Convolution Neural Network for image classification

Pham Thuy Tien Le

Abstract

Image classification has many applications in various fields such as healthcare system, autonomous vehicle manufacture, and security sector. Therefore, this paper aims at building a Convolution Neural Network (CNN) which can generate good predictions for this image classification task. The CIFAR-10 dataset were employed in this paper. In addition, four main experiments were conducted to figure out (1) if pre-processing steps are important to improve model's performance, (2) compare performance of different architectures in classifying images, (3) if data augmentation methods can mitigate overfitting problem and increase accuracy scores, and (4) evaluate effects of chosen hyperparameters on model's performance. According to the achieved results, the accuracy score of CNN models built on pre-processed datasets was much higher than the figure for the original dataset, hence pre-processing steps are important in improving model's performance. In terms of data augmentation techniques, if the model is too simple with just several layers, these techniques can affect negatively to model's performance. However, if the model is complicated enough, with a deeper network, data augmentation methods can increase model's performance, reduce overfitting problem, and increase accuracy scores. In addition, among different architectures, Resnet-50 obtained the highest accuracy score on both the training dataset and the validation dataset. After tuning hyperparameters, the Resnet-50 using ReLU activation function and Adam optimizer achieved highest accuracy scores. In addition, on the testing dataset, this best CNN obtained the accuracy score of about 92%, and most of classes were predicted correctly about 90% their total images.

1. Introduction

Image classification are becoming more important in various industry fields. For example, in autonomous automobile manufacture, image classification are necessary because it can support self-driving cars in detecting and recognizing other objects such as other vehicles, pedestrians, animals, traffic lights on the road. In healthcare system, image classification can support health professionals in diagnosing patient's health problems or diseases such as lung disease, heart disease, and cancer. In addition, image classification can be used to automatically categorize products on e-commerce websites, which can result in better customer's experience because customers can find relevant products quickly. Image classification also can promote the development of security sector such as surveillance, and facial recognition, since it can be used to identify person, objects, and then potential threats can be addressed quickly. Due to various applications of image classification, this paper aims at developing a Convolution Neural Network (CNN) which can perform image classification task well, using CIFAR-10 dataset. Particularly, several CNN architectures were conducted, and a comparison between these architecture's performance was carried out to find out a CNN model which can achieve the best performance, and generate good prediction on unknown datasets.

The organization of this paper is as follows: Sec. 2 presents literature review with several related research papers, Sec. 3 shows details of the chosen method in this paper, Sec. 4 provides details of experiments and analysis of results, and Sec. 5 summarises key points of the paper and presents discussion for future improvements.

2. Literature review

Many related research papers [4,8–10,13,15] concentrating on image classification tasks using the CIFAR-10 dataset were implemented. These research papers employed various Machine Learning (ML) methods and Deep Learning (DL) methods. In terms of DL methods, these papers conducted several different CNN architectures [8, 9, 15], and experimented various values for hyperparameters such as learning rate, momentum, activation function, and pooling method [8, 10, 13].

The research conducted by Zongze Li [4] used the CIFAR-10 dataset to classify red cars. Because this re-

search aimed at distinguish red cars from other cars, only images containing automobile were employed. Values of each channel in a single image is summed, and the proportion of red channel were computed by dividing summation value of red channel by total summation values of red, green and blue channels. If this proportion is greater than one-third, this image is categorized as a red car. Authors employed three different methods for this classification task, which were Support Vector Machine (SVM), Random Forest (RF), and Convolution Neural Network (CNN). SVM was specified with regularization parameter of 1.0, kernel and degree of Radial basis function (RBF) and 3 respectively. RF was built with 100 trees and Gini criterion. CNN was constructed with 11 epochs in this paper. Without pre-processing steps, the highest accuracy score of RF was only about 40%, while the figures for SVM and CNN were around 46% and 46.5% respectively. After converting the response variable to binary classes with red cars relabelled as 1 and other cars relabelled as 0, the accuracy score of all these methods increased dramatically to over 94%.

The CIFAR-10 dataset was also employed in the research [13]. In this paper, a CNN model was built with Bayesian optimization. There were several stacks included in this model, each stack comprised a convolutional layer with filter size across layers of 32x32, 16x16, and 8x8 respectively. Batch normalization layer and a Rectified Linear Unit (ReLU) layer were also employed. Batch size was set to 128, and Stochastic Gradient Descent (SGD) with momentum was employed to smooth gradient descent steps. Additionally, four hyperparameters were tuned to find the optimal model, including initial learning rate, momentum, ridge regularization, and network depth. After tuning hyperparameters, the best CNN model was a model with network depth of 2, learning rate of 0.003, momentum of 0.936, and Ridge regularization (L2) value of $2.819\text{e}{-07}$. In terms of its performance, this model obtained high accuracy score on testing data for airplanes, automobile, trucks, frogs, ships, and horses, which were greater than 80%. On the contrary, the accuracy score for other objects were pretty low, for example, 66% for birds, 57% for cats, 74% for deer, and 68% for dogs. Although this model performed well on some specific objects and images, it also had disadvantages since the acquisition function used in Bayesian optimization set the search space early, the model might missed important features.

In the research conducted by Remerscheid *et al*. [9], authors constructed a SmoothNet architecture with 5 consecutive SmoothBlocks, followed by convolutional layers, average pooling layers, linear layers, and Scaled Exponential Linear Unit (SELU) layers. Each SmoothBlock included convolutional layers, max pooling layers, group normalization, and SELU activation function. In addition, Xavier-Glorot weight initialization, a learning rate decay of 0.9 every five epochs, an initial learning rate of 0.002, a momentum of 0.9, a weight decay of 0.0002, and a batch size of 256, were also employed in this experiment. The CIFAR-10 and ImageNet datasets were used in this research. This SmoothNet architecture achieved the accuracy score on the CIFAR-10 dataset was about 73.5%, while the figure for the ImageNet dataset was only 69.7%. In addition, this SmoothNet architecture was experimented to achieve higher accuracy score than other architectures such as ResNet-18, ResNet-34, DenseNet-121, and EfficientNet-B0.

A research [10] was conducted to evaluate the effect of mixed fuzzy pooling method in convolution networks, using the CIFAR-10 dataset and the MNIST dataset. The mixed fuzzy pooling is a combination of max pooling and average pooling methods. A CNN was built with an input layer, a convolutional layer with ReLU activation function, a pooling layer, a fully connected layer, a classification layer, and an output layer. According to the results, model using mixed fuzzy pooling method obtained higher accuracy scores on these both datasets than models using only fuzzy pooling method. However, although the accuracy score on the MNIST dataset was high (around 90%), the figure for the CIFAR-10 dataset was much lower with just about 20% - 27%.

In the research paper [15], the performance of PCNN model, which was a learning algorithm penalizing the loss function with danger samples but not all samples to enable CNN to pay more attention to danger samples and to learn effective information more accurately, was compared with the other two traditional CNN models. These three CNN models including PCNN, traditional CNN, and CNN with Ridge regularization (L2), were evaluated on the CIFAR-10 dataset. They were built with 1 input layer, 2 consecutive convolutional layers with 64 filters, a 2x2 max pooling layer, then 2 consecutive convolutional layers with 128 filters, a 2x2 max pooling layer, 3 consecutive convolutional layers with 256 filters, followed by a fully connected layer. Particularly, all convolutional layers had kernel size of 3x3, batch size was 50, learning rate was set to 0.001, and Dropout method were employed to prevent models from overfitting problem. Based on the results of this research, PCNN just achieved slightly higher accuracy score than other models, particularly, accuracy score of PCNN was 96.44%, the figure for traditional CNN was 96.15%, and CNN with L2 regularization was 96.3%.

The research conducted by Misra [8] also employed the CIFAR-10 dataset to evaluate the effect of chosen activation functions in CNN models, while keeping other parameters unchanged. There were various networks used to carry out this experiment, including ResNet-20, WRN-10-2, SimpleNet, Xception Net, Capsule Net, Inception ResNet v2, DenseNet-121, MobileNet-v2, ShuffleNet-v1, Inception v3, and Efficient Net B0. According to the results, Mish, a novel self regularized non-monotonic activation function, obtained higher accuracy score than other activation functions such as ReLU and Swish, on all of these architectures. The highest accuracy score that a CNN with Mish activation function obtained was 92.02%, based on ResNet-20 network.

3. Method description

3.1. Data Splitting

Training data stored in 5 different batches were gathered into 1 training dataset, including 50000 images, with 5000 images for each class. Hence, the training set had 50000 rows, 3072 columns for image's pixels, and 1 column for image's labels. The testing dataset was divided into a validation set and a testing set, with the ratio of 5:5. After splitting, each set has 5000 images. The distribution of classes in these three datasets were shown in the below Fig. 1.

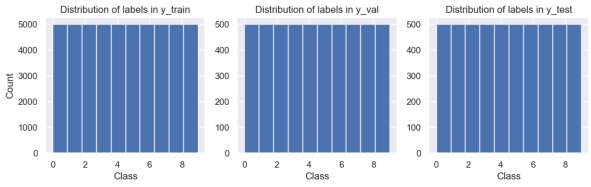

Figure 1. Distribution of classes in the training dataset, the validation dataset and the testing dataset. The ratio between these classes are the same in all three datasets.

As shown in the Fig. 1, Stratified sampling method was employed to keep the proportion of classes the same in both the validation set and the testing set after spliting. In addition, random state was defined to reproduce the same randomness in future improvements.

3.2. Data Pre-processing

Data pre-processing steps were performed in the training dataset first, then these steps were applied on the validation set and the testing set. Pre-processing methods considered in this paper were Standard scaling method, Max-Min scaling method, and Data augmentation methods such as flipping, rotation, and contrast.

3.2.1 Standard scaling method

In terms of Standard scaling method, the training data was normalized to have mean value of 0 and standard deviation of 1, based on the following Eq. 1:

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

with n represents for the total number of observations in the training set, mean is computed by Eq. 2:

$$\mu = \frac{1}{n}\sum_{i=1}^{n}(x_i) \tag{2}$$

and standard deviation is calculated as Eq. 3:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2} \tag{3}$$

3.2.2 Max-Min scaling method

In terms of Max-Min scaling method, the training data was scaled to a specific range with values going from 0 to 1, according to the following Eq. 4. Based on this equation, $x.min(axis = 0)$ represents for the minimum feature value, and $x.max(axis = 0)$ shows the maximum feature value. Additionally, in Eq. 5, because the range defined for Max-Min scaling method in this paper was values ranging between 0 and 1, max and min in this equation was set to 1 and 0 respectively.

$$x_{std} = \frac{x – x.min(axis = 0)}{x.max(axis = 0) – x.min(axis = 0)} \tag{4}$$

$$x_{scaled} = x_{std} \times (max – min) + min \tag{5}$$

3.2.3 Data Augmentation

Besides Standard scaling method and Max-min scaling method, other pre-processing methods considered in this paper were methods of data augmentation. Data augmentation methods were used to make the training data more diverse by using some data transformation techniques, which can lead to better model's performance because model is trained to learn from a high level of diversity of training set. When data augmentation methods are used to increase the number of observations in the training dataset, they also can mitigate potential overfitting problems since models can not overfit all the observations in this set. Therefore, in terms of advantages, data augmentation methods can support models

to generalize better, increase the diversity of the training dataset, improve model's performance, increase the accuracy score of model's prediction. However, augmented datasets can comprise the biases of the datasets. There are several data augmentation methods that can be used to pre-process the training dataset, which are flipping, rotation, scaling, cropping, translation, random bright, contrast, saturation. In this paper, three data augmentation methods considered were random flipping, random rotation, and random contrast. Random flipping was used to flip images input horizontally and vertically, while random rotation was employed to rotate images by 0 to 360 degrees clockwise, based on a fraction of 2Pi, and contrast was used to change the degree of separation between the darkest and brightest areas in images with a contrast factor. In addition, contrast was adjusted independently for each channel of images during training.

3.3. Convolution Neural Network

In image classification, Convolution Neural Network (CNN) models commonly comprise convolutional layers, and pooling layers. An example of a CNN model employed to classify images were shown in the Fig. 2.

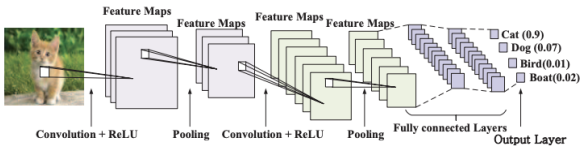

Figure 2. An example of a Convolution Neural Network architecture [17]. This architecture includes an input layer, convolutional layers, pooling layers, fully connected layers, and an output layer.

Convolutional layers uses filters to perform convolutional operations on the input data [11]. There are two common types of convolutional layers, including the same convolutional layer and the valid convolutional layer [11]. The same convolutional operation allows zero pads to the input data while the valid convolutional operation do not add any extra pads [11]. When defining a convolutional layer, some parameters such as the number of filters, kernel size, stride size should be specified [11]. Besides, pooling layers are used to reduce the number of feature map's dimensions. There are 2 common types of pooling layer such as max pooling layer and average pooling layer. In max pooling layer, the output value for each pooling region is the maximum value of its input values, as shown in Fig. 3. In image classification, this type of pooling method selects the most important features of objects in images, such as their edges, and ignores other less important information. Different from max pooling layer, average pooling layer takes the average value of all input values in each pooling region, as shown in Fig. 3. This method can keep more information of the input data, not only pixels having the largest value in pooling region like max pooling method. However, average pooling method can dilute the most outstanding features. When creating pooling layers, the type of pooling method, kernel size, and stride size also need to be defined [11].

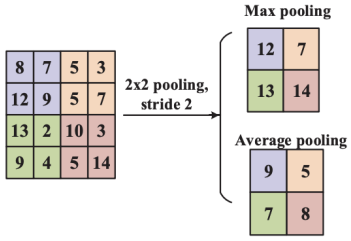

Figure 3. Two common types of pooling layer [17]

In this paper, CNN models were built with an input layer, convolutional layers, pooling layers, fully connected layers, and the output layer with 10 neurons and Softmax function. These 10 neurons in the output layer were used for 10 classes contained in the dataset. The loss function was Sparse categorical crossentropy since labels were encoded as integers from 0 to 9. In addition, Early stopping method was employed to prevent models from potential overfitting problem, because this problem can make models too fit to the training dataset and lead to high generalized errors when these models were applied on the validation and testing dataset. Early stopping method was defined with patience of 5, and monitored by validation loss values. In addition, because the classification task in this paper was related to image classification, several appropriate architectures were conducted to find out the optimal architecture, including Visual Geometry Group (VGG16), GoogLeNet, InceptionV3, and Residual Neural Network (ResNet-50).

3.3.1 Visual Geometry Group (VGG16)

VGG-16, a standard deep CNN architecture first proposed by Andrew Zisserman and Karen Simonyan in 2013, includes an input layer, 13 convolutional layers (with kernel size of 3x3), 5 max pooling layers with size 2x2, and 3 fully connected layers, as shown in the below Fig. 4. All convolutional layers in this architecture employed a kernel size of 3x3 since this kernel size were seen as the optimal size to capture left-right and

up-down information of the input image. VGG-16 architecture was used in this paper since VGG-16 is one of the most popular image recognition architectures.

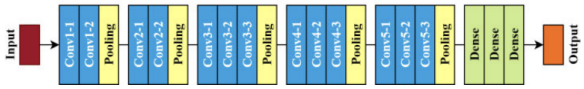

Figure 4. Architecture of the VGG-16 model [7]

Additionally, according to the above Fig. 4, besides an input layer and an output layer, VGG-16 architecture included multiple consecutive 3x3 convolutional layers, along with a max pooling layer. After 4 blocks of convolutional layers and max pooling layers, there were 3 consecutive fully connected layers using ReLU activation function. In addition, the output layer of this architecture also employed Softmax activation function.

3.3.2 GoogLeNet

GoogLeNet architecture was constructed based on Inception modules, as shown in the below Fig. 5. This architecture included 9 Inception modules, in which the input can be passed through different types of layers at once to extract distinct features parallelly and then concatenate them. In particular, Inception modules were created with several convolutional layers with kernel size 1x1 for dimensionality reduction, a convolutional layer with kernel size of 3x3, a convolutional layer with kernel size of 5x5, and a 3x3 max pooling layer, as presented in the below Fig. 6.

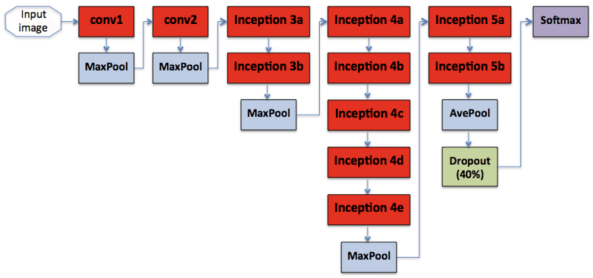

Figure 5. Architecture of the GoogLeNet model [2]

Besides these Inception modules, there were 2 blocks, each block included a convolutional layer and a max pooling layer, created before adding Inception modules. In addition, a max pooling layer was built after the first 2 Inception modules, and another max pooling layer was added after the next 5 Inception modules. After the last 2 Inception modules, an average pooling layer and a Dropout layer were constructed. This

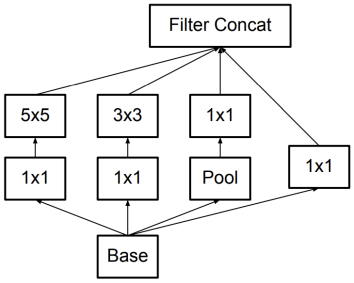

Figure 6. Inception module of GoogLeNet architecture [12]

Dropout layer was employed as a regularization method to prevent model from being overfitting by ignoring 40% neurons during training. This architecture employed Softmax function in the output layer for classification.

3.3.3 InceptionV3

InceptionV3 architecture also can be used for image classification. Like GoogLeNet, InceptionV3 employed many Inception modules. In addition, this architecture can employ other methods such as Factorized convolutions and Auxiliary classifiers, along with Batch normalization method. Particularly, in this architecture, a convolutional layer with kernel size 5x5 can be factorized into smaller size, such as two 3x3 convolutional layers [12]. This method can reduce the number of parameters and then lead to lower computation. In addition, auxiliary classifiers were used to improve the convergence of this deep network by pushing useful gradients to the lower layers, hence mitigate vanishing gradient problem [12]. An example schematic diagram of InceptionV3 architecture were shown in the below Fig. 7.

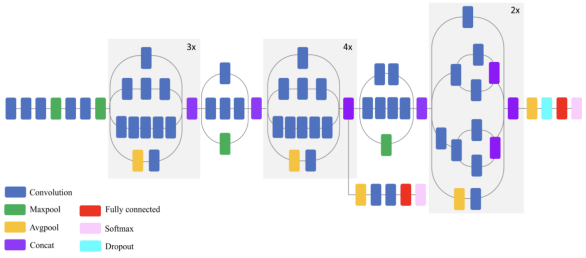

Figure 7. Architecture of the InceptionV3 model [6]

3.3.4 ResNet-50

ResNet-50 is a CNN architecture including a network depth of 50 layers. This architecture was pretrained on a million images from the ImageNet dataset with 1000

different labels. The network had an image input size of 224x224. ResNet-50 architecture were shown in the below Fig. 8. This architecture included 5 main stages. In the first stage (Stage 1), the input data ran through a convolutional layer, Batch normalization, a ReLU activation function, and a max pooling layer. From Stage 2 to Stage 5, there were 4 consecutive blocks containing convolutional layers and identity layers. The final part of this architecture included an average pooling layer, and a fully connected layer.

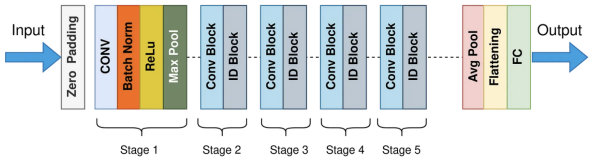

Figure 8. Architecture of the ResNet-50 model [1]

Except for Stage 1, in other stages, there were 1 block of convolutional layers, and 2 blocks of identity layers. Each block included 3 convolutional layers and 3 batch normalization layers. Batch normalization layers were included between every 2 convolutional layers.

3.3.5 Activation function

Activation functions compute weighted sum of inputs and biases to define whether a neuron's input is important to be activated [18]. In addition, activation functions are used to introduce the property of non-linearity to a DL model [18]. According to Misra [8], Mish activation function can result in better model's performance than ReLU and Swish activation function. Hence, in this paper, an experiment is carried out to compare model's performance when applying Mish activation function and other activation functions such as Rectified Linear Unit (ReLU), and Scaled Exponential Linear Unit (SELU).

Although a ReLU activation function can mitigate vanishing gradient problem and increase computational simplicity, its three main problems are: (1) Non zero mean, (2) Negative missing and (3) Unbounded output. Non zero mean problem happens because only positive parts are remained unchanged while all negatives values are converted to 0, ReLU function is non-negative and its mean is obviously greater than 0 [18]. Therefore, this problem can affect negatively to model's convergence [18]. When negative missing problem happens, ReLU can reduce model's ability to fit and train the data because all negative values are converted to 0 while this negative part can be helpful for the model. Unbounded output problem means that output of ReLU can range from 0 to infinity [18]. The way that ReLU activation function works was presented in the Eq. 6. A graph of ReLU activation function was shown in the Fig. 9.

$$f(x) = max(0, x) \tag{6}$$

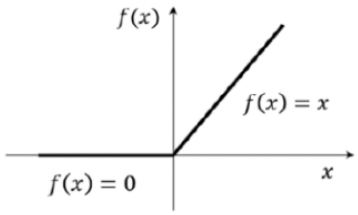

Figure 9. A graph of ReLU activation function [5]. In this activation function, all negative values are converted to 0.

A SELU activation function also can mitigate vanishing gradient problem, decrease training time, and push the mean of activation function closer to 0 [16]. Like ELU activation function, SELU creates a smooth curve for negative values by multiplying them by a positive α slowly. In addition, SELU even modifies these negative values in a restricted manner with a scale parameter λ (greater than 1) to ensure that a slope of the line going through positive inputs larger than one, shown in Eq. 7. Besides, SELU is a self-normalizing function, in which a layer keeps its previous layer's mean and variance. A graph of SELU activation was shown in Fig. 10.

$$f(x) = \lambda\{ \begin{matrix} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{if } x \leq 0 \end{matrix} \tag{7}$$

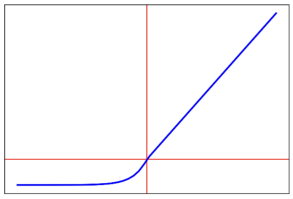

Figure 10. SELU activation function [3]. This function creates a line with slope greater than one for positive inputs.

Mish activation function are computed as the Eq. 8. According to this equation, this activation function has a lower bound, but it does not have an upper bound. This property can prevent models from gradient saturation problem [14]. In addition, because Mish activation function is non-monotonicity, it can keep the network gradient flow stable during training. The graph of Mish activation function were presented in the below Fig. 11.

$$f(x) = xtanh(ln(1 + e^x)) \tag{8}$$

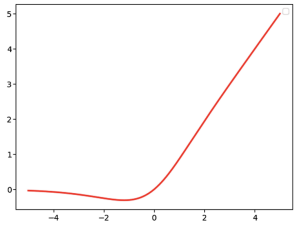

Figure 11. A graph of Mish activation function [14]

3.3.6 Softmax function

Softmax function was considered in this paper since it is necessary for multi-label classification. This function, employed in the output layer, returns probabilities of being in each class for each image. Then these images can be classified into a specific class which have the highest probability. Softmax function formula were shown in the below Eq. 9.

$$softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{9}$$

According to this Eq. 9, K represents for the number of classes, for example, K equals to 10 in this paper. Additionally, $softmax(z_i)$ returns a probability that an image belongs to a class i. This formula needs to be calculated for each class, then the class having the highest value of $softmax(z)$ is the final class of the image.

3.3.7 Optimizer

Optimizers are used to compute optimal parameters which can minimize the loss functions. In this paper, Stochastic Gradient Descent (SGD) with learning rate of 0.001 were employed in the baseline CNN. SGD minimizes the loss function through calculating the gradients, and learning rates are used to control the speed in updating parameters estimates. In tuning hyperparameters stage, Adaptive Moment Estimation (Adam), and Nesterov-accelerated Adaptive Moment Estimation (Nadam) were also considered. While Adam is a combination of SGD with momentum and Root Mean Squared Propagation (RMSProp), Nadam employs SGD with Nesterov acceleration. SGD with momentum and Nesterov momentum can prevent models from being trapped in local minima, instead, they can reach the global minima and find out optimal parameter estimates.

3.4. Model assessment

After the final CNN model was trained, this model was applied on the testing dataset to evaluate its performance and generate predictions. Main metrics such as sensitivity score, specificity score, accuracy score, and precision score are computed to assess the final CNN model's performance on the testing data. These metrics are calculated as following equations:

$$\text{Sensitivity (TPR)} = \frac{\text{True Positive}}{\text{True Positive + False Negative}} \tag{10}$$

$$\text{Specificity} = \frac{\text{True Negative}}{\text{True Negative + False Positive}} \tag{11}$$

$$\text{Accuracy} = \frac{\text{True Positive + True Negative}}{\text{Total Positive + Total Negative}} \tag{12}$$

$$\text{Precision (PPV)} = \frac{\text{True Positive}}{\text{True Positive + False Positive}} \tag{13}$$

$$\text{f1} = \frac{2(\text{PPV} * \text{TPR})}{\text{PPV} + \text{TPR}} \tag{14}$$

4. Experiments and results

In this paper, the CIFAR-10 dataset was employed, downloaded from https://www.cs.toronto.edu/~kriz/cifar.html. The dataset comprises 60000 32x32x3 images, meaning that each image has 1024 pixels, along with three channels for red, green, and blue (RGB). The first layer of 1024 entries is the red channel, the second layer of 1024 entries represents for the green channel, and the last layer of 1024 entries represents for the blue channel. Among these images, 50000 images were used for training models, stored in 5 training batches, and 1 testing batch included 10000 images. Both training and testing dataset included 3072 columns, representing for 3072 image's pixels, and 1 column showing image's labels. Ten classes comprised in this dataset are vehicles such as airplane, automobile, ship, truck, and animals such as bird, cat, deer, dog, frog, and horse. Each class had 10000 images, their respective labels were shown in the below Tab. 1, and their images were shown in the below Fig. 12. There were four experiments conducted in this paper. The first experiment was carried out to compare model's performance on the original dataset and the pre-processed dataset, in order to

Figure 12. Images in the CIFAR-10 dataset

Label	Class
0	airplane
1	automobile
2	bird
3	cat
4	deer
5	dog
6	frog
7	horse
8	ship
9	truck

Table 1. 10 classes and respective labels in CIFAR-10 dataset

figure out if pre-processing steps are necessary for models to perform better. The second experiment was implemented to find out the CNN architecture that achieved highest accuracy score for this image classification task, using the CIFAR-10, among several different architectures. The third experiment was executed by applying data augmentation methods on this best architecture to figure out if these methods can improve model's performance. The last experiments was carried out by training the best CNN architecture with various values of chosen hyperparameter, to define the optimal hyperparamenters which can lead to better model's performance.

In the first experiment, a baseline CNN model was built with 1 input layer, 1 output layer with 10 neurons, using Softmax function and Sparse categorical crossentropy loss function. This baseline model contained 1 convolutional layer with 64 filters and kernel size of 3x3, followed by a 2x2 max pooling layer, a fully connected layer with 256 neurons and two Dropout layers. SGD optimizer with learning rate of 0.001 was employed in this baseline CNN model. Additionally, this model ran through 300 epochs, and batch size of 100. Early stopping method was also employed to prevent models from potential overfitting problem. The baseline model was trained on the original dataset, a dataset pre-processed by using Standard scaling method, Max-Min scaling method, and other data augmentation methods (flipping, rotation, and contrast). The training accuracy score and validation accuracy score of the baseline model were shown in Tab. 2. In terms of the original dataset, the baseline CNN model converged after around 5 epochs with the accuracy score on the training set and the validation set of only 10%. However, this baseline model performed better on the dataset pre-processed by only Standard scaling method, with the maximum accuracy score on the training set was about 81%, and the maximum accuracy score on the validation set was nearly 67%, after 43 epochs. Because this accuracy score on the training set was around 14% higher than the figure for the validation set, this gap can raise a signal for potential overfitting problem. On the pre-processed data using only Max-Min scaling method, after 127 epochs, this baseline model obtained the accuracy score on the training dataset of about 79%, and the figure for the validation dataset of nearly 66%. Through this experiment, the dataset should be pre-processed before training CNN models in order to improve the model's performance because accuracy score of models trained on the pre-processed dataset were significantly higher than the figure for the original dataset. However, according to this Tab. 2, when a CNN model only contains several layers, its capability is low, data augmentation methods applied on this model can make it perform worse, which was shown by the decrease in the training accuracy score and validation accuracy score. Hence, these data augmentation methods only should be considered in deeper and more complicated neural networks. Because the accuracy score of models trained on standard scaled dataset was the highest among other methods, the pre-processed training data using Standard scaling method was used to train other CNN architectures and analyze further.

The second experiment was conducted to compare the performance of different architectures, using the dataset pre-processed by Standard scaling method. Architectures considered in this paper are VGG16, GoogLeNet, GoogLeNet with auxiliary classifiers, InceptionV3, InceptionV3 with fine tuning, and Resnet-50. These architectures employed Sparse categorical crossentropy loss function, SGD optimizer with 0.001, and batch size of 100. Early stopping method also employed to mitigate potential overfitting problem. The output layer of these architecture used Softmax function for classification since classes were encoded as numbers from 0 to 9. In this paper, VGG16 architecture was built based on the VGG architecture shown in Fig. 4. The first two convolutional layers had 64 filters, then the num-

ber of filters increased to 128 in the next two convolutional layers, and went up to 256 in the next three convolutional layers, before remaining unchanged at 512 for the last convolutional layers. After that, 3 blocks with each block including a fully connected layer and a Dropout layer were added to the model. GoogLeNet was built with 9 Inception modules, convolutional layers, max pooling layers, average pooling layer, and Dropout layer. This architectures started with a stack including 2 convolutional layers and 2 max pooling layers before adding the first two Inception modules. After these layers, a 3x3 max pooling layer was created, followed by 5 consecutive Inception modules. After that, another max pooling layer was built, then the last two Inception modules. After the average pooling layer, A Dropout layer was built to ignore 40% neurons, then the remaining neurons were passed to the output layer. GoogLeNet with auxiliary classifiers was built with the same structure as the above GoogLeNet architecture. However, two auxiliary classifiers were created between Inception modules since these auxiliary classifiers can support the model to improve its convergence, and prevent the model from vanishing gradient problem. Particularly, an auxiliary classifier was added to this architecture after every three Inception modules. Each auxiliary classifier includes an average pooling layer, a convolutional layer with ReLU activation function and kernel size of 1x1 used for dimensionality reduction, a fully connected layer, a Dropout layer and an output layer using Softmax function. Because InceptionV3 was pretrained to classify other objects, in this paper, layers of the base Inception model were set to not trainable, and several new layers were added, including an average pooling layer, a fully connected layer with ReLU activation function, a Dropout layer, and an output layer. In this paper, because ResNet-50 was pretrained on Imagenet weights with input size of 224x224, 32x32x3 images in the CIFAR-10 dataset were up-sampled by a factor of 7x7 to the size of 224x224x3 before being passed to ResNet50 function. Then the output value ran through an average pooling layer, 2 fully connected layers with 1024 and 512 neurons respectively, using ReLU activation function, before going to the output layer. All these architectures were set to run through 50 epochs. The performance of all architectures on the training dataset and validation dataset were shown in the Tab. 3.

After comparing model's performance of different CNN architectures, according to the results shown in the Tab. 3, Resnet-50 (with Relu activation function and SGD optimizer) obtained the highest accuracy score on both the training dataset (above 99%) and the validation dataset (nearly 92%), hence, data augmentation methods were considered in this Resnet-50 architecture to figure out if they can affect to model's performance before tuning hyperparameters. This experiment was conducted by adding data augmentation layers after the resizing layer in the Resnet-50 architecture. Flipping method was used to flip the input images horizontally and vertically. Rotation method was employed to rotate the input images with a factor of 0.2 while contrast method modifies the difference between the darkest and brightest areas in the input images with a factor of 0.2. The Tab. 4 shows the performance of this architecture, with different methods of data augmentation. According to this Tab. 4, data augmentation techniques can perform better on Resnet-50 architecture than the baseline model since this Resnet-50 is more complicated and deeper with the greater number of layers and parameters. These methods can mitigate the overfitting problem, which was shown through the decrease in the difference between training accuracy score and testing accuracy score. This gap of the original Resnet-50 architecture was nearly 7%, while the figure for Resnet-50 with random flip was just about 5%. However, using many data augmentation techniques was not always appropriate since including all methods, such as random flip, random rotation and random contrast methods in this Resnet-50, reduced the model's performance significantly, with the training accuracy score of only 58.71% and the validation accuracy score of only 23.16%. In addition, through this experiment, the best architecture yielding the highest accuracy score on validation dataset (92.30%) was Resnet-50 with random flip method, its graph of training loss, tranining accuracy, validation loss and validation accuracy was shown in the Fig. 13. However, while validation accuracy score of this Resnet-50 was only 0.7% higher than the figure for the original Resnet-50, it required much longer time to converge. Therefore, the Resnet-50 without data augmentation was still selected to implement the experiment of tuning hyperparameters.

Last, the original Resnet-50 architecture was used to tune hyperparametes such as activation functions, and optimizers. There were 7 combinations of hyperparameters which were tuned in this experiment, keeping other parameters remain unchanged, as shown in Tab. 5. According to Tab. 5, Resnet-50 architecture obtained high validation accuracy score, regardless of different combinations of activation function and optimizer. In general, their validation accuracy scores were about 80% or above. The best CNN model in this paper is the Resnet-50 architecture with SELU activation function, SGD optimizer, and its learning rate is 0.001. The below

Original	Standard scale	Max-Min scale	Flip	Rotation	Contrast	Training accuracy	Validation accuracy
✓						10%	10%
	✓					**80.92%**	**66.64%**
		✓				79.09%	65.64%
✓			✓			10%	10%
	✓		✓			71.37%	65.36%
		✓	✓			59.97%	58.38%
✓				✓		10%	10%
	✓			✓		57.56%	56.36%
		✓		✓		44.95%	45.88%
✓					✓	10%	10%
	✓				✓	44.93%	37.34%
		✓			✓	74.06%	64.48%
✓			✓	✓		10%	10%
	✓		✓	✓		51.02%	49.16%
		✓	✓	✓		42.10%	41.18%
✓			✓		✓	10%	10%
	✓		✓		✓	38.22%	28.22%
		✓	✓		✓	65.94%	61.78%
✓				✓	✓	10%	10%
	✓			✓	✓	37.50%	34.34%
		✓		✓	✓	47.26%	46.00%
✓			✓	✓	✓	10%	10%
	✓		✓	✓	✓	32.94%	29.20%
		✓	✓	✓	✓	44.29%	44.32%

Table 2. Performance of the baseline model when applying different pre-processing methods

Architectures	Training loss	Training accuracy	Validation loss	Validation accuracy
VGG16	0.3803	88.34%	0.5497	82.86%
GoogLeNet	1.2190	54.64%	1.2441	53.38%
GoogLeNet with auxiliary classifiers	3.2261	61.98%	3.2923	60.10%
InceptionV3	0.7286	74.85%	0.6961	76.38%
Resnet-50	**0.0385**	**99.31%**	**0.2859**	**91.60%**

Table 3. Performance of different CNN architectures

Architectures	Training loss	Training accuracy	Validation loss	Validation accuracy
Resnet-50	0.0385	99.31%	0.2859	91.60%
Resnet-50 with random flip	**0.0844**	**97.46%**	**0.2421**	**92.30%**
Resnet-50 with random flip and random rotation	0.2544	91.16%	0.2585	91.50%
Resnet-50 with random flip, random rotation and random contrast	1.1673	58.71%	3.8990	23.16%

Table 4. Performance of the Resnet-50 architecture with different data augmentation methods

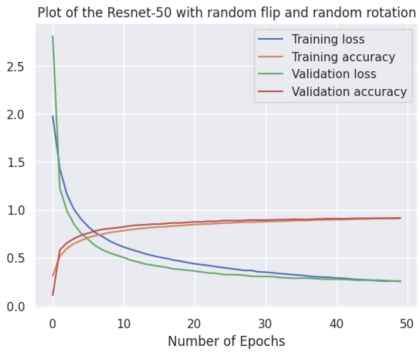

Figure 13. Performance plot of the Resnet-50 model with random flip and random rotation

Fig. 14 showed changes in training loss, training accuracy, validation loss, and validation accuracy of this best model through each epoch. In addition, this model converged after 17 epochs with the training accuracy score of 99.56% and the validation accuracy score of 91.70%, which was just slightly higher than the figure for the original Resnet-50 using ReLU and SGD optimizer.

Activation function	Optimizer	Validation accuracy
ReLU	Adam	86.82%
ReLU	SGD	91.60%
ReLU	Nadam	87.44%
SELU	Adam	76.78%
SELU	**SGD**	**91.70%**
SELU	Nadam	88.86%
Mish	Adam	82.58%

Table 5. Maximum validation accuracy score of Resnet-50 models with different combination of hyperparameters

However, in comparison with the baseline model, this best model's validation accuracy score was nearly 25%

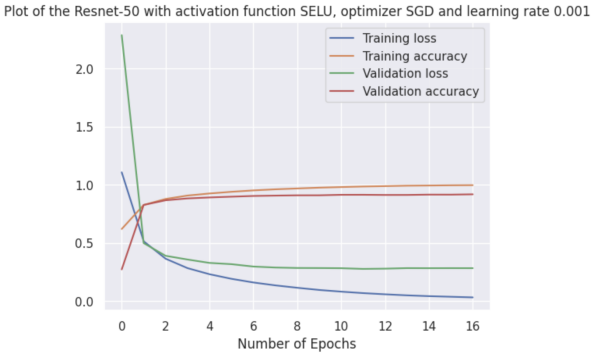

Figure 14. Performance plot of the Resnet-50 with SELU activation function, SGD optimizer and learning rate of 0.001.

higher than the figure for the baseline CNN. In addition, the difference between training accuracy score and validation accuracy score of this best model was only about 8%, which was 6% lower than the figure for the baseline model. Several reasons that can lead to this improvement. First, this best Resnet-50 has a deeper network with a greater number of layers than the baseline model, which can help model to learn more information of the input data. Second, because vanishing gradient and non-zero mean are two of common problems that can negatively affect model's performance, the SELU activation function also can contributed in this improvement of accuracy scores since it can mitigate vanishing gradient problem and push the mean of activation function closer to 0. In addition, different from ReLU activation function which converted all negative values to 0, SELU creates a smooth curve going through these negative numbers, hence it still can keep information of these input data, and improve model's performance. In addition, this best model's layers and number of parameters were shown in the Fig. 15.

Layer (type)	Output Shape	Param #
input_5 (InputLayer)	[(None, 32, 32, 3)]	0
up_sampling2d_2 (UpSampling2D)	(None, 224, 224, 3)	0
resnet50 (Functional)	(None, 7, 7, 2048)	23587712
global_average_pooling2d_2 (GlobalAveragePooling2D)	(None, 2048)	0
flatten_2 (Flatten)	(None, 2048)	0
dense_4 (Dense)	(None, 1024)	2098176
dense_5 (Dense)	(None, 512)	524800
classification (Dense)	(None, 10)	5130

Total params: 26215818 (100.01 MB)
Trainable params: 26162698 (99.80 MB)
Non-trainable params: 53120 (207.50 KB)

Figure 15. Layers and number of parameters of the best Resnet-50 model with SELU activation function, SGD optimizer and learning rate 0.001.

This best CNN model was used to evaluate model's performance on the testing dataset. In particular, its testing loss was about 0.258, and its accuracy score and precision score was around 91.3%. Compared to related works, this best Resnet-50 architecture achieved much higher accuracy score than research papers [9,10], however, its accuracy score was just similar to the research [8], and about 5% lower than the paper [15]. In terms of individual classes, each label was predicted accurately more than 90% of its number of images, except for label 2, 3 and 5 since they were only predicted precisely about 87%, 81% and 85% respectively, as shown in the below Fig. 16. In terms of predictions, around

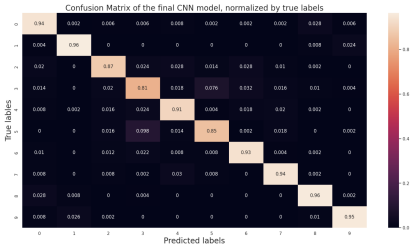

Figure 16. Confusion matrix of the Resnet-50 with SELU activation function, SGD optimizer, and learning rate of 0.001, normalized by true labels.

90% of predicted labels were the same with true labels, except for predictions of label 3 and 5, since only 82% labels predicted as 3 were correct, and 88% labels predicted as 5 were precise, as shown in the below Fig. 17. Other assessed metrics for each class were shown in the Tab. 6.

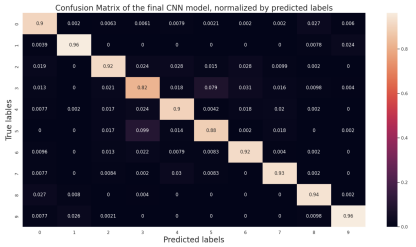

Figure 17. Confusion matrix of the Resnet-50 with SELU activation function, SGD optimizer, and learning rate of 0.001, normalized by predicted labels.

4.1. Code section

Code chunk, employed to execute all experiments in this paper, was uploaded to https://github.com/Tien-le98/Assignment1_DL. This code chunk used to implement data loading, data splitting, data pre-

Label	Precision	Recall	f1-score
0	0.9	0.94	0.92
1	0.96	0.96	0.96
2	0.92	0.87	0.89
3	0.82	0.81	0.81
4	0.90	0.91	0.90
5	0.88	0.85	0.87
6	0.92	0.93	0.93
7	0.93	0.94	0.94
8	0.94	0.96	0.95
9	0.96	0.95	0.96

Table 6. Precision score, Recall score, and f-1 score of the Resnet-50 model prediction

processing, building CNN architectures, tuning hyperparameters and evaluating model's performance.

5. Discussion and conclusion

Through results of experiments in this paper, preprocessed data steps are important in improving model's performance. In particular, Standard scaling method can improve model's performance better than other pre-processing methods. Additionally, although data augmentation techniques can have positive effects on model's performance, if the model does not have a large number of layers, these techniques can make it perform worse. It was shown through the decreased accuracy score of the baseline CNN model when data augmentation methods were employed. On the contrary, if the neural network is deeper, these data augmentation techniques can promote its performance and mitigate overfitting problem, leading to higher accuracy scores, which was shown in the performance of Resnet-50 architecture when applying different data augmentation methods. However, choosing appropriate number and type of data augmentation methods should be considered thoroughly since employing many techniques can lead to worse model's performance. Besides, among different architectures which are appropriate in image classification, Resnet-50 were seen as the best architecture, achieving the highest accuracy score on the training dataset and the validation dataset. After tuning hyperparameters, the Resnet-50 with SGD optimizer, and SELU activation function obtained the highest accuracy score on the validation dataset because SELU activation function can alleviate vanishing gradient, and non-zero mean problems. Therefore, hyperparameters can affect model's performance, tuning hyperparameters steps can contribute in choosing the best model, which can result in better overall accuracy, meaning that better model's performance. However, there are several points which should be improved in future research in order to promote model's performance. First, because this paper only implemented hyperparameter tuning for only activation functions and optimizers, a comprehensive approach should be conducted to figure out the effect of other hyperparameters to model's performance such as weight initializations, learning rate schedule, various pooling methods, and the number of hidden layers. Second, other data augmentation such as translation, cropping, saturation should be considered to figure out their effect on model's performance, in order to find the most appropriate data transformation for the input images. In terms of data augmentation methods, several combination of these techniques should also be experimented since their joint effect can positively change model's performance. Third, running time should be taken into account since architectures considered in this paper were computational and required significant running time. Last, signals of overfitting problem occurred during training, which was presented through the difference between training accuracy scores and validation accuracy scores. Hence, to mitigate this problem, several regularization techniques such as Dropout, L1 (Lasso) and L2 (Ridge) regularization method should be considered in future improvements.

References

[1] The annotated resnet-50. https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758. Accessed: 2023-10-23. 6

[2] Googlenet architecture implementation in keras with cifar-10 dataset. https://machinelearningknowledge.ai/googlenet-architecture-implementation-in-keras-with-cifar-10-dataset/. Accessed: 2023-10-23. 5

[3] Diego Aquino-Brítez, Andrés Ortiz, Julio Ortega, Javier León, Marco Formoso, John Q. Gan, and Juan José Escobar. Optimization of deep architectures for eeg sig-

nal classification: An automl approach using evolutionary algorithms. *Sensors*, 21, 2021. 6

[4] Zongze Li. Identification for red automobile based on cifar-10 using machine learning models. *Journal of Physics: Conference Series*, 2428, 2023. 1

[5] Z. Li, W.T. Nash, S.P. O'Brien, Y. Qiu, R.K. Gupta, and N. Birbilis. cardigan: A generative adversarial network model for design and discovery of multi principal element alloys. *Journal of Materials Science Technology*, 125, 2022. 6

[6] Masoud Mahdianpari, Bahram Salehi, Mohammad Rezaee, Fariba Mohammadimanesh, and Yun Zhang. Very deep convolutional neural networks for complex land cover mapping using multispectral remote sensing imagery. *Remote Sensing*, 10, 2018. 5

[7] Mohammad Shahjahan Majib, MD. Mahbubur Rahman, T. M. Shahriar Sazzad, Nafiz Imtiaz Khan, and Samrat Kumar Dey. Vgg-scnet: A vgg net-based deep learning framework for brain tumor detection on mri images. *IEEE Access*, 9, 2021. 5

[8] Diganta Misra. Mish: A self regularized non-monotonic activation function. 2020. 1, 3, 6, 12

[9] Nicolas W. Remerscheid, Alexander Ziller, Daniel Rueckert, and Georgios Kaissis. Smoothnets: Optimizing cnn architecture design for differentially private deep learning. 2022. 1, 2, 12

[10] Teena Sharma, Nishchal K. Verma, and Shahrukh Masood. Mixed fuzzy pooling in convolutional neural networks for image classification. *Multimedia Tools and Applications*, 82, 2023. 1, 2, 12

[11] Yanan Sun, Bing Xue, Mengjie Zhang, Gary G. Yen, and Jiancheng Lv. Automatically designing cnn architectures using the genetic algorithm for image classification. *IEEE Transactions on Cybernetics*, 50, 2020. 4

[12] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, and Jonathon Shlens. Rethinking the inception architecture for computer vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5

[13] A. Helen Victoria and G. Maragatham. Automatic tuning of hyperparameters using bayesian optimization. *Evolving Systems*, 12, 2021. 1, 2

[14] Zhihe Xun, Liang Wang, and Yunqing Liu. Improved face detection algorithm based on multitask convolutional neural network for unmanned aerial vehicles view. *Journal of Electronic Imaging*, 31, 2023. 6, 7

[15] Jie Yang, Junhong Zhao, Lu Lu, Tingting Pan, and Sidra Jubair. A new improved learning algorithm for convolutional neural networks. *Processes*, 8, 2020. 1, 2, 12

[16] Tao Yang, Yadong Wei, Zhijun Tu, Haolun Zeng, Michel A. Kinsy, Nanning Zheng, and Pengju Ren. Design space exploration of neural network activation function circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 38, 2019. 6

[17] Huo Yingge, Imran Ali, and Kang-Yoon Lee. Deep neural networks on chip - a survey. *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, 2020. 4

[18] Yuan Zhou, Dandan Li, Shuwei Huo, and Sun-Yuan Kung. Shape autotuning activation function. *Expert Systems With Applications*, 171, 2021. 6