNN is composed of several stages: a convolution layer, a pooling layer, and a fully-connected layer [2]. Among them, the convolution layer and the pooling layer serve to extract features of the input data, and the fully-connected layer serves to classify the features extracted earlier. Unlike other algorithms, the CNN processing process includes an abstraction step for extracting features, so it has the advantage of being able to perform direct computation without a separate pre-processing process for extracting features of input data. Also, because it learns using the features of data, it shows good performance in local feature extraction and classification, and the dropout technique can prevent overfitting problems, so it is widely used in image recognition [3, 4]. When determining the structure of CNN, hyperparameters such as kernel size, stride, and number of channels must be determined. These parameters not only determine the overall structure of the CNN, but also directly affect the performance such as learning time and accuracy. Therefore, in order to obtain the expected result without falling into the overfitting problem, optimization work for the hyperparameter must be preceded. In this regard, various papers and research results can be confirmed, but it is common that there are still no specific rules for the correlation and performance impact of each layer structure for image classification and drop-out parameters. When setting the hyperparameter of, after deciding a lot of parts based on experience or the designer's intuition, it is a situation that requires a lot of trial and error. Looking at the CNN structure that actually shows good performance, in the LeNet-5 structure, the size of all kernels was set equal to 5, but Alexnet set the size of the kernel to 11, 5, 3, and ZFNet set the size of the kernel to 7, 5, 3. As a result, it can be seen that the hyperparameter was set according to the designer's intuition [2, 5, 6].

Metrics

In this experiment, the structure of the hyperparameter such as the channel and stride of the CNN abstraction stage is fixed to a specific value, and the batch size, the structure of the CNN layer, the structure of the fully connected layer, and the dropout ratio are

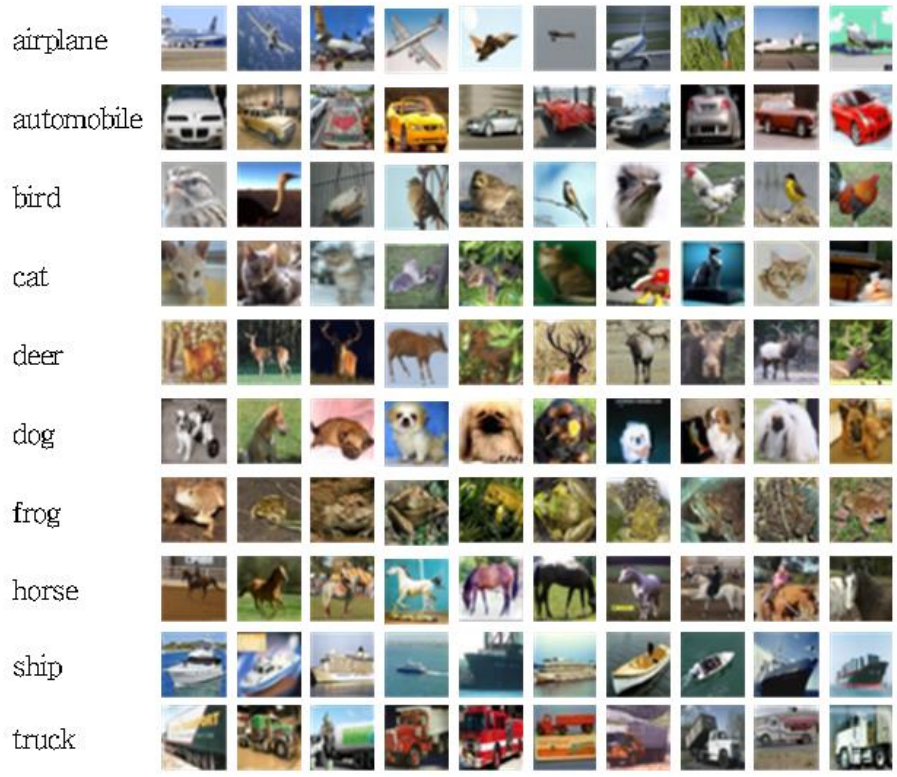
set as variables, based on the training accuracy of the neural network. Simulation was carried out by updating the variables.

Analysis

Data Exploration

CIFAR-10 and CIFAR-100 are sub-sets of 80,000,000 small image data sets, which were collected by Alex Krizhevsky, Vinod Nair and Geoffrey Hinton. The CIFAR-10 Dataset contains a total of 60000 32x32 color images of 10 classes, and consists of 6000 images per class. 50000 training images and 10000 test images are included. This dataset is divided into 5 training batches and 1 test batch, and there are 10000 images in each batch. The test batch contains exactly 1000 images randomly selected for each class. The training batch contains the remaining images in random order, but some training batches may contain more images than other classes. Here is a sample of 10 random images of each, as well as the class of the data set.

Exploratory Visualization

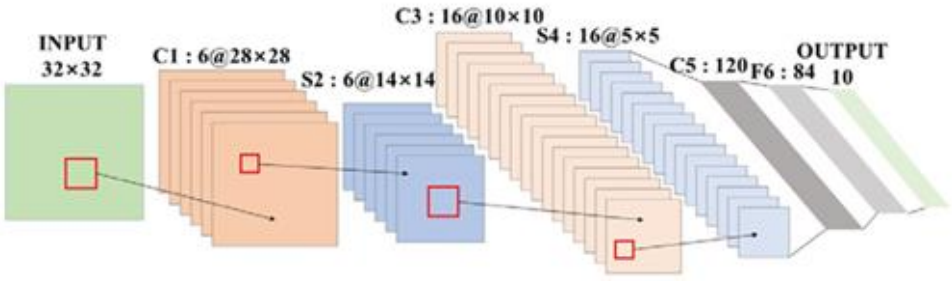


Each class is completely mutually exclusive, and there is no overlap between car and truck labels. "Automotive" includes sedans, SUVs, etc. "Truck" includes only big trucks and not pickup trucks. The implementation downloads the CIFAR-10 data set for Python. It is necessary to consider the logic that minimizes the memory consumption of the computer as the data set is batch processed. The CIFAR-10 data set consists of five batches: data_batch_1 and data_batch_2, and each batch contains labels and images from one of the following ten classes.

Algorithms and Techniques

Convolutional Neural Network

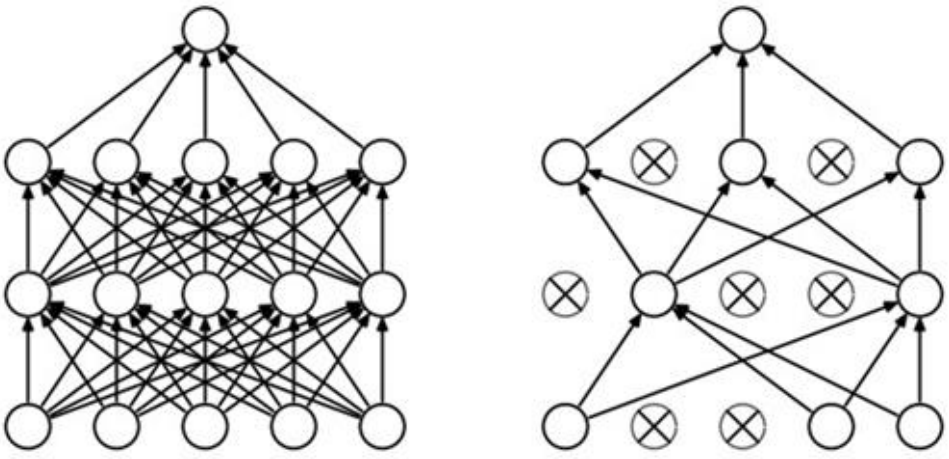
CNN is a type of multi-layer neural network and shows good performance in the field of pattern recognition, so it is mainly used to classify and recognize data such as images and texts. A general multilayer neural network has a structure consisting of an input layer, a hidden layer, and an output layer, so that the output is calculated by directly calculating the original data, whereas the CNN consists of extracting features of the original data and classifying the extracted features. Therefore, it does not operate directly on the original data, but extracts the features of the original data and then operates [7, 8]. In general, CNN consists of a convolution layer, a pooling layer, a fully connected layer, and an output layer. The convolution layer and pooling layer serve to extract features of the original data, and the fully connected layer serves as a classifier to classify features. Since a general multilayer neural network performs calculations for all nodes, the number of parameters to be trained is large, but CNN performs convolution and pooling operations when extracting features, so the size of the extracted features is smaller than that of the original data and shares weight Therefore, it has the advantage that the number of parameters to be learned is small. Figure 1 shows the structure of LeNet-5, a kind of CNN [2].



C1, C3, and C5 in the figure above play a convolution operation on the data coming into the convolution layer using a kernel filter of size 5×5 and extracting features of the data coming through the input. Layers S2 and S4 are pooling layers, and generally reduce the dimension using the max pooling method. Convolution and pooling are repeated several times to extract features of data, and a classifier is placed behind the layer where data extraction is completed to complete the CNN structure. The F6 layer is a fully-connected layer that has the same structure as a general multilayer neural network and serves to classify the features extracted in the feature extraction step [9].

dropout ratio

One of the CNN hyperparameters to be considered in this report is the dropout ratio. A deep neural net with many hyperparameters is a very complex machine learning algorithm, but overfitting caused by various causes can be a serious problem that degrades the performance of the neural net. In addition, as the neural net becomes more complex, the learning speed becomes very slow, so it is not easy to cope with overfitting by combining various complex large networks. To effectively cope with this problem, we can think of a method of randomly removing several nodes in the neural network while training the model. This has the effect of preventing excessive interference between units in the neural network. Through this method, the overfitting phenomenon can be reduced and the performance of the supervised neural network can be improved [3].



(a) Standard Neural Net (b) After applying dropout.

In order to observe the change in prediction accuracy according to the dropout ratio change, the test cases were classified in the following manner.

epoch	conv_ksize	conv_strides	pool_ksize	pool_strides	conv_num_outputs	cnn	fc	fc_out	batch_size	dropout(%)
20	5,5	1,1	2,2	2,2	16	1	1	256	64	10
20	5,5	1,1	2,2	2,2	16	1	1	256	64	20
20	5,5	1,1	2,2	2,2	16	1	1	256	64	30
20	5,5	1,1	2,2	2,2	16	1	1	256	64	40
20	5,5	1,1	2,2	2,2	16	1	1	256	64	50
20	5,5	1,1	2,2	2,2	16	1	1	256	64	60
20	5,5	1,1	2,2	2,2	16	1	1	256	64	70
20	5,5	1,1	2,2	2,2	16	1	1	256	64	80

Here, hyperparameters such as number of epochs for training, kernel size, stride, max-pooling size and max-pooling stride of CNN neural network, number of CNN nodes, CNN layer, fully connected layer, fully connected layer node number and training batch size Is fixed to the same value. The dropout ratio was changed from 10% to a maximum of 80% to observe the change in the prediction accuracy of learning and testing.

Benchmark

When performing neural network training through CNN, the simplest method to determine the structure of CNN is the cross-validation [10] technique, which selects the structure with the best performance by performing a neural network of various structures. However, cross-validation can be said to be a limited method because the best performance is guaranteed only within a few artificially selected structures among the numerous conditions that can constitute a CNN. The most common strategy used for hyperparameter optimization is the so-called grid search method, in which a specific range is set for each parameter and the number of all cases that can be combined within it is substituted. The disadvantage of this grid search method is that the time and effort for calculation increase exponentially depending on the number of parameters to be optimized and the desired search level [11]. Recently, there is a research result that the random search [12] method, which performs randomly selecting hyperparameters within the desired search range, shows better results in finding a model with computational cost or accuracy than grid search. However, when setting the optimal CNN structure, the grid search or random search method is not a method of referring to the previous evaluation result when evaluating a new hyperparameter set. Recently, when performing hyperparameter optimization, Bayesian optimization is also widely used [13, 14, 15], and Bayesian optimization is a method of generating a probability model M based on the previous evaluation result of the objective function f. In order to implement Bayesian optimization, the Gaussian process model (probability model M) and Spearmint [13] technique using the Sequential Model-based Algorithm Configuration (SMAC) based on the random forest of the Gaussian process are most often used. According to the research results of [16], it is reported that Bayesian

optimization has poor predictive performance and high computational cost in a CNN structure involving high-dimensional and complex hyperparameters. In the study of [13], a Bayesian optimization technique based on a Gaussian process was used to optimize hyperparameters such as the learning rate of CNN, epoch, and initial weight of the convolution and fully-connected layers. Many of the hyperparameters in this study have continuous values and are affected by the normalization of the data, but have little correlation with the structure of the CNN itself. [17, 18, 19] also conducted optimization studies on the continuous hyperparameter values of deep neural networks in a similar manner, and in [19, 20, 21, 22, 23, 24] studies, unlike previous studies, Applied techniques that can automatically update hyperparameter values are proposed, such as adjusting the learning rate or reducing the weight value to improve the execution speed of backpropagation. In addition, early stopping [25, 26] may be used when the accuracy of the artificial neural network is no longer improved as learning progresses or the error rate increases as the epoch for learning progresses. [27] In the report, an effective technique that can be used to initialize the weights of the convolution and fully-connected layers is proposed, and the Evolutionary algorithm is the most widely used in automating the structure of learning algorithms of artificial neural networks such as CNN It can be called an algorithm. [28] The genetic algorithm used in the study is used to optimize the size and number of filters in the convolution layer. It has a structure consisting of three convolution layers and one fully-connected layer, and hyperparameters such as layer depth and pooling size are optimized. The accuracy is not as high as 75% because it is not in the state. The genetic algorithm has a disadvantage that the cost for calculation is quite high, because all cases (population) involved in the genetic algorithm are composed of individual CNN objects, and each object must undergo a learning, correction, and verification process. In the study of [29], optimization studies were conducted only for fully-connected layers of artificial neural networks, but a regularization technique is used to automatically set the number of units of the corresponding fully-connected layer. In recent years, attention has been focused on the structural design-related parts of deep learning, and [30] applied reinforcement learning and RNN to the study to search for the optimal structure of neural networks, showing quite impressive results. In [31], it was proposed to use Evolution of Augmenting Topologies (NEAT) based on CoDeepNEAT to determine the shape of each layer of neural network and its hyperparameters. In [32], a genetic algorithm is used to design a complex CNN structure, and each of the above experiments used a method of simultaneously performing more than 250 variance tasks to find the optimal structure. Reinforcement learning based on Q-learning [33] was utilized as a method to search for an optimal structure for only one layer at a time, and

the depth of the corresponding layer uses a value preset by the researcher. However, the results obtained through reinforcement learning required considerable system resources and a long time of effort. [34] In the study, we used a method of visualizing and monitoring these neural network structure optimization tasks, and the speed was improved by reducing the stride and kernel size of the first layer of the convolution. However, the problems inherent in the CNN structure used in the study or issues in progress had a disadvantage that a researcher with specialized knowledge must directly analyze. In addition, humans had to be directly involved in the selection and selection of a new CNN structure. Most of the existing hyperparameter optimization or neural network model selection is a method of verifying various models in order to maximize the accuracy of the validation set. [35] In the study, deconvnet was proposed, which visualizes images learned through neural networks, measures and monitors accuracy, and suggests a new method to reduce the error rate of CNNs. This is a method that can be automatically set through. However, there is a disadvantage in that it cannot be applied to datasets other than image data because optimization is performed through visualization of the learned image. In this report, rather than a viewpoint for optimizing the CNN structure and hyperparameters targeted in these studies, the values for each hyperparameter presented as a study target are gradually changed to a grid search method within a certain range, respectively. By observing the change in neural network accuracy, we intend to use it as the basis for designing and optimizing artificial neural networks for new datasets and research subjects in the future.

Methodology

Data Preprocessing

image data normalization

Implement the data normalize function to get the image data x, convert it into a normalized numpy array in the range 0 to 1, and construct the return object in the same form as x.

one-hot encoding

Implement one_hot_encode function for data pre-processing. The input x is a class(Label) list, has a function to return a label list as a one-hot encoded numpy array, and performs one-hot encoding so that the range of values is from 0 to 9.

Implementation

Implement convolution and max pooling layer

The neural network finally implemented needs input (image) data, label (class) changed by one-hot encoding, and dropout setting value. TF Placeholder,'x' is initialized and returned according to the image data structure input through the neural_net_image_input function, and the batch size is not set so that the system automatically sets it at runtime. The remaining 'y' and 'keep_prob' variables are also initialized in the same way through the neural_net_label_input and neural_net_keep_prob_input functions. Convolution layers are known as effective neural networks for image data classification, and to implement the representative structure of CNN, the convolution layer and max pooling functions are implemented together through the conv2d_maxpool function. To briefly explain the details, weight and bias are generated using the kernel size, output number, and input data type, and the code that finally applies the CNN is generated using the kernel size and stride data. To apply max pooling to CNN, refer to the pool kernel size and pool stride size of the input parameter. These parameters are parts that need to be customized when applying the model in the future, and are not subject to consideration in this report, so a fixed value should be used when proceeding with each verification case.

Implement a fully-connected layer

I implemented the flatten function to change the dimension of the tensor given as an input parameter from a 4-D tensor to a 2-D tensor. The object returned from the function becomes a batch size*flatten image size array. I implemented a function to apply a fully connected layer to the tensor given as an input parameter in the fully_conn function. The returned object has a batch size*num_outputs array. By implementing the output function, I implemented the logic so that the tensor of the fully connected layer can be finally output.

Model creation through training

Implement the conv_net function to create a convolutional neural network model, which takes a series of images x and prints the log. All of the layer creation functions created above are used to finally construct this model.

a. apply convolution and max pool layers

b. apply a flatten layer

c. apply fully connected layers

d. apply an output layer

e. return the output

In addition, the related logic within the function was applied to reflect the dropout ratio (keep_prob) to maximize model accuracy by minimizing vanishing gradient issues.

Refinement

For training progress, parameters such as epoch number, batch_size, and keep_probability (dropout ratio) are defined in advance.

a. epoch = 40

b. batch_size = 512

c. keep probability = 0.9

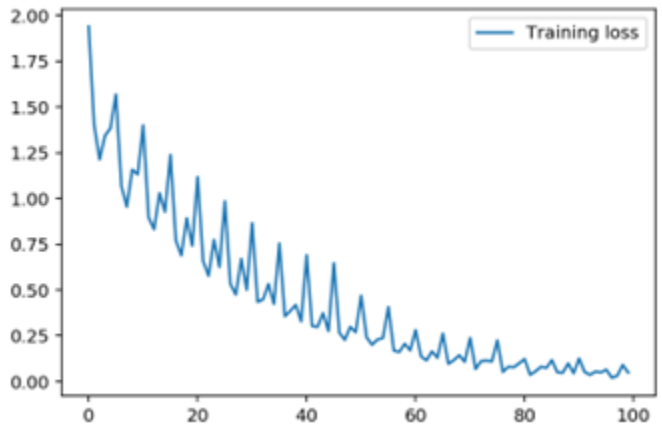
Results

Model Evaluation and Validation

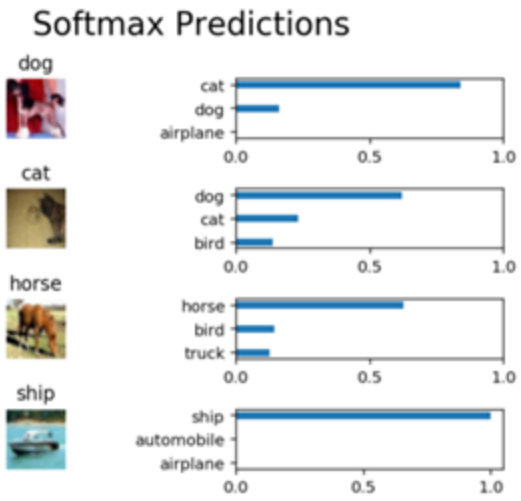
Dropout-ratio change (10 ~ 80%) test

(1) dropout ratio 10%

It can be seen that the change trend of training loss according to the progress of learning changes as follows, and the prediction accuracy for the training dataset is confirmed to be about 61.96%.

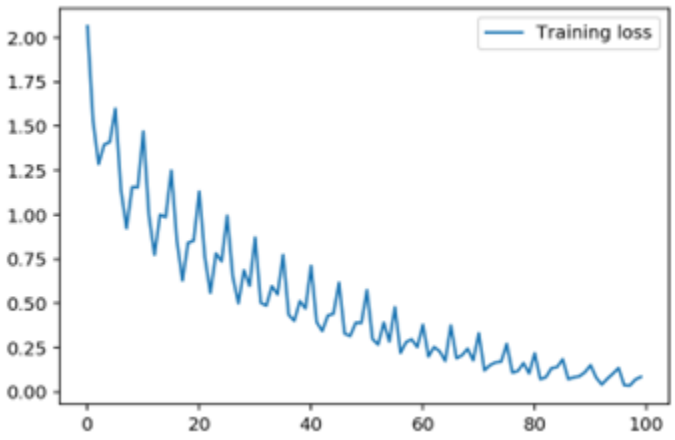


As a result of applying the test dataset to the final training model, the accuracy is confirmed to be about 61.86%, and it can be seen that an error occurs in the classification of cat and dog image data when the test is performed on 4 data through random sampling as shown below.

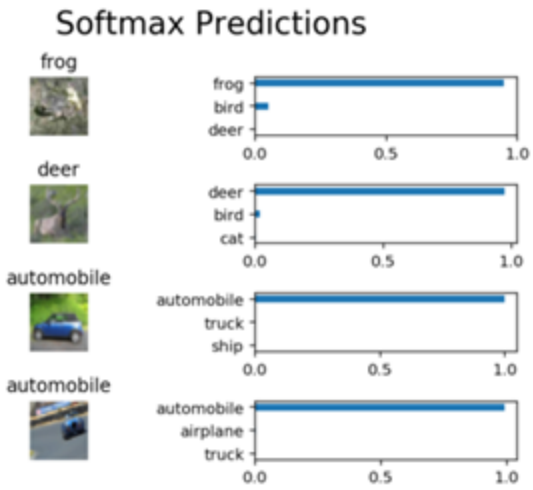


(2) dropout ratio 20%

It can be seen that the change trend of training loss according to the progress of learning changes as follows, and the prediction accuracy for the training dataset is confirmed to be about 62.94%.

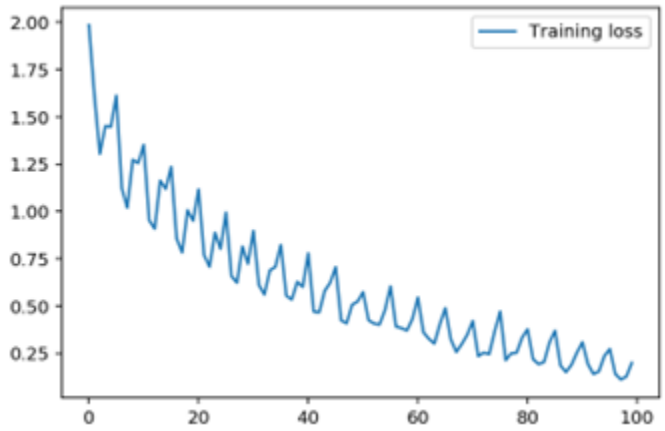


As a result of applying the test data to the final learning model, accuracy is found to be about 62.41%, and it can be seen that image data such as frog, deer, and automobile are predicted without error during the test of four data through random sampling as shown below.

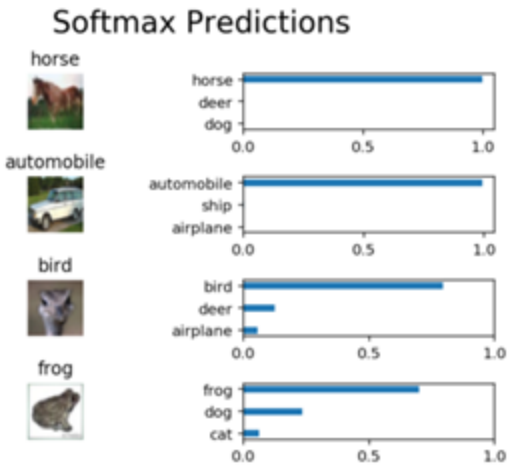


(3) dropout ratio 30%

It can be seen that the change trend of training loss according to the progress of learning changes as follows, and the prediction accuracy for the training dataset is confirmed to be about 63.06%.

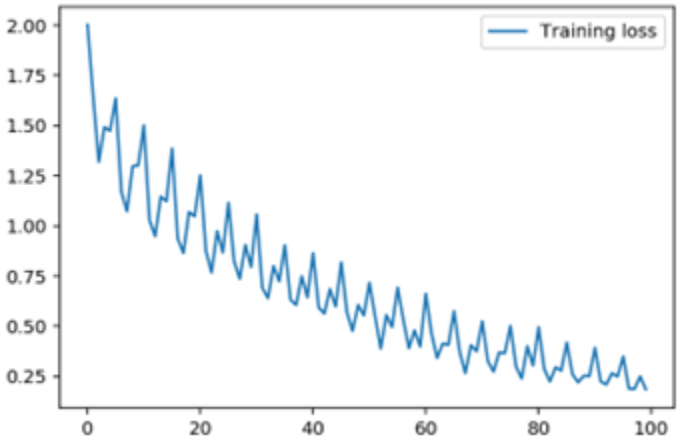


As a result of applying the test data to the final learning model, accuracy is found to be about 63.69%, and it can be seen that image data such as horse, automatic, bird, and fog are predicted without error when testing four data through random sampling as shown below.

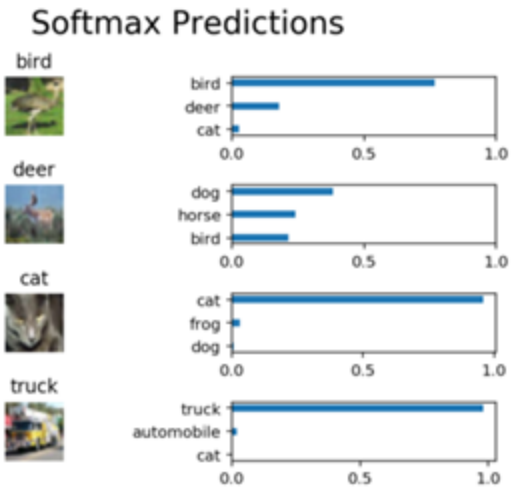


(4) dropout ratio 40%

It can be seen that the change trend of training loss according to the progress of learning changes as follows, and the prediction accuracy for the training dataset is confirmed to be about 63.04%.

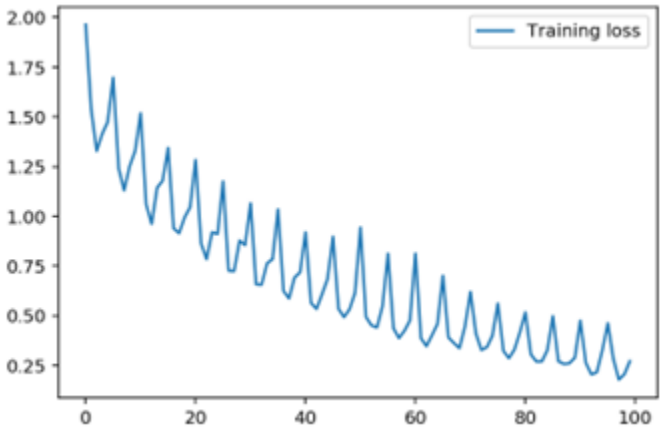


As a result of applying the test data to the final learning model, the accuracy was found to be about 63.18%. During the test of four data through random sampling as shown below, it was predicted that image data such as bird, cat, and truck were error-free, but prediction error occurred for deer.

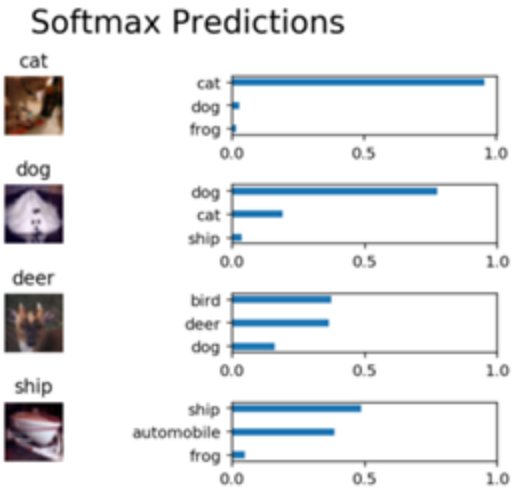


(5) dropout ratio 50%

It can be seen that the change trend of training loss according to the progress of learning changes as follows, and the prediction accuracy for the training dataset is confirmed to be about 64.9%.

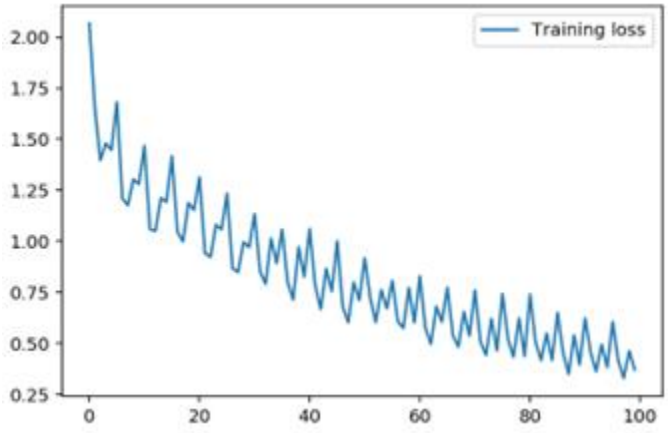


As a result of applying the test data to the final learning model, the accuracy was found to be about 65.39% and it can be seen that the image data such as cat, dog, and ship were predicted without errors during the test of the four data through random sampling as shown below, but the prediction error for deer occurred.

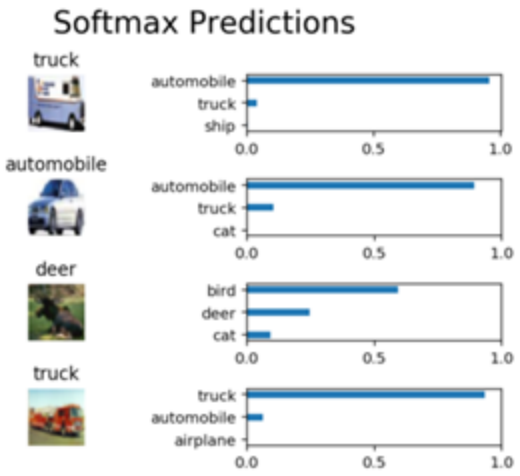


(6) dropout ratio 60%

It can be seen that the change trend of training loss according to the progress of learning changes as follows, and the prediction accuracy for the training dataset is confirmed to be about 65.22%.

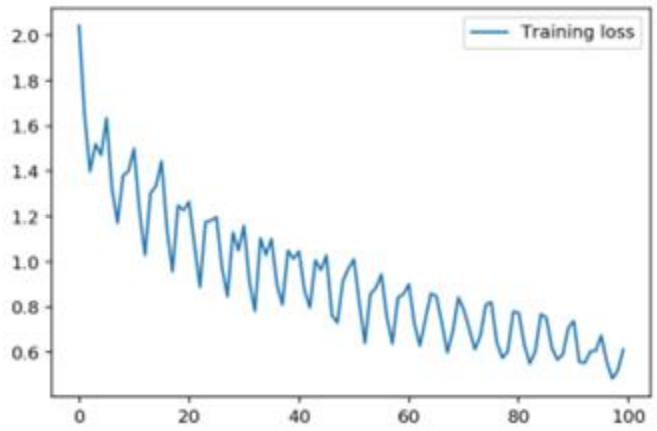


As a result of applying the test data to the final learning model, the accuracy was found to be about 64.67%. During the test of four data through random sampling as shown below, it was predicted that the automobile image data was error-free, but the prediction error occurred for data such as truck and deer.

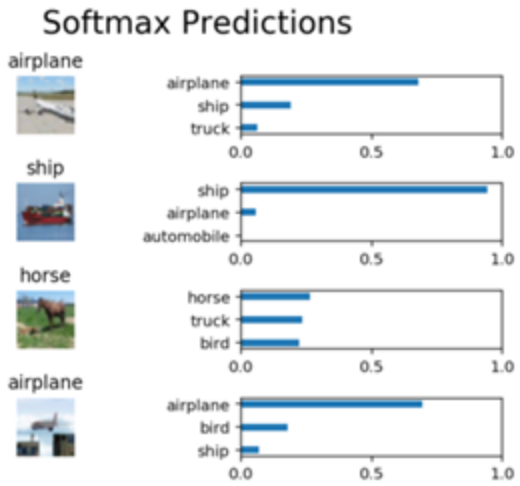


(7) dropout ratio 70%

It can be seen that the change trend of training loss according to the progress of learning changes as follows, and the prediction accuracy for the training dataset is confirmed to be about 62.88%.

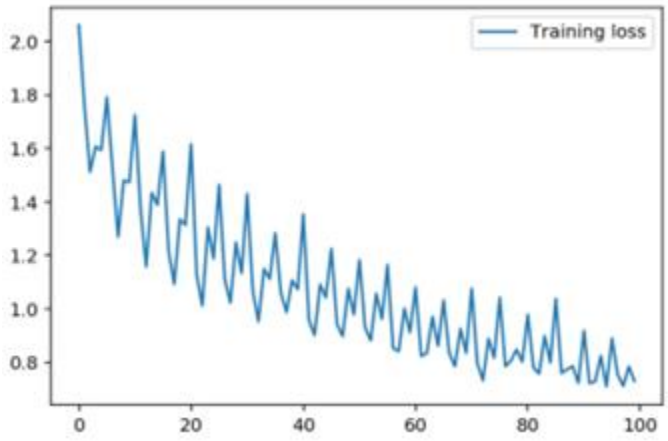


As a result of applying the test data to the final learning model, the accuracy is found to be about 63.14%, and it can be seen that the image data such as airplane, ship, and hose were predicted without error during the test of four data through random sampling as shown below.

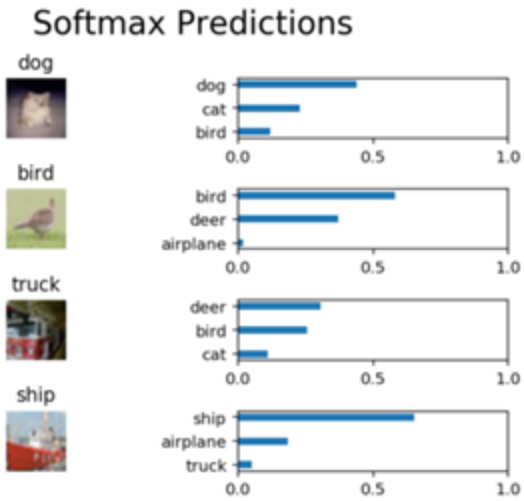


(8) dropout ratio 80%

It can be seen that the change trend of training loss according to the progress of learning changes as follows, and the prediction accuracy for the training dataset is confirmed to be about 61.92%.



As a result of applying the test data to the final learning model, the accuracy is about 62.36%. During the test of four data through random sampling, it can be seen that the prediction error for the truck image occurred without any error.

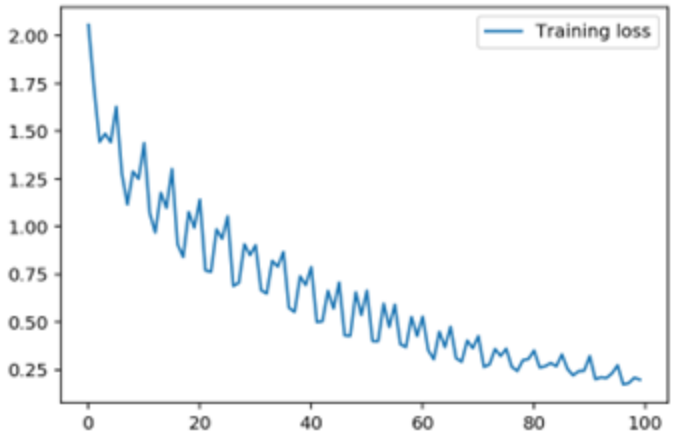


Batch size change (128 ~ 2048) test

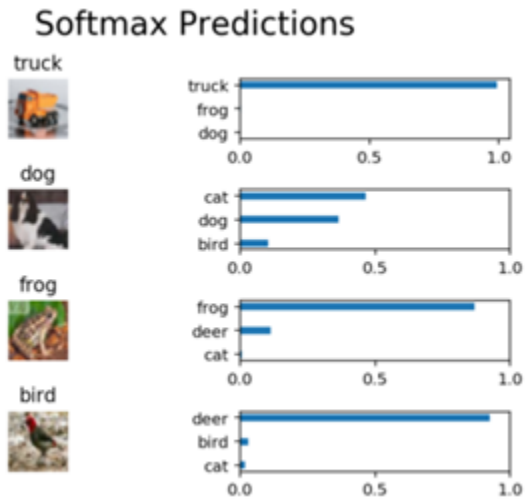
The CNN hyperparameter to be considered in this report is the change in prediction performance according to the change in the mini batch size of the dataset used when training the neural net. In general, a small batch size has a disadvantage in that it takes a lot of time for model training, and as the size of the batch size increases, it has the advantage of shortening the time, but it is known that the prediction performance decreases and the corresponding system computational performance is required a lot. In this report, I will examine the changes in prediction performance according to the batch size change, targeting the images of the CIFAR10 dataset, and examine the verification and improvement of the existing theory. The batch size used for verification is set to 128, 256, 512, 1024, and 2048, respectively, and the results of learning and testing are considered.

(1) batch size 128

It can be seen that the change trend of training loss according to the progress of learning changes as follows, and the prediction accuracy for the training dataset is confirmed to be about 64%.

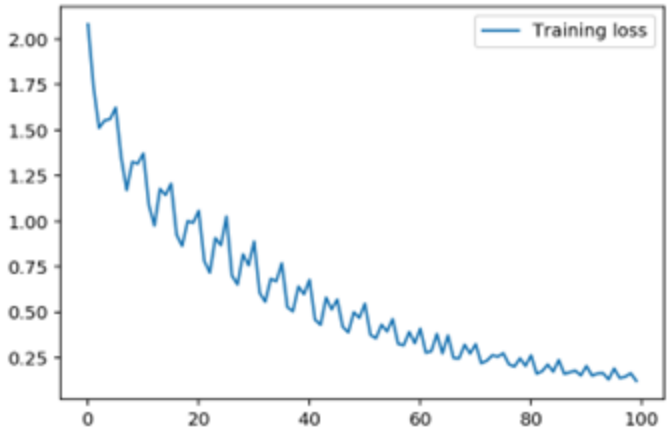


As a result of applying the test data to the final learning model, the accuracy is about 64.1%. During the test of four data through random sampling, it can be seen that the prediction error for dog and bird occurred even though it was predicted without error for image data such as trucks and frogs.

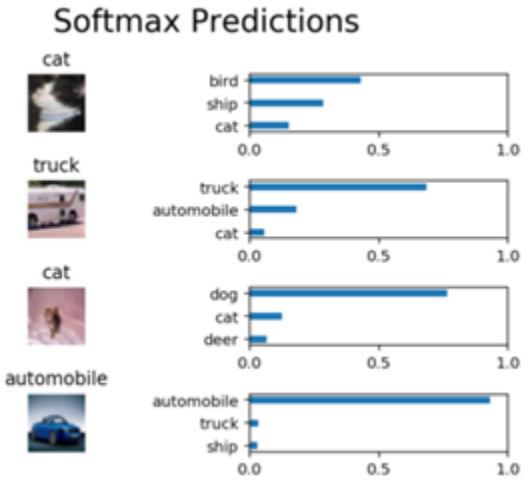


(2) batch size 256

It can be seen that the change trend of training loss according to the progress of learning changes as follows, and the prediction accuracy for the training dataset is confirmed to be about 65.06%.

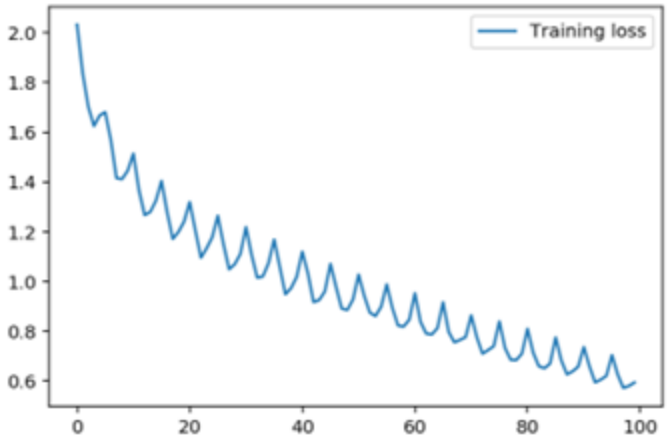


As a result of applying the test data to the final learning model, the accuracy is about 65.53%. During the test of four data through random sampling, it can be seen that the prediction error for cat occurred even though the prediction data such as truck and automobile were predicted without error.

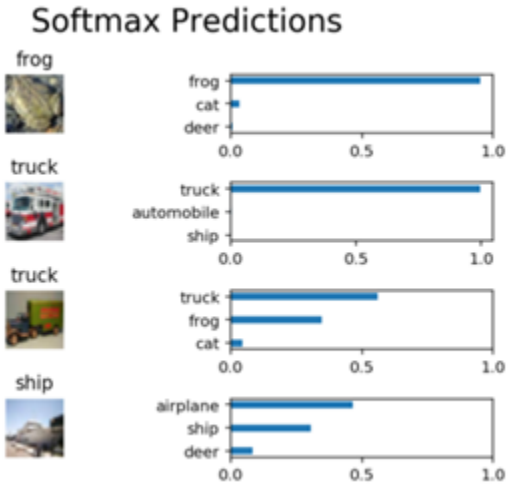


(3) batch size 512

It can be seen that the change of training loss according to the progress of learning changes as follows, and the prediction accuracy for the training dataset is confirmed to be about 63.68%.

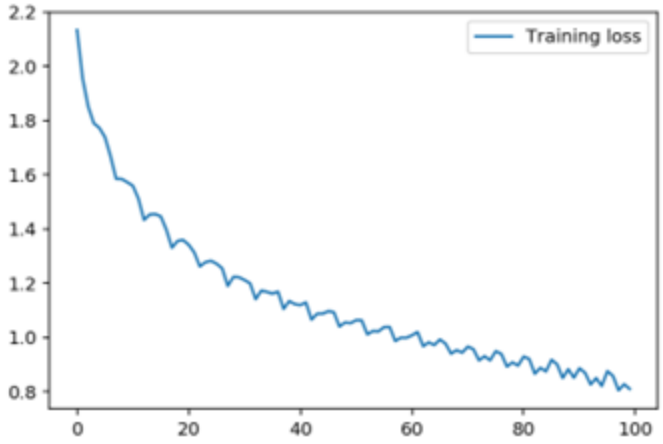


As a result of applying the test data to the final learning model, the accuracy is about 64.06%, and it can be seen that the image data such as frog and truck were predicted without error during the test of the four data through random sampling as shown below, but a prediction error occurred for the ship.

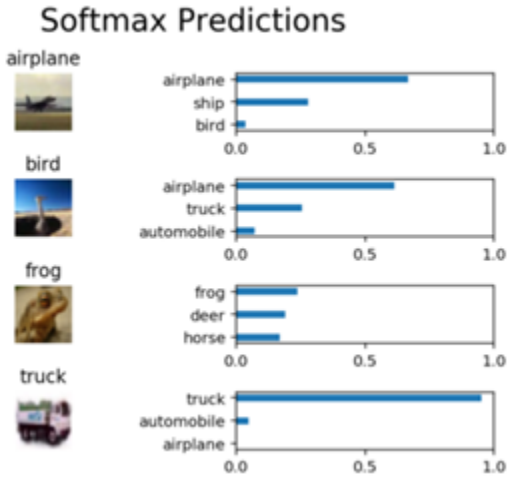


(4) batch size 1024

It can be seen that the trend of changes in training loss according to the progress of learning changes as follows, and the prediction accuracy for the training dataset is confirmed to be about 63.24%.

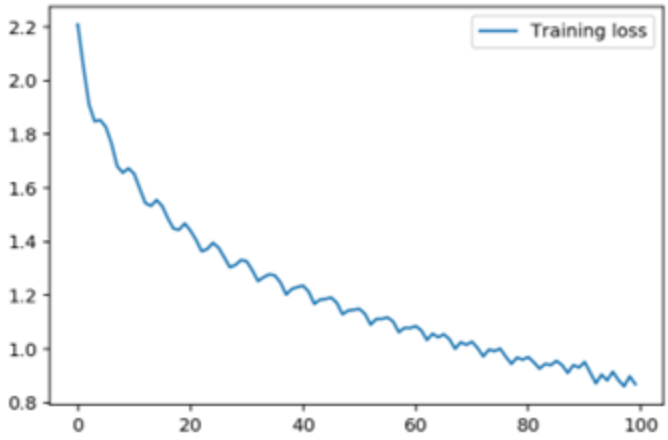


As a result of applying the test data to the final learning model, the accuracy is found to be about 63.06%. During the test of four data through random sampling as shown below, the prediction data of airplane, fog, and truck were predicted without errors, but the prediction error for bird was found.

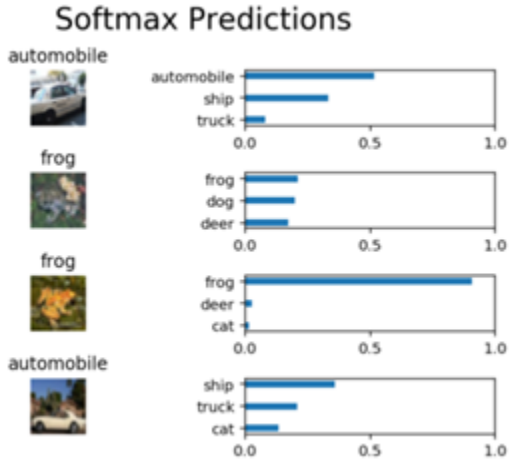


(5) batch size 2048

It can be seen that the change trend of training loss according to the progress of learning changes as follows, and the prediction accuracy for the training dataset is confirmed to be about 60.68%.



As a result of applying the test data to the final learning model, the accuracy is found to be about 60.41%, and it can be seen that the prediction error for automobile occurred even though image data such as frog was predicted without error during the test of four data through random sampling as shown below.

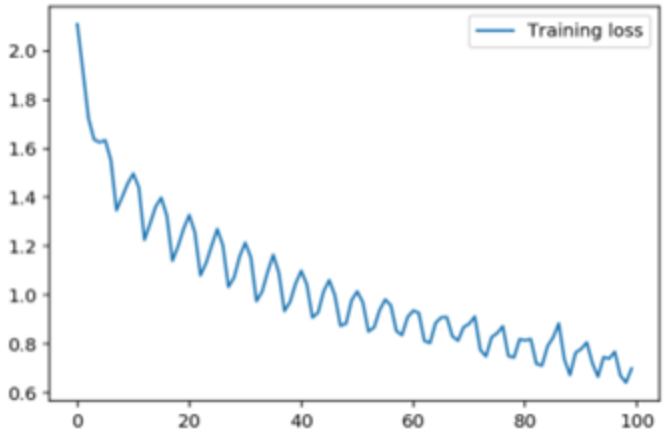


Convolutional layer change (2 ~ 6) test

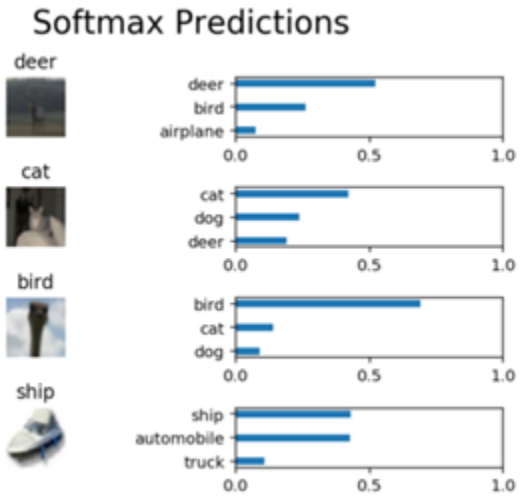
In a simple structured CNN, I would like to observe and analyze the change in accuracy according to the CNN structure change through changes in the number of convolutional layers and nodes.

(1) 16, 32 node 2 convolutional layer

It can be seen that the change trend of training loss according to the progress of learning changes as follows, and the prediction accuracy for the training dataset is confirmed to be about 65.3%.

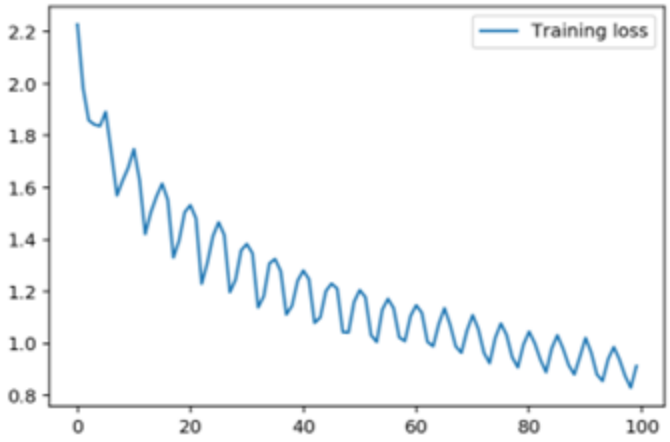


As a result of applying test data to the final learning model, the accuracy is about 66.13%. During the test of four data through random sampling, it can be seen that the image data such as der, cat, bird, and ship were predicted without error.

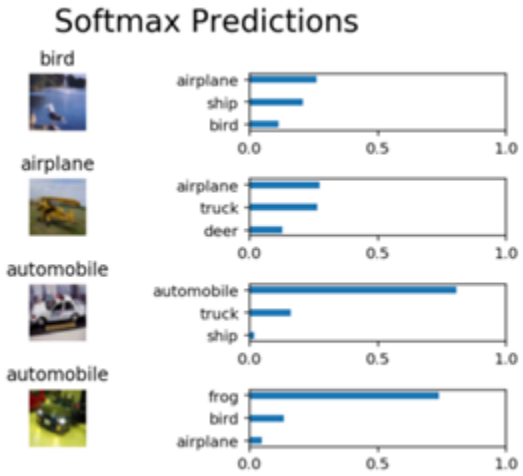


(2) 16, 32, 64 node 3 convolutional layer

It can be seen that the change trend of training loss according to the progress of learning changes as follows, and the prediction accuracy for the training dataset is confirmed to be about 62.34%.

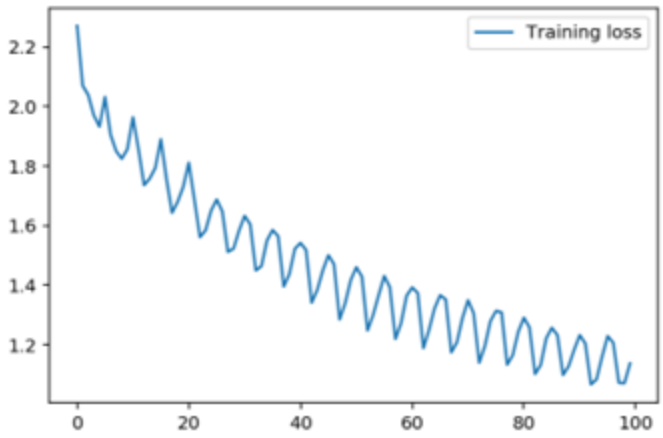


As a result of applying the test data to the final learning model, the accuracy is found to be about 60.41%, and it can be seen that the prediction error for bird, automatic, etc. was made without error when testing four data through random sampling as shown below.

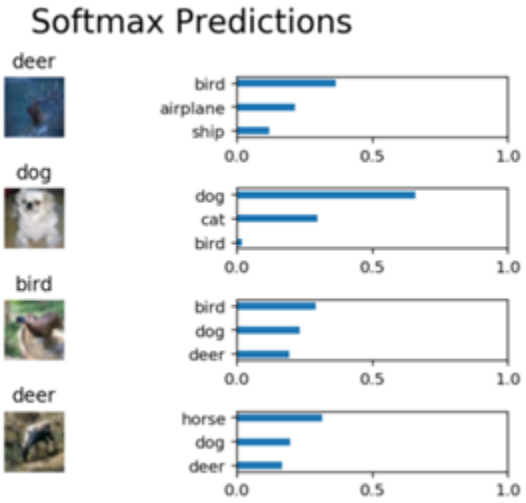


(3) 16, 32, 64, 128 node 4 convolutional layer

It can be seen that the trend of changes in training loss according to the progress of learning changes as follows, and the prediction accuracy for the training dataset is confirmed to be about 54.68%.

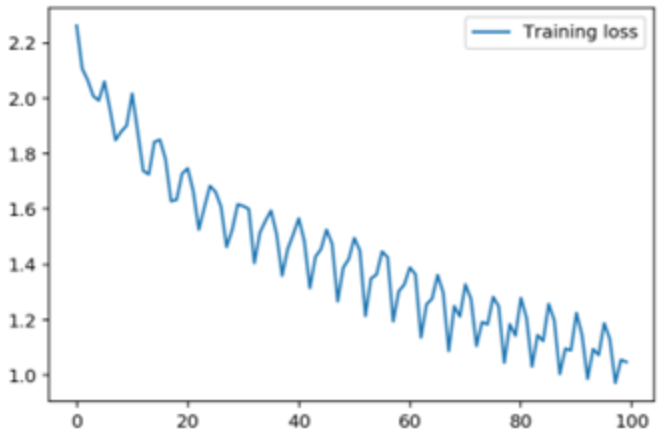


As a result of applying the test data to the final learning model, the accuracy is about 54.72%, and it can be seen that the image data such as dog, bird, etc. were predicted without error during the test of the four data through random sampling as shown below, but a prediction error occurred for deer.

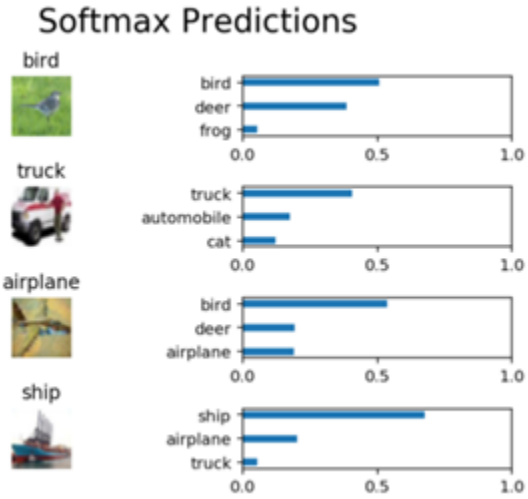


(4) 16, 32, 64, 128, 256 node 5 convolutional layer

It can be seen that the change of training loss according to the progress of learning changes as follows, and the prediction accuracy for the training dataset is confirmed to be about 55.7%.

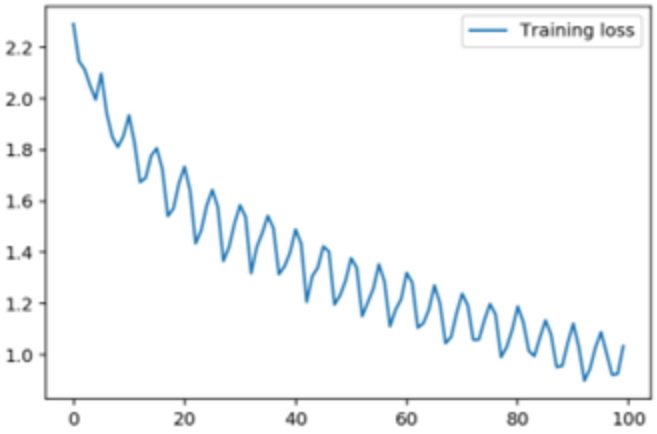


As a result of applying test data to the final learning model, the accuracy is about 54.49%. During the test of four data through random sampling, it can be seen that the prediction error for the airplane occurred even though the prediction data such as bird, truck, and ship were predicted without error.

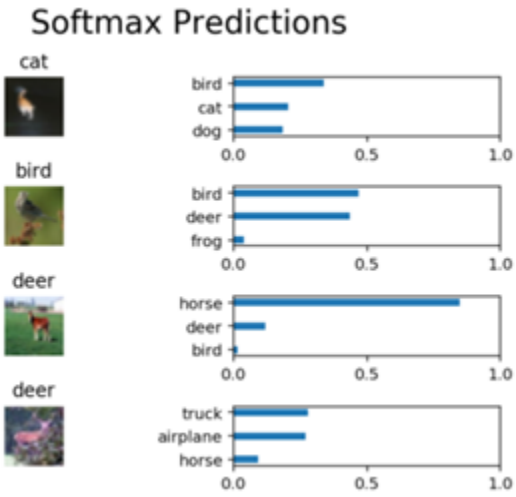


(5) 16, 32, 64, 128, 256, 512 node 6 convolutional layer

It can be seen that the change trend of training loss according to the progress of learning changes as follows, and the prediction accuracy for the training dataset is confirmed to be about 53.16%.



As a result of applying the test data to the final learning model, the accuracy is about 54.05%, and it can be seen that the prediction error for cat, der, etc. occurred even though the prediction data such as bird was predicted without error during the test of the four data through random sampling as shown below.

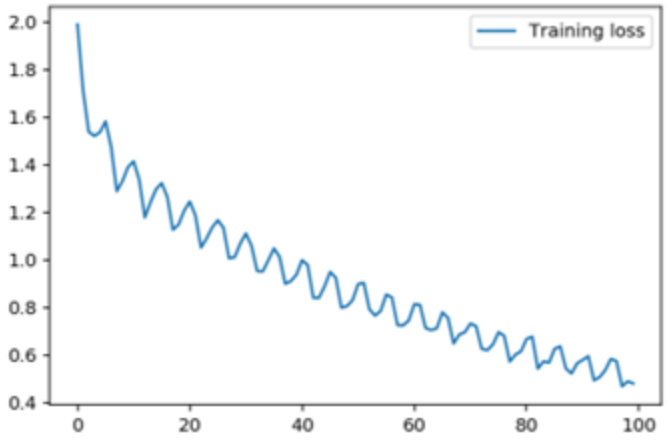


Fully connected layer change (1 ~ 5) test

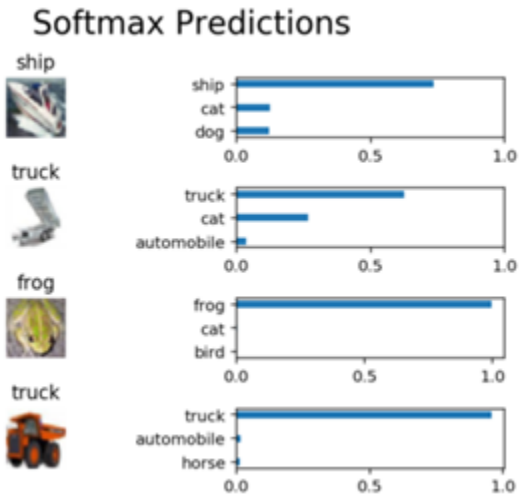
In a simple structured CNN, I would like to observe and analyze the change in neural network accuracy through changes in the number of nodes and fully connected layers connected to the convolution layer and the output layer.

(1) 512 node fully connected layer

It can be seen that the change trend of training loss according to the progress of learning changes as follows, and the prediction accuracy for the training dataset is confirmed to be about 68.04%.

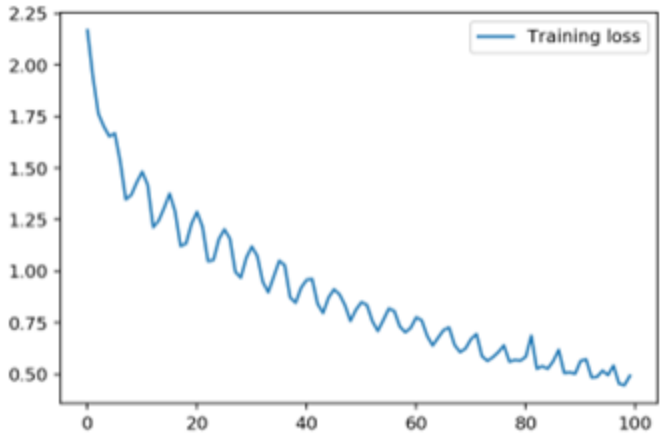


As a result of applying the test data to the final learning model, the accuracy is about 67.8%. During the test of four data through random sampling, it can be seen that the image data such as ship, frog, and truck were predicted without error.

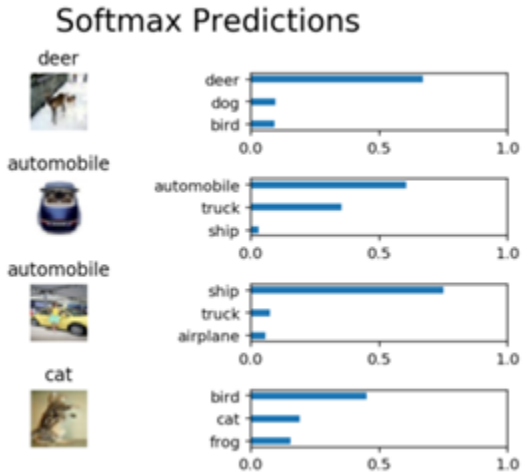


(2) 512, 256 node 2 fully connected layer

It can be seen that the change trend of training loss according to the progress of learning changes as follows, and the prediction accuracy for the training dataset is confirmed to be about 67.2%.

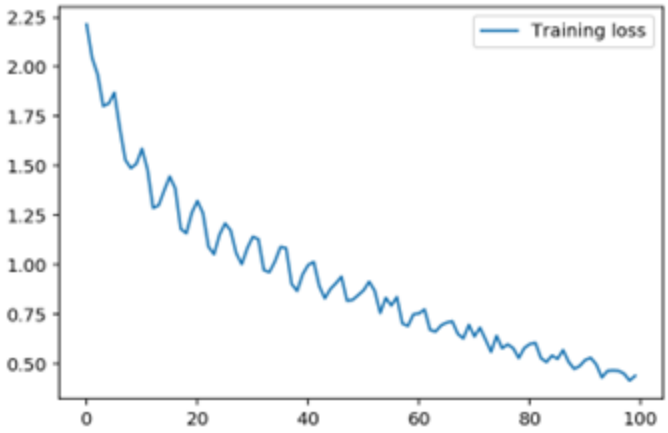


As a result of applying the test dataset to the final training model, the accuracy is confirmed to be about 67.5%. When the test for 4 data through random sampling is performed as follows, image data such as deer were predicted without error, but you can see an error for automobile and cat.

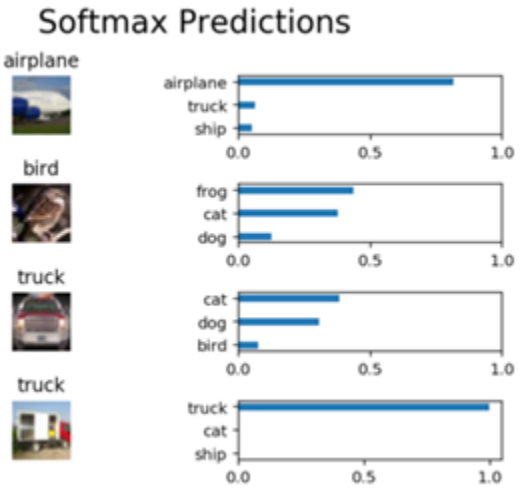


(3) 512, 256, 128 node 3 fully connected layer

It can be seen that the change trend of training loss according to the progress of learning changes as follows, and the prediction accuracy for the training dataset is confirmed to be about 67.64%.

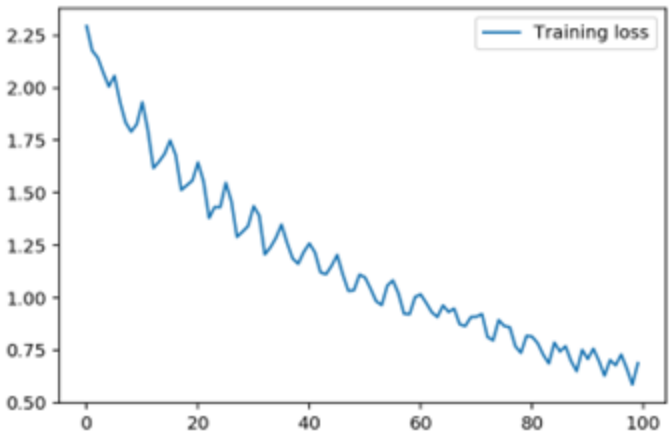


As a result of applying the test data to the final learning model, the accuracy is confirmed to be about 66.31%, and the image data such as airplane were predicted without error during the test of four data through random sampling as shown below, but errors occurred for birds and trucks.

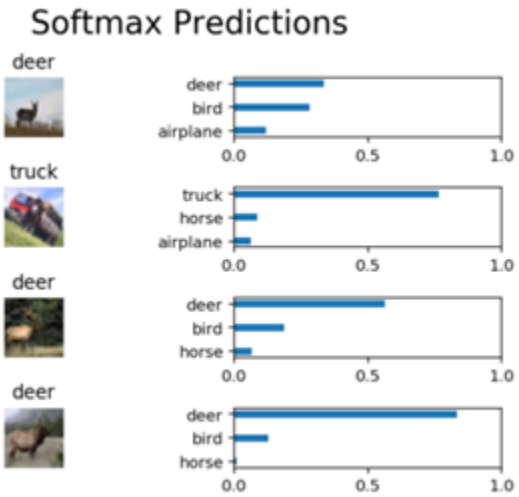


(4) 512, 256, 128, 64 node 4 fully connected layer

It can be seen that the change trend of training loss according to the progress of learning changes as follows, and the prediction accuracy for the training dataset is confirmed to be about 61.46%.

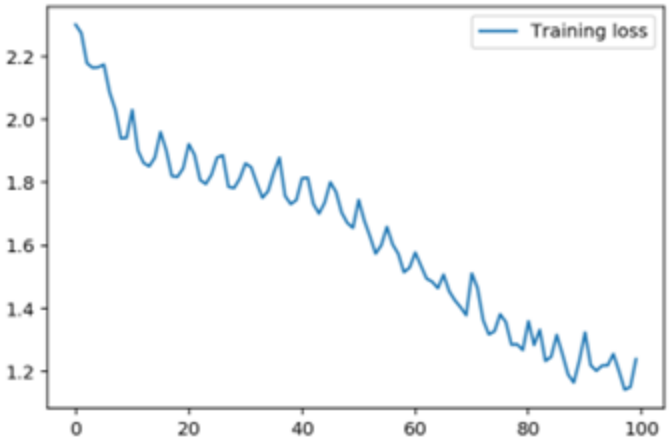


As a result of applying test data to the final learning model, the accuracy is about 61.79%. During the test of four data through random sampling as shown below, it can be seen that the image data such as der, truck, etc. were predicted without error.

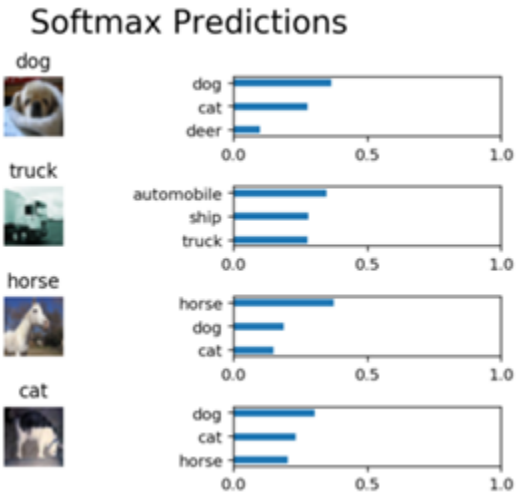


(5) 512, 256, 128, 64, 32 node 5 fully connected layer

It can be seen that the change trend of training loss according to the progress of learning changes as follows, and the prediction accuracy for the training dataset is confirmed to be about 48.34%.



As a result of applying the test data to the final learning model, the accuracy is about 48.82%. In conducting the test on four data through random sampling as shown below, it can be seen that the prediction error for the truck and the cat occurred even though the image data such as dogs and horses were predicted without errors.



Justification

In this report, a simple structure of CNN is implemented using Python-based Tensorflow, and CIFAR-10 image set is trained on the neural network, according to changes in dropout ratio, batch size, CNN layer structure, and fully connected layer structure. I tried to observe the change in CNN's image learning performance and interpret its meaning. Basically, it is impossible for humans to grasp the meaning of the learning result through the neural network in the form of a black box, or to grasp the final result through any calculation or interpretation. Therefore, after setting some hyperparameters as fixed constants, I change the values of each parameter to be

observed for optimization, and examine the results of how the final accuracy changes, thereby suggesting how the parameter affects the neural net. It was attempted to observe and interpret the phenomenon using the method.

Conclusion

Free-Form Visualization

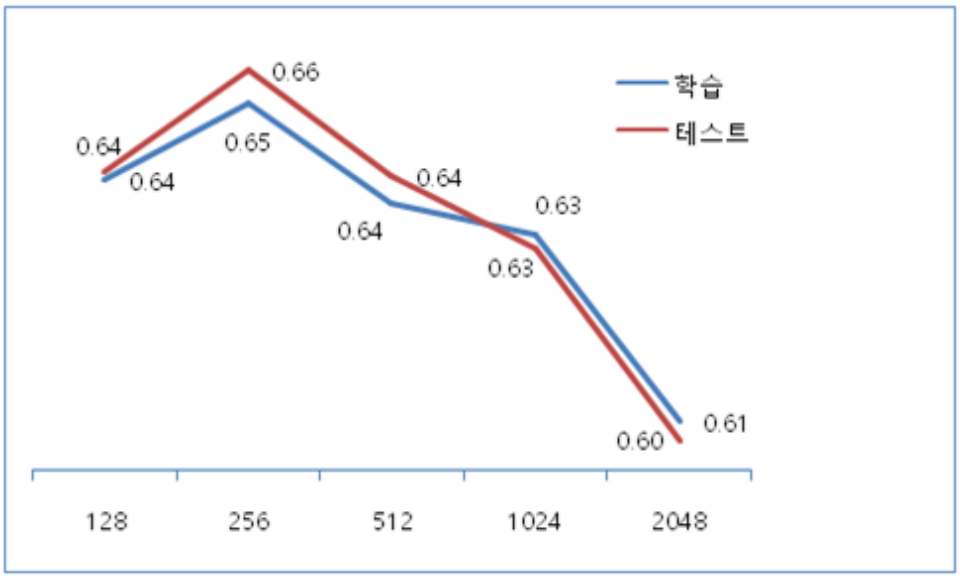
Performance change according to dropout ratio change

As a result of verification through the simple CNN implemented for this report, it can be seen that the test accuracy according to the dropout ratio change gradually decreases after the maximum value of about 65.4% is formed at the dropout ratio level of 50-60%.

epoch	conv_ksize	conv_strides	pool_ksize	pool_strides	conv_num_outputs	cnn	fc	fc_out	batch_size	dropout(%)	loss	training_accuracy	test_accuracy
20	5,5	1,1	2,2	2,2	16	1	1	256	64	10	0.0479	0.6196	0.6186505573
20	5,5	1,1	2,2	2,2	16	1	1	256	64	20	0.0849	0.6294	0.6241104299
20	5,5	1,1	2,2	2,2	16	1	1	256	64	30	0.1997	0.6306	0.6236942679
20	5,5	1,1	2,2	2,2	16	1	1	256	64	40	0.1854	0.6304	0.6317875116
20	5,5	1,1	2,2	2,2	16	1	1	256	64	50	0.271	0.649	0.6538614465
20	5,5	1,1	2,2	2,2	16	1	1	256	64	60	0.3716	0.6522	0.646695586
20	5,5	1,1	2,2	2,2	16	1	1	256	64	70	0.6114	0.6288	0.6313699427
20	5,5	1,1	2,2	2,2	16	1	1	256	64	80	0.73	0.6192	0.6236066888

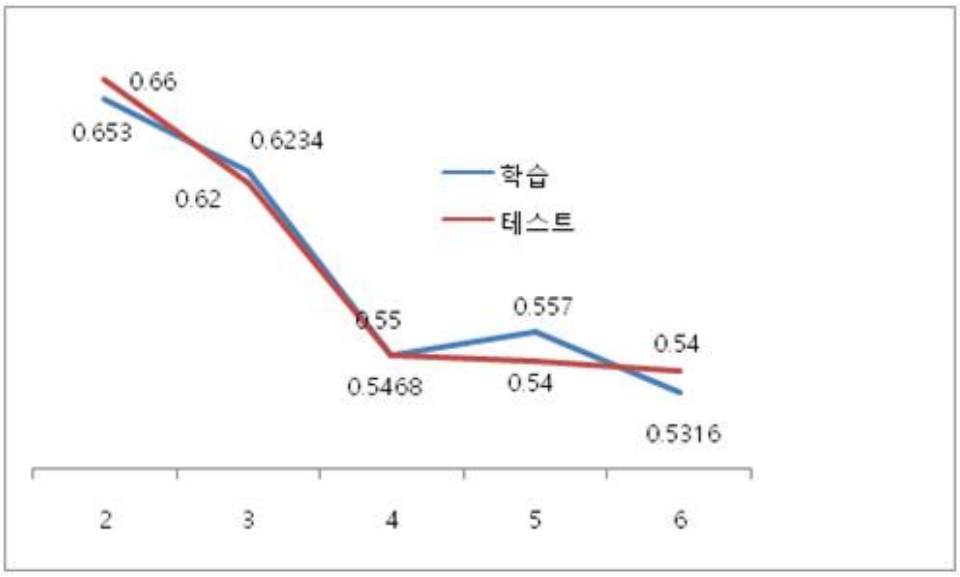
Performance change according to batch size change

The results of the investigation of the accuracy trend according to the batch size change are as follows. It showed the best performance at batch size 256, but it can be seen that it continues to decrease when the size increases by two multiples.



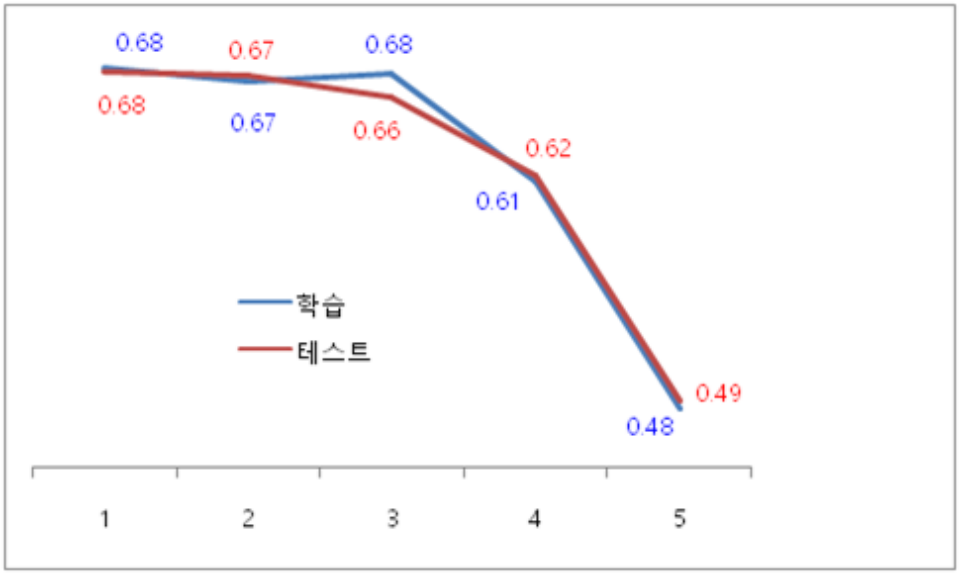
Performance change according to change of convolutional layer

The results of the investigation of the accuracy trend according to the change of the convolution layer structure (complexity) are as follows. Overall, the performance is deteriorating with the addition of layers, and it is understood that simply increasing the stacking arrangement of convolutional layers or adding nodes is not an unconditional improvement in performance. As for related phenomena, it is considered that additional research on accuracy change and improvement plans through various types of layer structure through diversification of the data to be analyzed and output node structure change, etc. are required.



Performance change according to the change of fully connected layer

The results of the investigation of the accuracy trend according to the change in the fully connected layer structure (complexity) are as follows. Overall, performance is rapidly deteriorating with the addition of layers, and even if the network complexity is increased by simply adding a fully connected layer like a CNN layer or adding a node, it is found that it is not an unconditional improvement in performance. As for related phenomena, it is considered that additional research on accuracy change and improvement plans through various types of layer structure through diversification of the data to be analyzed and output node structure change, etc. are required.



Reflection

Looking at the results of this study, it can be seen that the test accuracy according to the dropout ratio change gradually decreases after the maximum value of about 65.4% is formed at the dropout ratio level of 50-60%. Although it can have a value, it is shown as the actual experimental results that an appropriate level of node and layer drop reduces the overfitting phenomenon that can commonly occur in the neural network learning structure and improves the performance of CNN image learning. In the case of the batch size parameter, the size is doubled and the test is conducted. As a result of the test, it can be seen that the second setting, which is 512, shows the best accuracy, and as the size increases, the learning performance steadily decreases gradually. It is shown that the increase in mini batch size contributes to improving the learning speed when training a neural network for a large amount of data, but an excessive increase in size may cause a decrease in learning performance. The experiment according to the change of the complexity of the convolution layer shows that the overall performance is deteriorating due to the addition of layers and nodes, unlike expected, and that a simple layer and node addition and an increase in network complexity are not unconditionally improving the performance. The experimental results according to the change in the complexity of the fully connected layer show that the performance of the CNN is rapidly degraded compared to the convolutional layer as the layer is added as a whole, and the simple addition of a layer and the increase in network complexity are considered to be the cause of the rapid deterioration of the CNN performance.

Improvement

The performance degradation resulting from the complexity of these neural nets is thought to require further study of how the parameters affect performance within the neural network through various output node structure changes for various analysis and learning target data. In future research, I will supplement verification system infrastructure such as adding GPU and utilize proven high-performance CNN structure to obtain maximum exposure for experiments to derive more descriptive test results among hyperparameters and determine CNN layer structure.

References

[1] G. S. Choi, C. Yu, R. M. Jin, S. K. Yu and M. G. Chun, “Shortterm water demand forecasting algorithm using AR model and

MLP,”Journal of Korean Institute of Intelligent Systems, vol. 19, no. 5, pp.713-719, 2009.

[2] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, “Gradient-based learning applied to document recognition,”Proceedings of the IEEE, vol. 86 no. 11, pp. 2278-2324, 1998.

[3] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, “dropout: a simple way to prevent neural

networks from overfitting,”Journal of Machine Learning Research, vol. 15, no. 1, pp. 1929-1958, 2014.

[4] J. H. Yu and K. B. Sim, “Face Classification Using Cascade

Facial Detection and Convolutional Neural Network,”Journal of Korean Institute of Intelligent Systems, vol. 26, no. 1, pp. 70-75, 2016.

[5] A. Krizhevsky, I. Sutskever and G. E. Hinton, “Imagenet classification with deep convolutional neural networks”. In Advances in neural information processing systems, pp. 1097-1105, 2012.

[6] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,”In European conference on computer vision, pp. 818-833, 2014.

[7] W. Wang, J. Yang, J. Xiao, S. Li and D. Zhou, “Face Recognition

Based on Deep Learning,”Human Centered Computing, pp.812-820, 2014.

[8] S. Ahn, “Deep Learning Architectures and Applications,”Journal of Intelligence and Information Systems, vol. 22, no. 2, pp. 127-142, 2016.

[9] 이우영, 고광은, 김종우, 심귀보, “HS 알고리즘을 이용한 CNN의 hyperparameter 결정 기법”, Journal of Korean Institute of Intelligent Systems Vol.27. NO. 1. February 2017. pp. 022-028

[10] Kohavi, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In Proceedings of the 14th International Joint Conference on Artificial Intelligence, Montreal, QC, Canada, 20–25 August 1995; pp. 1137–1143.

[11] Schaer, R.; Müller, H.; Depeursinge, A. Optimized distributed hyperparameter search and simulation for lung texture classification in CT using hadoop. J. Imaging 2016, 2, 19.

[12] Bergstra, J.; Bengio, Y. Random search for hyper-parameter optimization. J. Mach. Learn. Res. 2012, 13, 281–305.

[13] Snoek, J.; Larochelle, H.; Adams, R.P. Practical bayesian optimization of machine learning algorithms. In Proceedings of the 25th International Conference on Neural Information Processing System, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 2951–2959.

[14] Hutter, F.; Hoos, H.H.; Leyton-Brown, K. Sequential model-based optimization for general algorithm configuration. In Proceedings of the 5th International Conference on Learning and Intelligent Optimization, Rome, Italy, 17–21 January 2011; pp. 507–523.

[15] Murray,I.;Adams, R.P.Slice sampling covariance hyperparameters of latent gaussian models. InProceedings of the 24th Annual Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 6–9 December 2010; pp. 1723–1731.

[16] Gelbart, M.A. 2015. Constrained Bayesian Optimization and Applications. Ph.D. Thesis, Harvard University, Cambridge, MA, USA, 2015.

[17] Li, L.; Jamieson, K.; DeSalvo, G.; Rostamizadeh, A.; Talwalkar, A. Hyperband: A novel bandit-based approach to hyperparameter optimization. arXiv 2016, arXiv:1603.06560.

[18] Loshchilov, I.; Hutter, F. CMA-ES for hyperparameter optimization of deep neural networks. arXiv 2016, arXiv:1604.07269.

[19] Luketina, J.; Berglund, M.; Greff, K.; Raiko, C.T. Scalable gradient-based tuning of continuous regularization hyperparameters. arXiv 2015, arXiv:1511.06727.

[20] Chan, L.-W.; Fallside, F. An adaptive training algorithm for back propagation networks. Comput. SpeechLang. 1987, 2, 205–218.

[21] Larsen, J.; Svarer, C.; Andersen, L.N.; Hansen, L.K. Adaptive Regularization in Neural Network Modeling. In Neural Networks: Tricks of the Trade; Springer: Berlin, Germany, 1998; pp. 113–132.

[22] Pedregosa, F. Hyperparameter optimization with approximate gradient. arXiv 2016, arXiv:1602.02355.

[23] Yu, C.; Liu, B. A backpropagation algorithm with adaptive learning rate and momentum coefficient. In Proceedings of the 2002 International Joint Conference on Neural Networks, Piscataway, NJ, USA, 12–17 May 2002; pp. 1218–1223.

[24] Zeiler, M.D. Adadelta: An adaptive learning rate method. arXiv 2012, arXiv:1212.5701.

[25] Caruana, R.; Lawrence, S.; Giles, L. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In Proceedings of the 2001 Neural Information Processing Systems Conference, Vancouver, BC, Canada, 3–8 December 2001; pp. 402–408.

[26] Graves, A.; Mohamed, A.; Hinton, G. Speech recognition with deep recurrent neural networks. In Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada, 26–31 May 2013; pp. 6645–6649.

[27] Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, 13–15 May 2010; pp. 249–256.

[28] Young, S.R.; Rose, D.C.; Karnowski, T.P.; Lim, S.-H.; Patton, R.M. Optimizing deep learning hyper-parameters through an evolutionary algorithm. In Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments, Austin, TX, USA, 15–20 November 2015.

[29] Kulkarni, P.; Zepeda, J.; Jurie, F.; Pérez, P.; Chevallier, L. Learning the structure of deep architectures using L1 regularization. In Proceedings of the British Machine Vision Conference, Swansea, UK, 7–10 September 2015; pp. 23.1–23.11.

[30] Zoph, B.; Le, Q.V. Neural architecture search with reinforcement learning. arXiv 2016, arXiv:1611.01578.

[31] Miikkulainen, R.; Liang, J.; Meyerson, E.; Rawal, A.; Fink, D.; Francon, O.; Raju, B.; Navruzyan, A.; Duffy, N.; Hodjat, B. Evolving deep neural networks. arXiv 2017, arXiv:1703.00548.

[32] Real, E.; Moore, S.; Selle, A.; Saxena, S.; Suematsu, Y.L.; Le, Q.; Kurakin, A. Large-scale evolution of image classifiers. arXiv 2017, arXiv:1703.01041.

[33] Baker, B.; Gupta, O.; Naik, N.; Raskar, R. Designing neural network architectures using reinforcement learning. arXiv 2016, arXiv:1611.02167.

[34] Zeiler, M.D.; Fergus, R. Visualizing and understanding convolutional networks. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014; pp. 818–833.

[35] Saleh Albelwi; Mahmood. A Framework for Designing the Architectures of Deep Convolutional Neural Networks. Computer Science and Engineering Department, University of Bridgeport, Bridgeport, CT 06604, USA