SMU Data Science Review

Volume 2 | Number 2 Article 1

2019

Improve Image Classification Using Data Augmentation and Neural Networks

Shanqing Gu
Southern Methodist University, shanqingg@smu.edu

Manisha Pednekar
Southern Methodist University, mpednekar@smu.edu

Robert Slater
Southern Methodist University, rslater@smu.edu

Improve Image Classification Using Data Augmentation and Neural Networks

Shanqing Gu, Manisha Pednekar and Robert Slater

Master of Science in Data Science
Southern Methodist University
Dallas, Texas 75275 USA
{shanqingg, mpednekar, rslater}@smu.edu

Abstract. In this paper, we present how to improve image classification by using data augmentation and convolutional neural networks. Model overfitting and poor performance are common problems in applying neural network techniques. Approaches to bring intra-class differences down and retain sensitivity to the inter-class variations are important to maximize model accuracy and minimize the loss function. With CIFAR-10 public image dataset, the effects of model overfitting were monitored within different model architectures in combination of data augmentation and hyper-parameter tuning. The model performance was evaluated with train and test accuracy and loss, characteristics derived from the confusion matrices, and visualizations of different model outputs with non-linear mapping algorithm t-Distributed Stochastic Neighbor Embedding (t-SNE). As a macro-architecture with 16 weighted layers, VGG16 model is used for large scale image classification. In the presence of image data augmentation, the overall VGG16 model train accuracy is 96%, the test accuracy is stabilized at 92%, and both the results of train and test losses are below 0.5. The overall image classification error rate is dropped to 8%, while the single class misclassification rates are less than 7.5% in eight out of ten image classes. Model architecture, hyper-parameter tuning, and data augmentation are essential to reduce model overfitting and help build a more reliable convolutional neural network model.

1 Introduction

In machine learning, image classification is a process to analyze the extracted image features and organize them into categories by using neural networks. In recent years, the neural network techniques have improved image classification accuracy quickly [1], such as AlexNet can achieve 15.3% image classification error rates[1]. These techniques are practically applied in many fields, such as artificial intelligence, medical diagnostics, E-commerce, gaming or automotive industries.

The convolutional neural network (CNN) is one of the most popular deep neural network (DNN) learning algorithms which can perform classification tasks directly from images. CNN models can produce the state-of-the-art classification

[1] AlexNet. https://en.wikipedia.org/wiki/AlexNet

predictions with directly learned features. A CNN model generally contains more than one convolutional layer for specialized linear operations and includes local or global pooling layers to perform nonlinear downsampling. After learning features from many layers, a fully connected layer outputs the probabilities for each class to be predicted. For example, these approaches were used in electrocardiogram classification to differentiate 12 rhythm classes for arrhythmia detection [2].

However, model overfitting and poor performance are common problems in applying neural network techniques because some of the high frequency features may not be useful in classification [3, 4]. Approaches to bring intra-class differences down and retain sensitivity to the inter-class variations are important to maximize model accuracy and minimize the loss function.

Generally, some strategies are proposed to reduce overfitting in refining model architecture perspective [5]: (1) a pooling layer between successive convolutional layers is periodically inserted to progressively reduce the spatial size of the representation and the number of parameters; (2) a favorable regularization type is selected to introduce additional information; (3) a dropout layer is used to avoid neuron interactions and learn more robust features in all training data [6]; (4) early stopping or limiting the number of parameters by filter size.

Besides these approaches, data augmentation is one of the important image preprocessing techniques to generate more training data and reduce model overfitting [7]. In this study, the image data from the CIFAR-10 (Canadian Institute For Advanced Research, CIFAR)[2] public image dataset is preprocessed with multiple data augmentation strategies. CNN model architectures are further refined to monitor the effects of model overfitting with optimal convolutional layers, size and number of the convolutional filters, optimizer and activation argument. The model performance was evaluated with train and test accuracy and loss, characteristics derived from the confusion matrices, and visualizations of different model outputs.

These approaches are finally applied to the VGG16 model, a CNN architecture developed as one of the complicated DNN models by Simonyan and Zisserman for large scale image recognition. It was named after the Visual Geometry Group (VGG) from Oxford in 2014 [8]. The VGG16 model is a 16-layer neural network without counting the max pooling layer and the *Softmax* layer. In this study, the model overall train accuracy for the VGG16 model with data augmentation is 96%, the test accuracy is stabilized at 92%, and both the results of train and test losses are below 0.5. The overall image classification error rate is dropped to 8%, while the single class misclassification rates are less than 7.5% in eight out of ten image classes. We conclude that it is essential to reduce mode overfitting and help build a more reliable CNN model with model architecture adjustment, hyper-parameter tuning and data augmentation.

[2] The CIFAR-10 dataset. https://www.cs.toronto.edu/ kriz/cifar.html

2 Background on Image Processing and Neural Networks

2.1 Image Data Processing

For a color image, it includes three channels per pixel whose color is stored as a number in a matrix. Image files can be kept either in raster image or vector image format. Raster image is made of pixels with the common formats like *jpeg, png, gif, tif,* or *bmp*. Comparatively, vector image is made of vector objects and can be scaled to any sizes. The common formats for vector image include *ai, eps, ps, svg, wmf,* and *pdf*. Many file formats contain both vector and raster images.

For an 8-bit true color image, any colors can be defined with red (R), green (G) and blue (B) values. Generally, any RGB colors can be expressed from 0 (least saturated) to 255 (most saturated). In RGB color model, a broad array of color can be reproduced with adding three colors differently. For example, the purple color can be expressed in decimal code (R, G, B) as rgb(128, 0, 128), and the violet color as rgb(238, 130, 238)[3]. With this system, there are 256x256x256 discrete color combinations.

The colormap RGB values can be divided by the maximum value 255 in order to convert the 0-255 system into the 0-1 system. The three-color components are stored in a 3D matrix of numbers. A 2D matrix can be convolved with different filters, such as filters for edge detection. As shown in Figure 1, each of these filters is used to convolve with one of the channels and produce three outputs correspondingly [9].

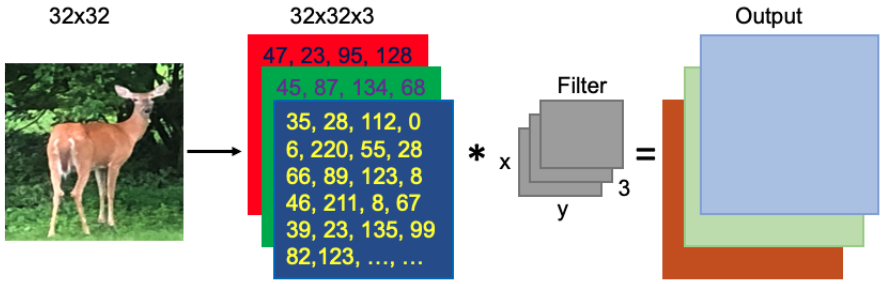

Fig. 1. A color image is represented as a 3D matrix of numbers (e.g. 32x32x3). The third dimension has three channels for red, green and blue colors. A 2D matrix can be convolved with one filter. Each of these filters is convolved with one of the channels, and three outputs are produced.

[3] RGB color codes. https://flaviocopes.com/rgb-color-codes/

2.2 Image Data Augmentation

For image preprocessing, a single image is represented with an array of pixels using grey scales or RGB values. The image data should be scaled with min-max normalization to increase the speed of learning. The categorical feature color can be transformed into a vector of three numerical values with one-hot encoding.

Data augmentation is one of the important image preprocessing techniques which can be conducted offline or online [7]. Offline augmentation techniques are used for increasing the size of small datasets, while online augmentation techniques are mainly applied to large datasets. Due to the long time for calculating the model gradients by using the entire large dataset, the data with online image augmentation techniques are looped over in mini-batches during each optimizer iteration with the selected batch size.

Image data augmentation techniques generate more training data from original data and require no additional memory for storage[4]. Generally, the generated images are small batched and discarded after model training. The common techniques to generate new images are: (1) flip horizontally or vertically; (2) rotate at some degrees; (3) scale outward or inward; (4) crop randomly; (5) translate; (6) add Gaussian noises to prevent overfitting and enhance the learning capability. In addition, there are also advanced augmentation methods by using conditional generative adversarial networks (GANs). Representative examples are shown in Figure 2.

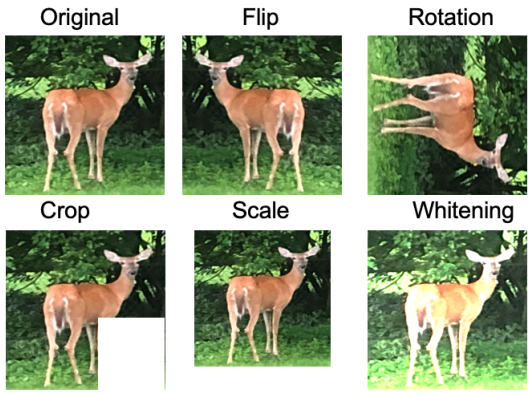

Fig. 2. Representative images with data augmentation techniques, such as flipping, rotating, cropping, scaling, and whitening. Data augmentation techniques are used to generate more training data. The generated images are small batched and discarded after model training process.

[4] Image Preprocessing. https://keras.io/preprocessing/image/

2.3 Convolutional Neural Networks

A neural network is typically composed of multiple layers with interconnected neurons [10]. This architecture is analogous to the connectivity pattern of neurons in the human brain and was inspired by the organization of the visual cortex [11]. Inside a neuron, there are several key components to compute the weighted sum of the inputs. A layer is defined as a container of neurons. Compared with the hidden layer, the input layer and the output layer are two special types of layers. The input layer is to accept input image data, while the output layer is to generate output values of the neural network.

CNN and DNN are two most common approaches for image classification. As one kind of extremely popular windowed and weight-shared DNN [1], CNN can take in an input image, assign weights and biases to various aspects in the image, and then differentiate one image from the other. CNN is characteristic of containing convolutional layers and pooling layers in architecture (Figure 3). The convolutional layers are the core building blocks of a CNN model. The learned filters are applied systemically to input images with these convolutional layers to extract important features. Comparatively, the pooling layers are new layers added after the convolutional layers to create down-sampled feature maps [12].

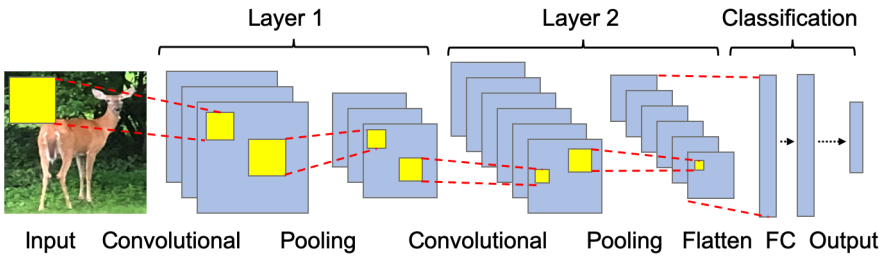

Fig. 3. An example of CNN architecture. It contains convolutional, pooling and fully-connected layers for regression. The yellow square is to show the example filter (a.k.a. convolutional kernel or K). The values from the output node represent the classified physical objects, such as deer.

For convolutional operation [13], the filter (a.k.a. kernel or K) can be seen as a rectangular patch moving through the full image from direction either left to right or top to bottom (shown as yellow squares in Figure 3). The input image is transformed into a feature map which represents what the filter has learned from this operation. These feature maps are further transformed into other feature maps in the succeeding layers. The filter argument controls the number of feature maps generated per convolutional layer. For multiple color channels (e.g. RGB), the filter has the same depth as that of the input image. Matrix multiplication needs to be performed afterwards. All the results are summed up with the bias to provide convoluted feature outputs. The final feature map is smaller than the

original input image since only the valued features are extracted in convolutional operations.

Stride and padding are two important properties of convolutional operation. Stride is defined as the number of pixels shifting over the input matrix. In other words, stride specifies how much to move the convolution filter at each step. The default value for stride is set to 1. Padding is the width of the square of additional cells. The cells added by padding usually contain zeros. For example, the 4x4 image can be converted to 5x5 image with the same padding.

Typically applied after each convolutional layer, a nonlinear activation argument or layer is applied from various activation argument types (*elu, selu, relu, tanh, sigmoid, hardsigmoid, softplus, softsign, linear*, and *exponential*)[5]. The nonlinear rectified linear unit *relu* tends to be the default activation for deep networks due to its simplicity and ability to enable fast training. The activation arguments *elu* and *selu* are the two variants of *relu*.

The pooling layer is periodically inserted into CNN architecture [13]. Its main function is to reduce image dimension by taking the maximum pixel value of a grid and compress each feature map. *Max* is the most common downsample pooling operation. The output of last pooling operation is a stack of feature maps (Figure 3). When stride value is set to 2, the input feature map size is reduced 50% for the output feature map size.

2.4 Build CNN with the Keras Neural Network Library

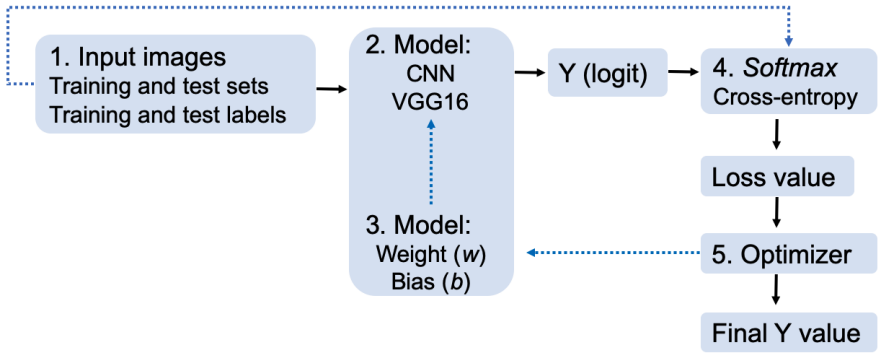

Fig. 4. Build CNN with Keras neural network library. (1) The images are taken as input with labels for training and test datasets, (2) the CNN model (e.g. VGG16) is selected to build the model architecture, (3) collect the weight matrix and bias vector from each layer, (4) get the loss results calculated from *Softmax* with cross-entropy function, (5) optimize the weights and update the calculated loss results with back-propagation.

[5] Usage of activations. https://keras.io/activations/

Keras neural network library[6] is to provide high-level building blocks for developing deep learning models. There are three backend implementations for Keras: TensorFlow, Theano, and CNTK. In this paper, we use the Keras neural network library with TensorFlow backend and build the Sequential model with a linear stack of layers to the constructor. As shown in (Figure 4), the Sequential model is built with these components:

(1) the training and test datasets with their labels. The input shape and a fixed batch size should be specified. The batches in small sizes are used to update the errors during each epoch instead of waiting until all train images are evaluated.

(2) the model with all of its layers (e.g. 16 layers for model VGG16). Besides convolutional and pooling layers, there are other layers to be required. For examples: (i) the dropout layer is a form of regularization that makes neural networks more robust. The key idea is to randomly drop units along with their connections from the neural networks during the training process. The dropout layer is not used in the output layer and for the predictions on test data; (ii) the BatchNormalization layer applies a transformation that maintains the activation mean close to 0 and the activation standard deviation to 1. It acts as a regularizer to prevent internal covariate shift; (iii) the fully connected layer is to convert the stack of feature maps into a vector format which is suitable for either a dropout or dense layer.

(3) the weight matrix (w) and bias vector (b) for each layer. The connections from the neurons or nodes are weighted, and the bias nodes function like an input node that always produces constant value 1 or another constant[7].

(4) the loss results calculated from *Softmax* with cross-entropy function. Activation argument *Softmax* in SGD momentum within the output layer is differentiable to train by gradient descent. After applying *Softmax* function, each component is in the interval (0, 1) and adds up to 1. Therefore, it is used to output the non-normalized neural network vector that represents the probability distributions of a list of potential outcomes [14].

(5) the optimizer is to use the calculated loss results to update the weights and bias with back-propagation. The optimizer is another argument required to compile a Keras neural network model. In general, there are seven optimizers (*SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax,* and *Nadam*)[8] to select.

In addition, image data have to be resized first into the same dimensions for neural network training. Pixels are usually standardized and then normalized to the range [0,1]. Besides L1 and L2 regularization types in neural networks, other specific regularizers (e.g. dropout, early stopping, batch-normalization) can be applied. Data augmentation is also considered as regularization technique for neural networks.

[6] Keras: The Python Deep Learning library. https://keras.io
[7] http://www.uta.fi/sis/tie/neuro/index/Neurocomputing2.pdf
[8] Usage of optimizers. https://keras.io/optimizers/

3 Image Dataset and Data Preprocessing

3.1 CIFAR-10 Dataset

In this paper, we use public image classification dataset CIFAR-10 which was initially released in 2009 [15]. The standardized CIFAR-10 dataset contains 50,000 train and 10,000 test images at the spatial resolution of 32x32 color pixels from ten different classes[9].

In Figure 5, the representative images from this dataset are shown. Each class corresponds to different physical objects (i.e. airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck). There are five training batches and each batch contains exactly 5,000 images from each class. The test batch contains exactly 1,000 randomly-selected images from each class. This indicates that all classes are completely mutually exclusive and class-balanced. The image data can be imported directly from Keras datasets[10].

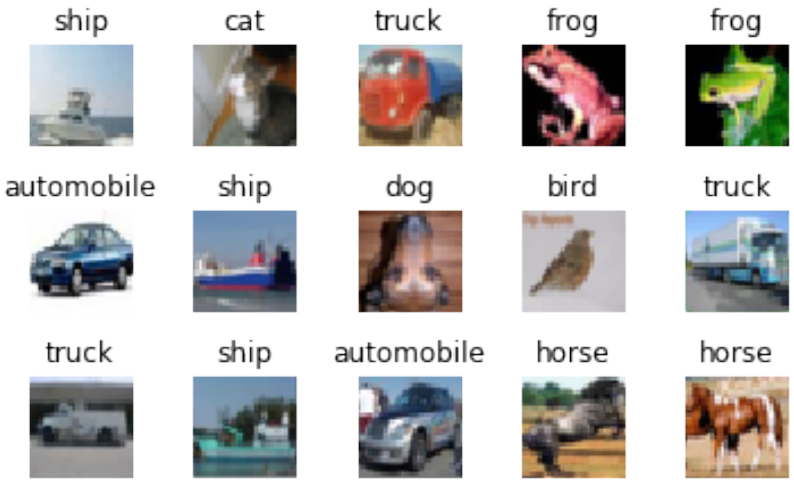

Fig. 5. Example images from the CIFAR-10 dataset. Each image is made up of 32x32x3 pixels with three RGB colors. This dataset consists of 60,000 color images in ten equal classes corresponding to different physical objects (i.e. airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck).

[9] The CIFAR-10 dataset. https://www.cs.toronto.edu/ kriz/cifar.html
[10] from keras.datasets import cifar10

3.2 Data Preprocessing with Data Augmentation

Data augmentation techniques are often used to regularize models which work with images in neural networks and other learning algorithms. With the labelled original training dataset, synthetic images can be created by various transformations to the original images.

Keras ImageDataGenerators[11] is the tool used for generating more training data from the original data to avoid model overfitting. It is conducted online by looping over in small batches during each optimizer iteration. There are some graphic parameters (e.g. rotation, shift, flip, add Gaussian noises) to help generate artificial images. In Figure 6, some example images generated by using the ImageDataGenerator are shown. Importantly, these images are not included in the original CIFAR-10 image dataset. These generated new image data are mini-batched and discarded after model training.

There are various data augmentation techniques: (1) flipping images horizontally or vertically; (2) rotating images at some degrees; (3) rescaling outward or inward; (4) randomly cropping; (5) translating by width and height shifts; (6) whitening, (7) shearing, (8) zooming and (9) adding 10% Gaussian noises to prevent model over-fitting and enhance learning capability.

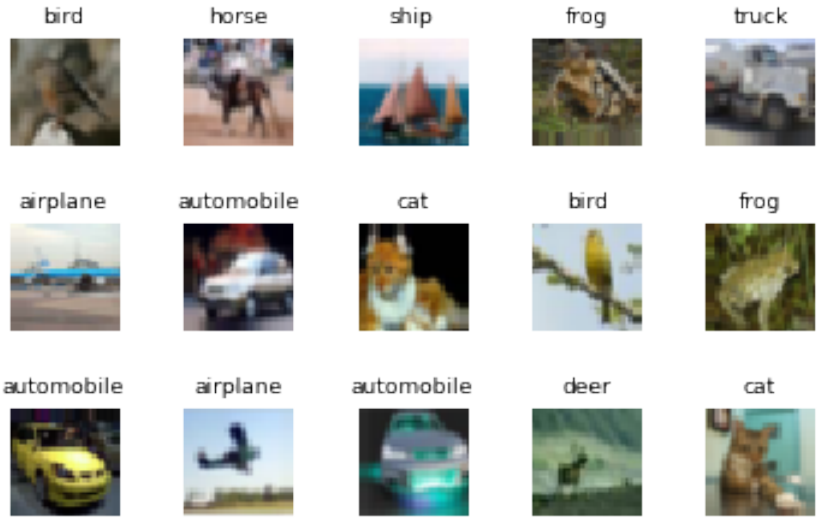

Fig. 6. Example images artificially generated with the CIFAR-10 dataset by using ImageDataGenerator. Data augmentation techniques create more new training data which are mini-batched and discarded after model training.

[11] Image Preprocessing. https://keras.io/preprocessing/image/

4 Model Developments

4.1 Model Selection

We start with a basic CNN model with two convolutional layers, one max-pooling layer, and one classifier layer for image classification. Secondly, the standard CNN models are built with six convolutional layers and one dense layer (a VGG16-like model). Before each convolutional layer, a batch normalization layer is applied. The dropout layers are also added between convolutional layers. The final fully connected *Softmax* layer produces a probability distribution over ten predicted output classes. Thirdly, VGG16 models were applied with optimized architectures, hyper-parameter tuning, and data augmentation techniques.

4.2 Hyperparameter Tuning

Model hyper-parameters are usually fixed settings and must be tuned in order to obtain optimal model performance. Comparatively, model parameters are fitted parameters that must be determined using the training dataset. Generally, 20% of train images are used as validation dataset for hyper-parameter selection except test dataset is used. The optimizer and activation argument applied to CNN models are screened from collections of candidates [16]. The optimizer *Adamax* and activation argument *elu* are selected after evaluating the accuracy and loss results from training, validation and test datasets. The learning and decay rates are also adjusted during the training process.

4.3 Model Evaluation

The model performance was evaluated with accuracy and loss function for the training, validation and test datasets in the absence or presence of applying multiple data augmentation techniques. The characteristics derived from confusion matrices are used to compare individual class prediction.

4.4 Model Output Visualization

Different model outputs in high dimensions are visualized with t-Distributed Stochastic Neighbor Embedding (t-SNE) [17]. t-SNE is designed for non-linear mapping which is based on probability distribution. For visualizing model outputs of neural networks, the extracted features can be visualized effectively in clusters by the t-SNE algorithm in two dimensions.

4.5 Model Implementation Environment

In Python (version 3.6) environment, Keras with TensorFlow backend was used to build up and train a CNN or DNN model. Intel® 8th GEN^{TM}CORE-i7-8750H processor, Graphic card NVIDIA GPU (GeForce GTX 1070) with cuDNN 7.3.1 for CUDA 10.0) and 16 GB DDR4-2666MHz of RAM were used to perform computations.

5 Results

5.1 Basic CNN Model M0 for Image Classification

A basic CNN Sequential model M0 is built to linearly stack the hidden and output layers for image classification. It consists of two convolutional layers (Conv1 and Conv2) followed by one flatten layer and one dense layer. The graphic representation of basic CNN model M0 is shown in Figure 7. Conv1 and Conv2 have the same input space of 32x32x3, while the pooling layer has the space size of 16x16x32 after dimensionality reduction. The flatten layer is used to adjust the feature maps to the dense layer. The output layer has ten neurons with one for each target class.

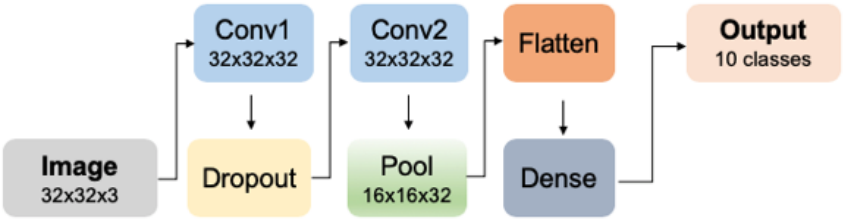

Fig. 7. Basic CNN model M0 architecture. The shapes of 3D feature maps are passed between layers and their connections. Conv1 and Conv2 have the same space size of 32x32x32, while the pooling layer has the space size of 16x16x32 after dimensionality reduction. The flatten layer is used to adjust the feature maps to the dense layer. The output layer has ten neurons with one for each target class.

In Figure 8, this basic M0 model performance is evaluated with train and test accuracy and loss curves. These curves are plotted together for better comparison. Accuracy is defined as the ratio of correctly predicted observation to the total observations, and loss is the categorical cross-entropy to predict class probabilities (i.e. the difference between the predicted value and the true value). The basic CNN model M0 is highly overfitted even after several epochs of model training. The train accuracy for model M0 is 29% higher than the test accuracy. The two loss curves are also widely separated. This suggests that more layers and neurons should be added into the CNN architecture for better model performance. Some regularizations should also be employed, such as dropout layers, kernel regularizers, and BatchNormalization layers.

Image misclassification happens when an image is incorrectly predicated with higher probability of other classes. For example, the cat in the first image is misclassified as a dog (Figure 9).

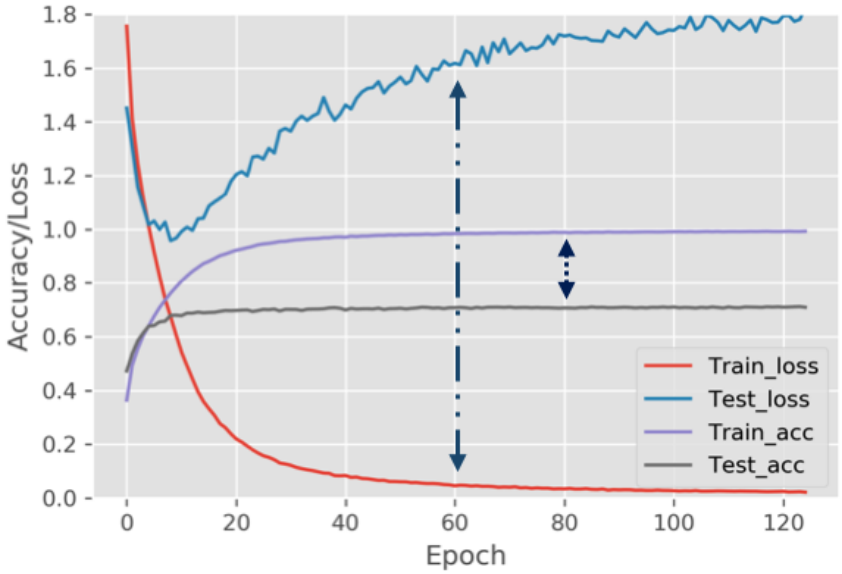

Fig. 8. Study basic M0 model performance with accuracy and loss plots. This basic CNN model used for CIFAR-10 image classification is highly overfitted, as indicated by the much lower test accuracy than the train accuracy together with the significant gap between these two loss curves. The gaps between accuracy and loss curves are shown with dotted and double-headed vertical arrows.

Fig. 9. Misclassified image samples by the basic CNN model M0. The upper panel y is to show the ground truth image and the lower panel p is to show the corresponding predicted image. This image misclassification is caused by model overfitting.

5.2 VGG-like CNN Model M-DA-0 Without Data Augmentation

For studying the effects of data augmentation, the VGG-like model M-DA-0 is built with six convolutional layers. As shown in Figure 10, model M-DA-0 is a three stacks of two convolutional layers (Conv1-6) followed by three max-pooling (Pool1-3) layer and the final dense layer. The three filter numbers applied in the stacking convolutional layers are 32, 64, and 128. An activation function, such as *elu*, takes the feature map (i.e. tensor) generated by the convolutional layer and creates the activation map as its output. Batch normalization is normally used to normalize the input layer by adjusting and scaling the activation function. The flatten layer is used to adjust the tensors to the dense layer with the *Softmax* layer as the output layer.

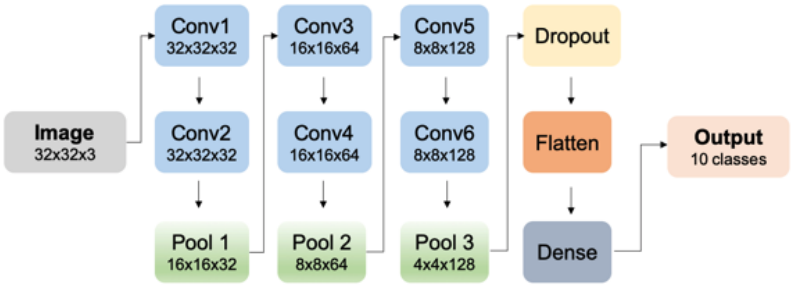

Fig. 10. A standard CNN model M-DA-0 architecture for image classification. This model pattern is three stacks of two convolutional layers (Conv1-6) followed by three max-pooling (Pool1-3) layer. The three filter numbers applied in the stacking convolutional layers slide from 32 to 64 and 128 separately. The flatten layer is used to adjust the feature maps to the dense layer with the probability output for ten classes.

Similar to basic M0 model, the M-DA-0 model performance is studied also with the accuracy and loss plots (Figure 11). The train accuracy is 11% higher than the test accuracy, although this gap is less than the 29% of the two curves in model M0 (Figure 8). The two loss curves for training and test datasets are still widely separated, as compared to model M0 (Figure 8). Overall, the model overfitting problems have been improved significantly, but still exist in model M-DA-0 which is more complicated in architecture than the basic M0 model.

5.3 Use Model M-DA-0 to Select Favorable Optimizers

The model overfitting problem in model M-DA-0 has been reduced significantly when compared with model M0, but it still needs hyper-parameter tuning before implementation. Seven different optimizers (*SGD, RMSprop, Adagrad, Adam, Adamax, Adadelta*, and *Nadam*) are compared with their contributions to model

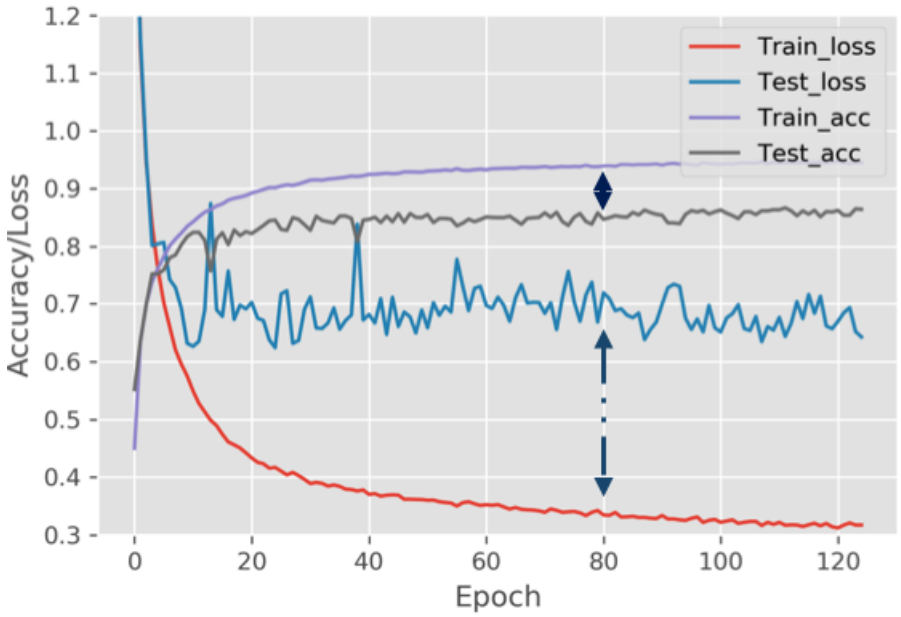

Fig. 11. Study the standard CNN model M-DA-0 performance with accuracy and loss plots. The gaps between the train and test accuracy plots indicate model M-DA-0 is overfitted. The test accuracy is lower than the train accuracy, and the two loss curves are widely separated. The gaps between accuracy and loss curves are marked with dotted and double-headed vertical arrows.

performance[12]. All optimizers are applied with step learning rates (initialized with 0.001, 0.0005 for epochs more than 75, 0.0003 for epochs more than 100) and weight decay rate (constant at 0.0001).

The results for accuracy and loss are summarized in Table 1 and plotted in Figure 12 for train accuracy, Figure 13 for train loss, Figure 14 for test accuracy, and Figure 15 for test loss. The top four optimizers with high model performance are *Adam, Adamax, RMSprop*, and *Nadam*. *Adam* uses the squared gradients to scale the learning rate and moves the average of gradients. *Adamax* and *Nadam* are two variants of *Adam*. *Adamax* is selected into further model development because of the high train accuracy at 96% and test accuracy at 86%. The test loss is 0.69 while the train loss is 0.18. The 10% gap between the train accuracy curves indicates further hyper-parameter optimizations are necessary.

In addition, optimizer *Adagrad* brings less model overfitting as indicated by the same 70% train and validation accuracies, while the test accuracy is close to 72%. Comparatively, *Adadelta* performs the worst among all seven optimizers.

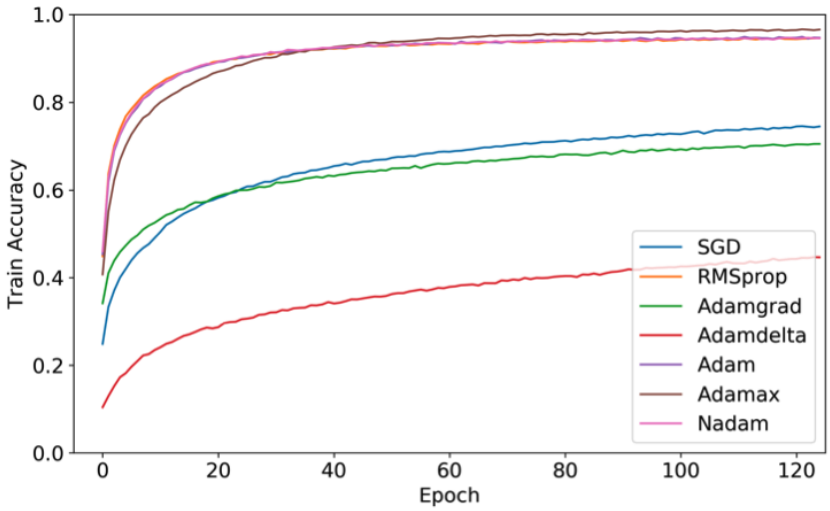

Fig. 12. Compare seven different optimizers with train accuracy in model M-DA-0 architecture. The results for train accuracy are shown in Table 1. The top four optimizers with high model performance are *Adam, Adamax, RMSprop*, and *Nadam*.

[12] Usage of optimizers. https://keras.io/optimizers/

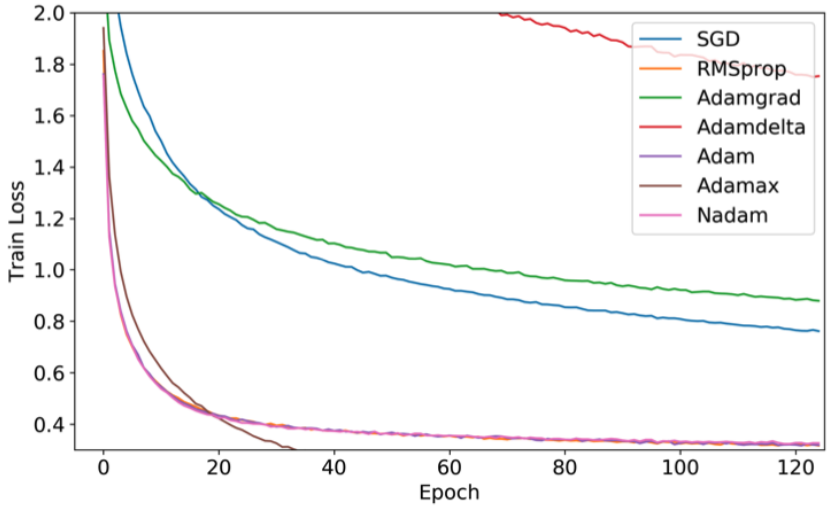

Fig. 13. Compare seven different optimizers with train loss in model M-DA-0 architecture. The results for train loss are shown in Table 1. Optimizer *Adamax* minimizes the loss function most.

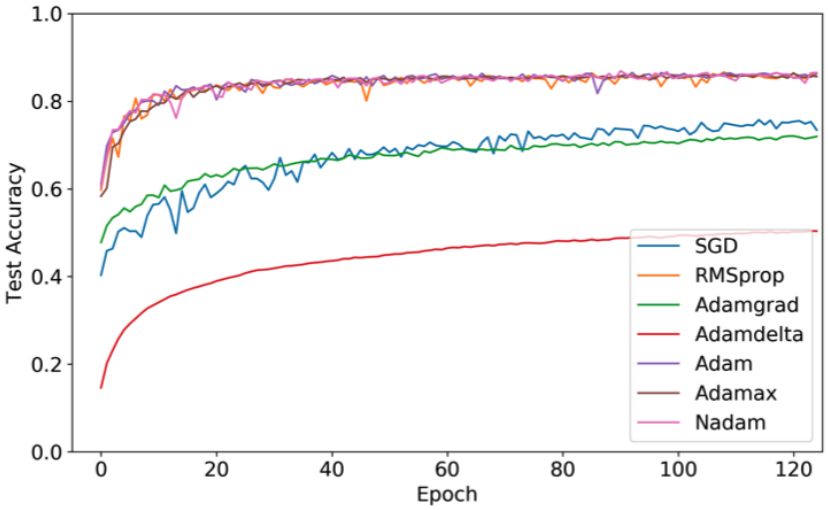

Fig. 14. Compare seven different optimizers with test accuracy in model M-DA-0 architecture. The results of test accuracy are shown in Table 1. The top four optimizers with high model performance are *Adam, Adamax, RMSprop*, and *Nadam*.

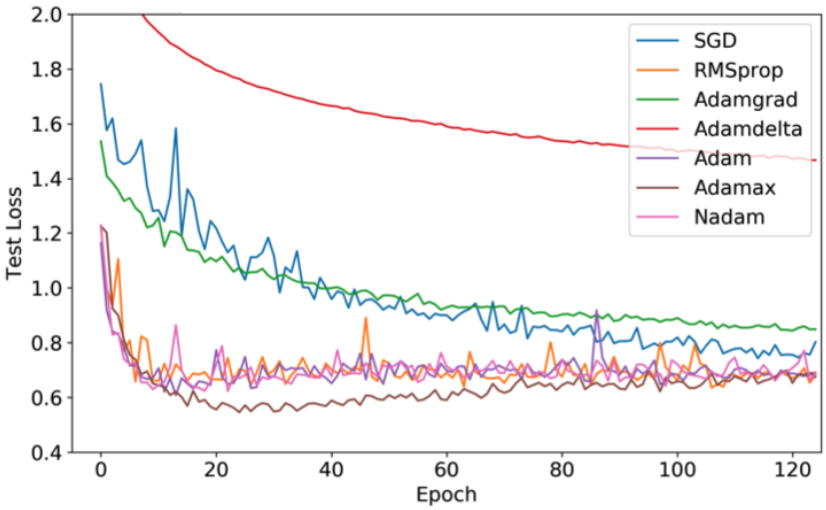

Fig. 15. Compare seven different optimizers with test loss in model M-DA-0 architecture. The results of test loss are shown in Table 1. Optimizer *Adamax* minimizes the loss function most.

Table 1. Summary of model M-DA-0 accuracy and loss using different optimizers

Optimizer	Train Acc.	Train Loss	Val. Acc.	Val. Loss	Test Acc.	Test Loss
1. SGD	0.7458	0.7612	0.8094	0.5841	0.7346	0.8023
2. RMSprop	0.9466	0.3129	0.8512	0.7293	0.8571	0.6808
3. Adagrad	0.7050	0.8768	0.7035	0.8994	0.7196	0.8488
4. Adam	0.9482	0.3159	0.8482	0.7779	0.8637	0.6742
5. Adamax	0.9629	0.1818	0.8508	0.7199	0.8566	0.6896
6. Adadelta	0.4466	1.7542	0.5462	1.3672	0.5036	1.4676
7. Nadam	0.9455	0.3268	0.8600	0.7270	0.8652	0.6916

5.4 Optimize Activation Argument Using Model M-DA-1

Compared with model M-DA-0, training with the augmented data in model M-DA-1 requires the training data to pass through a Keras ImageDataGenerator. These data are looped over in mini-batches indefinitely.

Different image data augmentation techniques and parameters are summarized in Table 2. Briefly, M-DA-0 is used as the control model without data augmentation. General techniques like rotation, horizontal flip and translation are used in all other three models M-DA-1 to -3. The Gaussian noises are added only into model M-DA-2, and other additional techniques (e.g. rescale, shear, or zoom) are only used in model M-DA-3.

Table 2. Summary of ImageDataGenerator parameters in four standard CNN models

Model	ImageDataGenerator Parameters
1. M-DA-0:	No data augmentation
2. M-DA-1:	rotation 15, width shift 0.1, height shift 0.1, horizontal flip
3. M-DA-2:	rotation 15, width shift 0.1, height shift 0.1, horizontal flip, zca epsilon: 1e-6, fill with nearest, Gaussian noise 0.1
4. M-DA-3:	rotation 45, width shift 0.2, height shift 0.2, horizontal flip rescale 1/255, shear 0.2, zoom 0.2, fill with nearest

The accuracy and loss results for are the ten available activation arguments (*elu, selu, relu, tanh, sigmoid, hardsigmoid, softlplus, softsign, linear*, and *exponential*)[13] are summarized in Table 3 and plotted in Figure 16 for train accuracy, Figure 17 for train loss, Figure 18 for test accuracy and Figure 19 for test loss. Nonlinear activation argument *elu* performs the best with the train accuracy at 90% which is close to the test accuracy 89%. This accuracy proximity happens in all observations of other nine activation arguments, indicating the data argumentation techniques, rather than the activation arguments, play important roles in model overfitting reduction.

The activation arguments *elu, selu, softplus* and *relu* are the top four arguments in improving model performance (Figure 16 and Figure 18). These accuracy curves are mostly overlapped. *Linear* performs the worst in both train and test accuracy plots. These observations indicate data augmentation techniques are important in preventing model overfitting. This is further supported with the loss function results and curves[14] (Table 3, Figure 17 and Figure 19).

Optimizer *Adamax* and activation argument *elu* are finally selected for further model development. In Figure 20, the train and test accuracy and loss are plotted together for better comparison. The overlapped accuracy and loss curves for training and test datasets indicate the significant model overfitting improvement after data augmentation and hyper-parameter tuning.

[13] Usage of activations. https://keras.io/activations/

[14] NA: Not available in computation

Table 3. Summary of model M-DA-1 accuracy and loss with activation arguments

Activation Arg.	Train Acc.	Train Loss	Val. Acc.	Val. Loss	Test Acc.	Test Loss
01. elu	0.9020	0.3411	0.8897	0.4090	0.8866	0.4188
02. selu	0.8953	0.3576	0.8782	0.4405	0.8789	0.4505
03. relu	0.8987	0.3484	0.8817	0.4308	0.8775	0.4407
04. tanh	0.8668	0.4335	0.8506	0.5158	0.8428	0.5372
05. sigmoid	0.8819	0.4011	0.8674	0.4756	0.8604	0.4917
06. hardsigmoid	0.8642	0.4466	0.8662	0.4695	0.8586	0.4957
07. softplus	0.9010	0.3441	0.8901	0.4361	0.8888	0.4223
08. softsign	0.8632	0.4496	0.8462	0.5360	0.8471	0.5264
09. linear	0.7585	0.7517	0.7776	0.7571	0.7769	0.7618
10. exponential	0.0997	NA	0.1002	NA	0.1002	NA

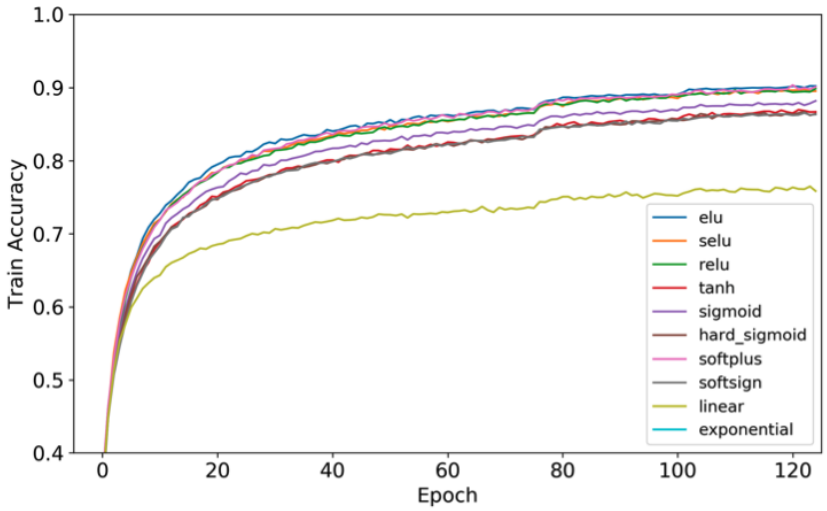

Fig. 16. Compare ten different activation arguments with train accuracy in model M-DA-1(for details about data augmentation, see Table 2). The activation argument *elu* performs the best and *linear* performs the worst (Table 3) among all candidates.

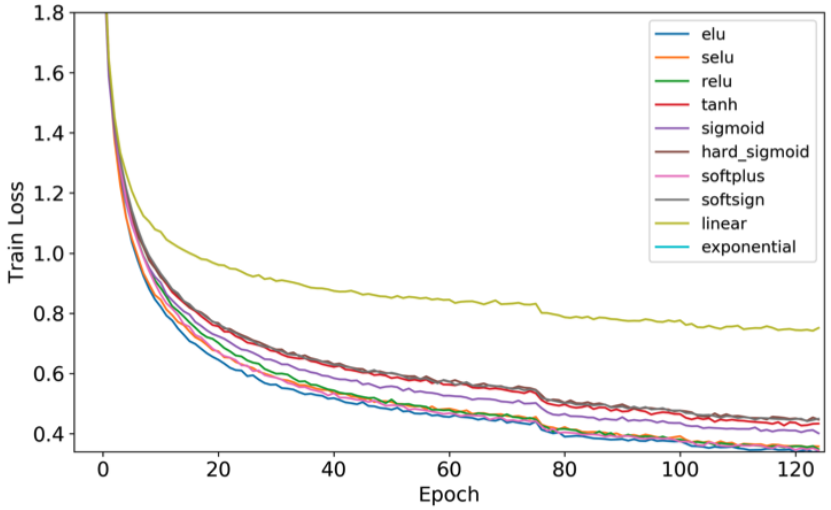

Fig. 17. Compare ten different activation arguments with train loss in model M-DA-1 (for details about data augmentation, see Table 2). The activation argument *elu* performs the best and *linear* performs the worst (Table 3) among all candidates.

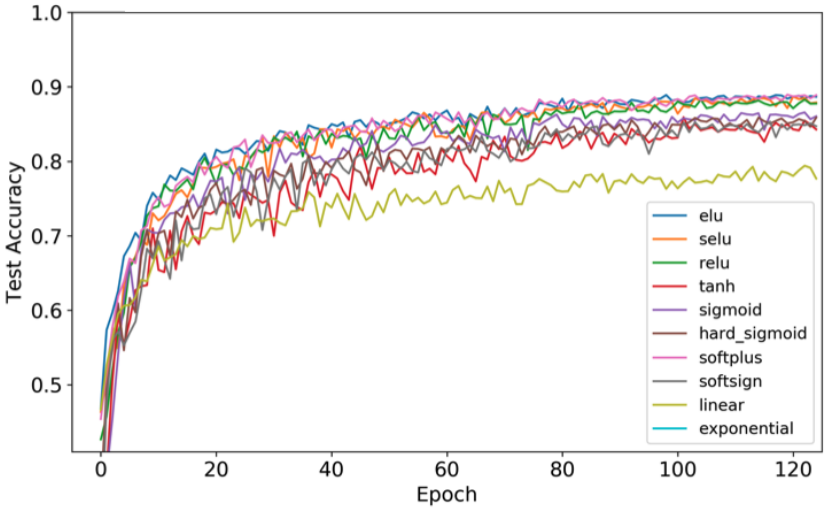

Fig. 18. Compare ten different activation arguments with test accuracy in model M-DA-1 (for details about data augmentation, see Table 2). The activation argument *elu* performs the best and *linear* performs the worst (Table 3) among all candidates.

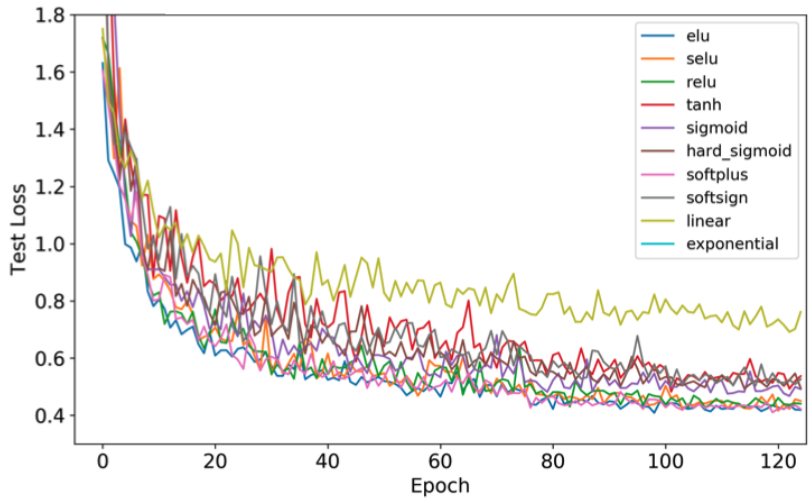

Fig. 19. Compare ten different activation arguments with test loss in model M-DA-1(for details about data augmentation, see Table 2). The activation argument *elu* performs the best and *linear* performs the worst (Table 3) among all candidates.

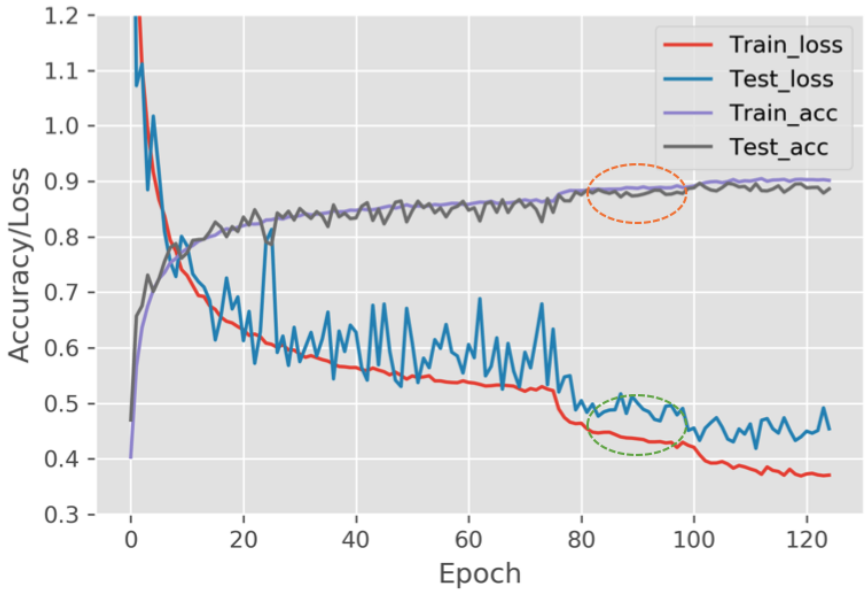

Fig. 20. Compare train and test accuracy and loss in model M-DA-1 with optimizer *Adamax* and activation argument *elu*. The train and test accuracy and loss traces are mostly identical in that there is less model overfitting after data augmentation. The tiny gaps for accuracy and loss curves are shown with orange and green dotted circles.

5.5 Data Augmentation with Gaussian Noises in Model M-DA-2

It is proposed that there are model improvements after adding Gaussian noises in data augmentation techniques. The idea is that adjusting the right amount of Gaussian noises can mitigate overfitting and enhance the learning capability. The Gaussian noises essentially have data points with the zero mean for all frequencies, effectively distorting the high frequency features.

In Figure 21, the train and test accuracy and loss plots for the M-DA-2 model are compared with the same optimizer *Adamax* and activation argument *elu* in architecture. The only difference, compared with the M-DA-1 model, is that extra 10% of Gaussian noises are added into model M-DA-2. Adding Gaussian noises has slight effects in improving train and test accuracies. The train accuracy is 90% and the test accuracy is almost the same at 89%. It can be explained that other data augmentation techniques leave little room for Gaussian noises to contribute, or *vice versa*. The last explanation seems more reasonable because of the smoothly early drop in loss curves.

5.6 Additional Data Augmentation Techniques in Model M-DA-3

In Figure 22, the M-DA-3 model contains additional and extreme data augmentation techniques (e.g. rescale, shear, zoom, 45% rotation) than in the M-DA-1 model (Table 2). The model accuracy results are unexpectedly low, while the loss function results are extremely high. These results indicate some data augmentation techniques are not helpful in improving model performance. For plots with M-DA models (Figure 11, Figure 20, Figure 21, and Figure 22), the results of train and test accuracy and loss are summarized in Table 4.

Table 4. Summary of model accuracy and loss in models M-DA-0, -1, -2, -3

Model	Train Acc.	Train Loss	Test Acc.	Test Loss
1. M-DA-0	0.9450	0.3175	0.8640	0.6427
2. M-DA-1	0.9016	0.3707	0.8863	0.4541
3. M-DA-2	0.9020	0.3382	0.8850	0.4229
4. M-DA-3	0.5600	1.2949	0.1536	13.5485

5.7 Confusion matrix tables for four models: M-DA-0, -1, -2, -3

A confusion matrix is used to study individual misclassification rates in 4 different models. In Figure 23 and Figure 24, columns represent true classes, and rows represent the classifier's predictions. The square matrix puts all correct classifications along the upper-left to lower right diagonal. The confusion matrix tables are to show 10-class image classification results for model M-DA-0 in the absence of data augmentation, and M-DA-1 in the presence of basic data

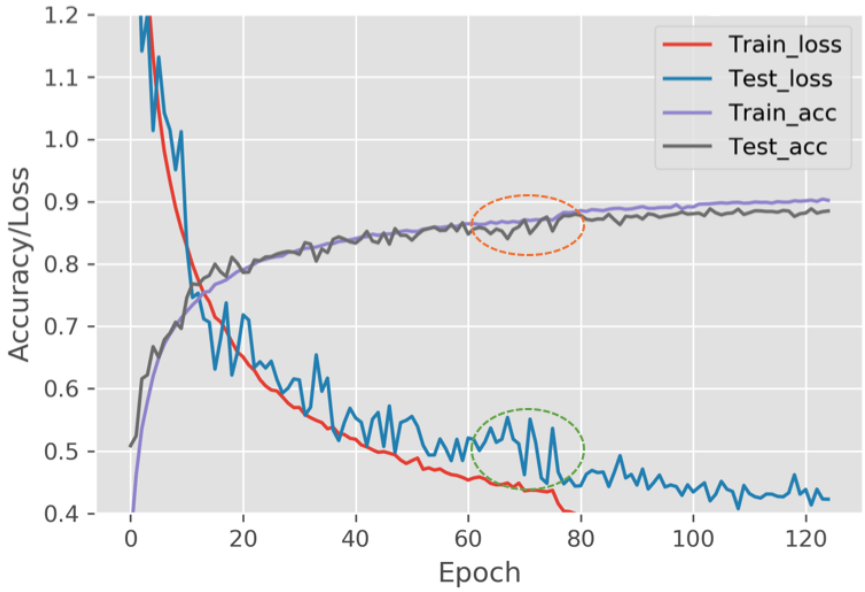

Fig. 21. Compare train and test accuracy and loss in model M-DA-2 with optimizer *Adamax* and activation argument *elu*. The model M-DA-2 includes extra 10% Gaussian noises as compared to model M-DA-1. The minimized gaps for nearly identical train and test accuracy and loss traces are shown with orange and green dotted circles.

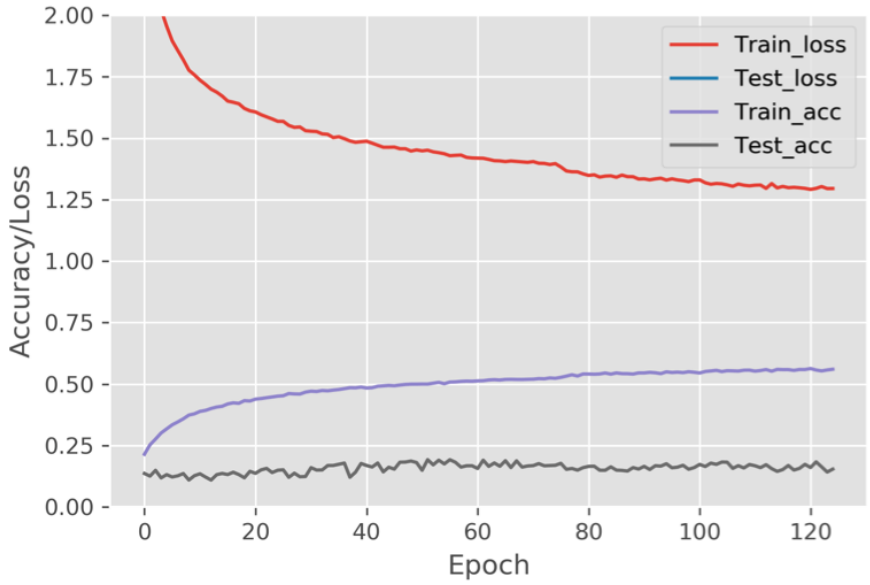

Fig. 22. Compare train and test accuracy and loss in model M-DA-3 with optimizer *Adamax* and activation argument *elu*. The M-DA-3 model includes additional and extreme data augmentation techniques than model M-DA-1. The model accuracy results are unexpectedly low, while the loss function results are extremely high. These results indicate some data augmentation techniques fail to improve model performance.

augmentation (for details about data augmentation, see Table 2). The image classification accuracy or misclassification error rate for individual, multiple or overall classes can be computed from these matrix tables. Compared with model M-DA-0, image classification error rates after applying data augmentation techniques in model M-DA-1 are dropped for all except three classes (airplane, bird and deer). Significantly, the image misclassification error rates in model M-DA-1 drop more than 20% for truck and automobile as compared with model M-DA-0.

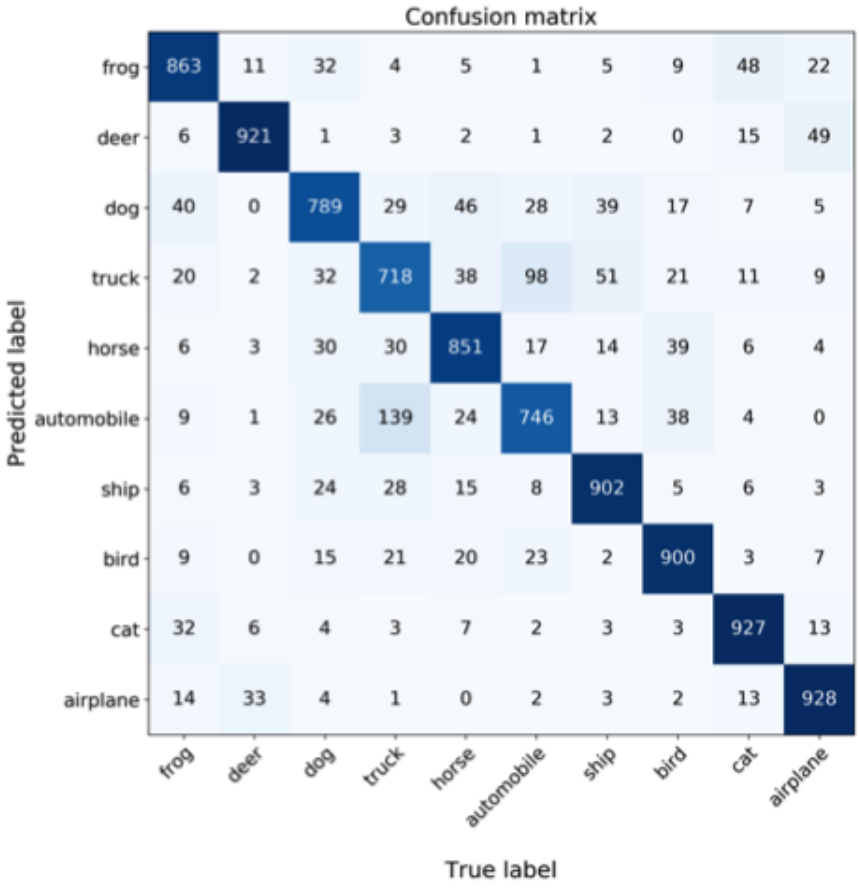

Fig. 23. Confusion matrix table for 10-class image classification using the M-DA-0 model without data augmentation. Columns represent true classes, and rows represent the classifier's predictions. The square matrix puts all correction classifications along the upper-left to lower right diagonal.

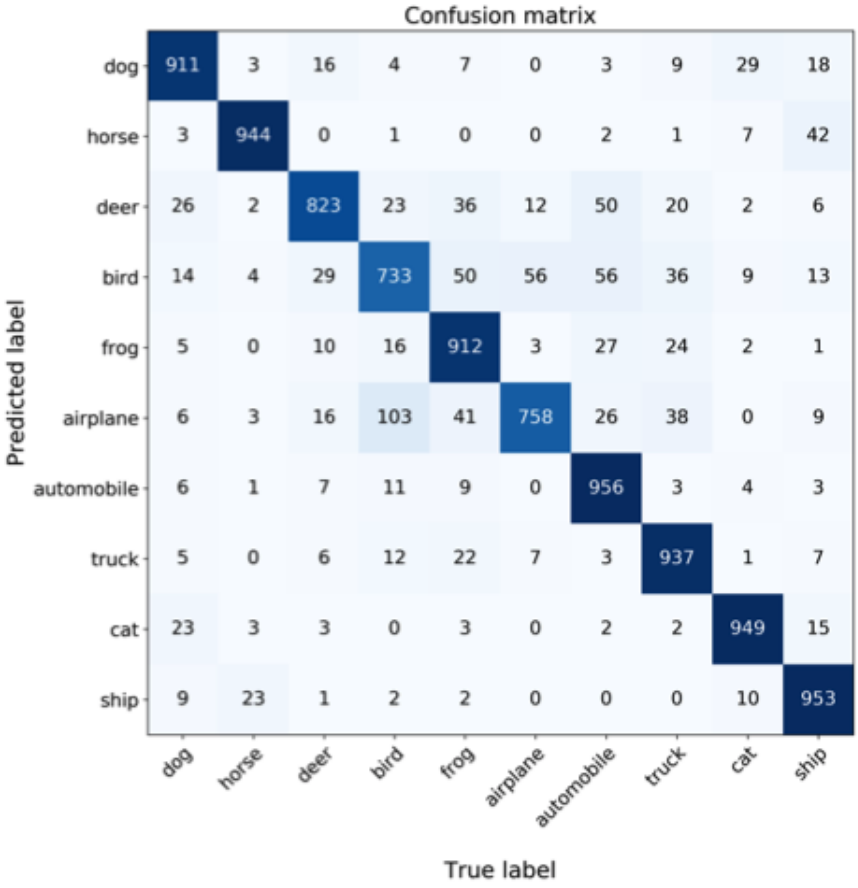

Fig. 24. Confusion matrix table for 10-class image classification using the M-DA-1 model with general data augmentation techniques. Columns represent true classes, and rows represent the classifier's predictions. The square matrix puts all correction classifications along the upper-left to lower right diagonal.

In Figure 25 and Figure 26, the individual image classification results are summarized in confusion matrix tables with model M-DA-2 in the presence of data augmentation techniques with Gaussian noises, and model M-DA-3 with additional and extreme data augmentation techniques (for details about data augmentation, see Table 2). Image classification error rates for models M-DA-2 and M-DA-1 are similar. Model M-DA-3 fails to provide helpful information for class predication, indicating additional and extreme data augmentation techniques are not supposed to perform better. These results are consistent with the summarized model performance results in Table 4.

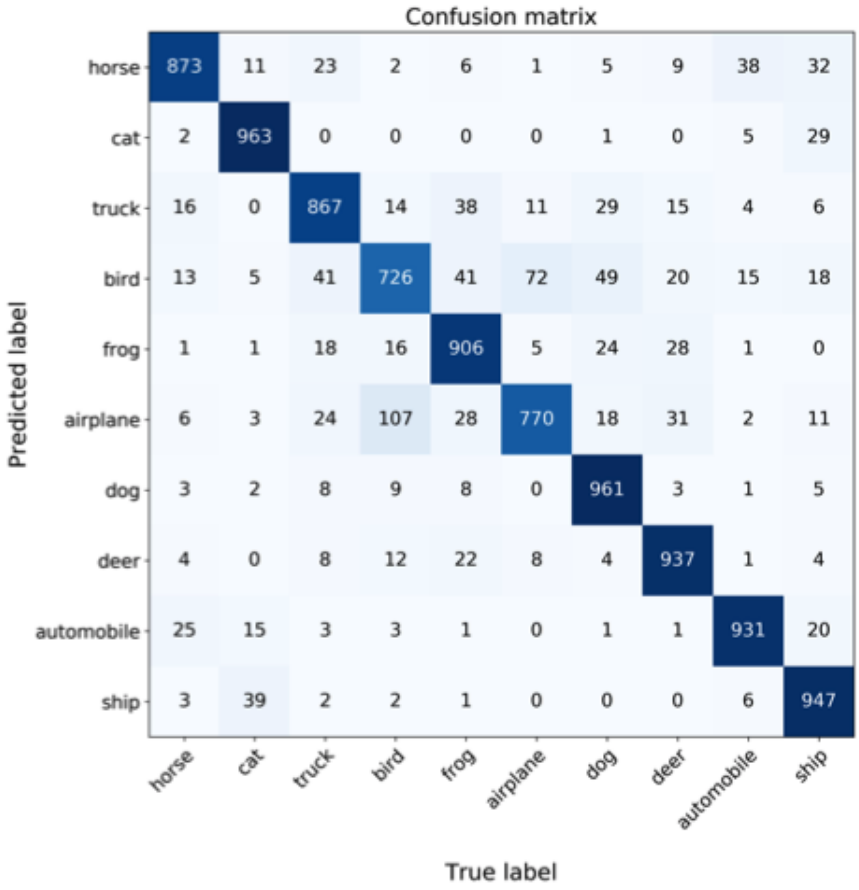

Fig. 25. Confusion matrix table for 10-class image classification using the M-DA-2 model with general data augmentation techniques and extra 10% Gaussian noises (for details about data augmentation, see Table 2). Columns represent true classes, and rows represent the classifier's predictions. The square matrix puts all correction classifications along the upper-left to lower right diagonal.

5.8 VGG16 Model for CIFAR-10 Image Classification

The macro-architecture of model VGG16 is shown in Figure 27. For this model pattern, there are two stacks of two convolutional layers and three stacks of three convolutional layers followed by one max-pooling layer after every stack. In total, there are 16 weighted layers in the VGG16 model. This model requires large weight space due to the depth and the number of fully connected layers in the end. The filter numbers slide from 64 to 128, 256 and 512 separately in the stacked convolutional layers. The flatten layer is used to adjust the feature maps to the dense layer with the output of ten classes.

The results of train and test accuracy and loss are shown for models VGG16-0 (Figure 28), -1 (Figure 29) and -2 (Figure 30) in the presence of optimizer *Adamax* and activation argument *elu*. The details about data augmentation techniques are listed in Table 5, and the results for model accuracy and loss are shown in Table 6. Model VGG16-0 is used as control and not applied with data augmentation. The difference between models VGG16-1 and VGG16-2 is that 10% Gaussian noises are added to model VGG16-2. Model overfitting exists in the VGG16-0 model (Figure 28), but not in other two models. Model VGG16-1(Figure 29) performs the best compared with other 2 VGG16 models, with the highest test accuracy at 92%.

Table 5. Summary of ImageDataGenerator parameters in three VGG16 models

Model	ImageDataGenerator Parameters
1. VGG16-0:	No data augmentation
2. VGG16-1:	rotation 15, width shift 0.1, height shift 0.1, horizontal flip
3. VGG16-2:	rotation 15, width shift 0.1, height shift 0.1, horizontal flip, zca epsilon: 1e-6, fill with nearest, Gaussian noise 0.1

The confusion matrix tables for 10-class image classification are shown in Figure 31 and Figure 32 for models VGG16-0 and -1. The details about data augmentation are listed in Table 5. VGG16-1 (Figure 32) is preprocessed with general data augmentation. The model VGG16-1 performs better than VGG16-0 from the test accuracy results as shown in Table 6. In model VGG16-1, the overall image classification error rate is dropped to 8%, and eight out of ten single class misclassification rates are less than 7.5%. Data augmentation significantly improves model performance.

The VGG16 model's classification report depicts the other important metrics that can be derived from the confusion matrix, including precision, recall, and F1 score. Precision is the ratio of correct positive predictions to the total predicted positive observations, recall is the ratio of correct positive predictions to the all observations in actual class, and F1 score is the weighted average of precision and recall. As shown in Table 7 for model VGG16-1, the overall model accuracy and F1 score are 92% indicating the optimized performance with data augmentation.

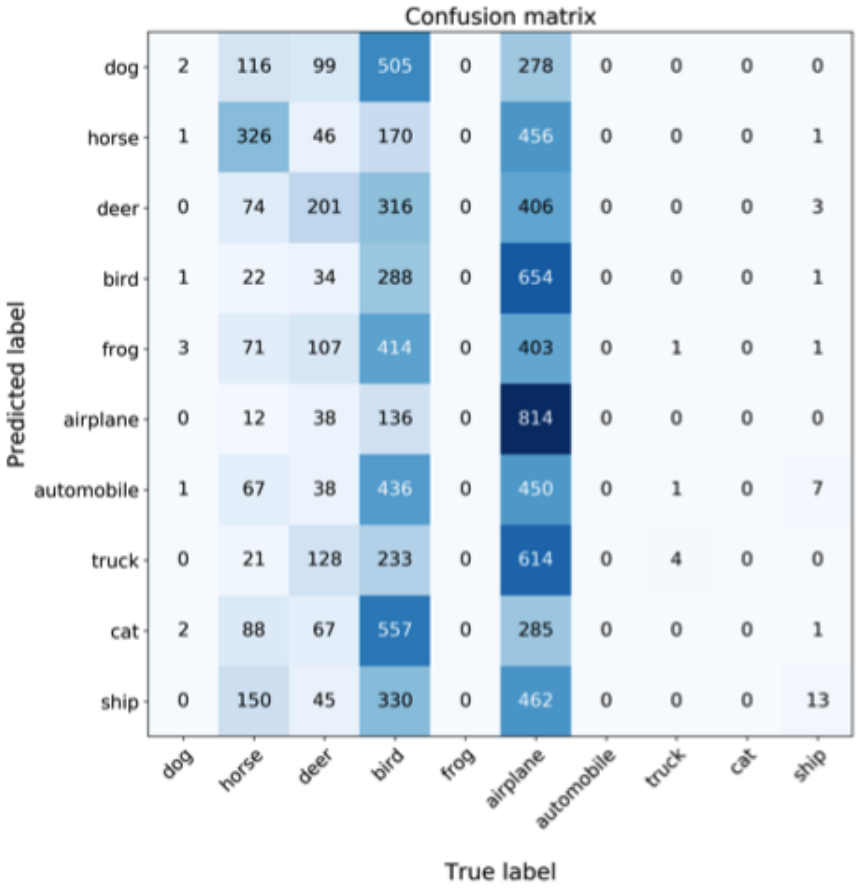

Fig. 26. Confusion matrix table for 10-class image classification with M-DA-3 model in the presence of additional and extreme data augmentation techniques (for details about data augmentation, see Table 2). Columns represent true classes, and rows represent the classifier's predictions. The square matrix puts all correction classifications along the upper-left to lower right diagonal. Using extra data augmentation techniques for the M-DA-3 model fails to provide helpful information for class predication, indicating more data augmentation techniques are not supposed to perform better.

Table 6. Summary of accuracy and loss in three VGG16 models

Model	Train Acc.	Train Loss	Test Acc.	Test Loss
1. VGG16-0	0.9772	0.3990	0.8854	0.8206
2. VGG16-1	0.9649	0.2976	0.9215	0.4794
3. VGG16-2	0.9650	0.3009	0.9217	0.5004

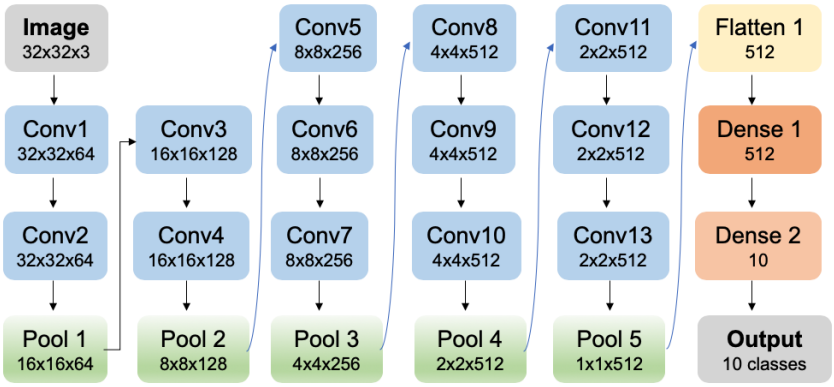

Fig. 27. Graphic representation of VGG16 model architecture used for CIFAR-10 image classification. For model pattern, there are two stacks of two convolutional layers and three stacks of three convolutional layers followed by one pooling layer after every stack [8]. The filter numbers slide from 64 to 128, 256 and 512 separately in the stacked convolutional layers. The flatten layer is used to adjust the tensors to the dense layer with the probability output for ten classes.

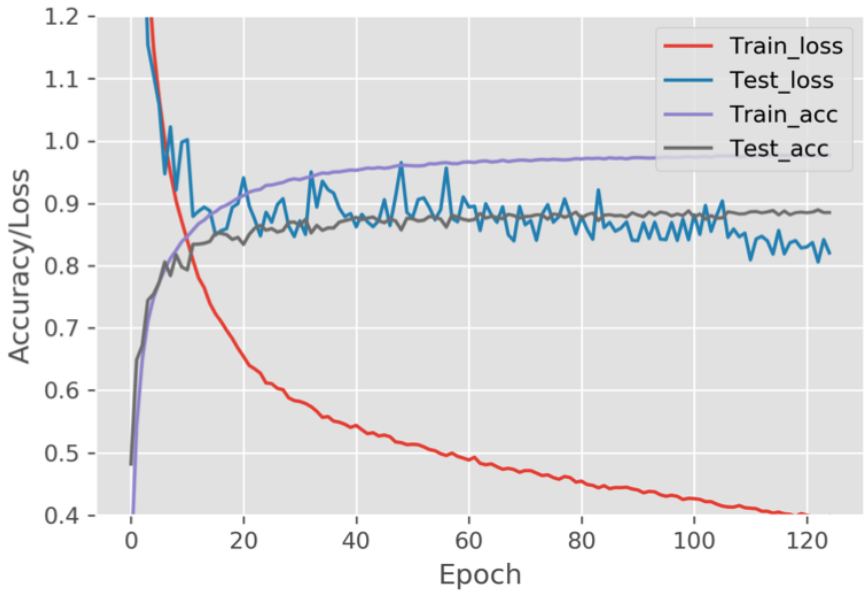

Fig. 28. Traces of train and test accuracy and loss in VGG16-0 model in the presence of optimizer *Adamax* and activation argument *elu*. The VGG16-0 model overfits when data augmentation is not applied. The accuracy and loss results are shown in Table 6.

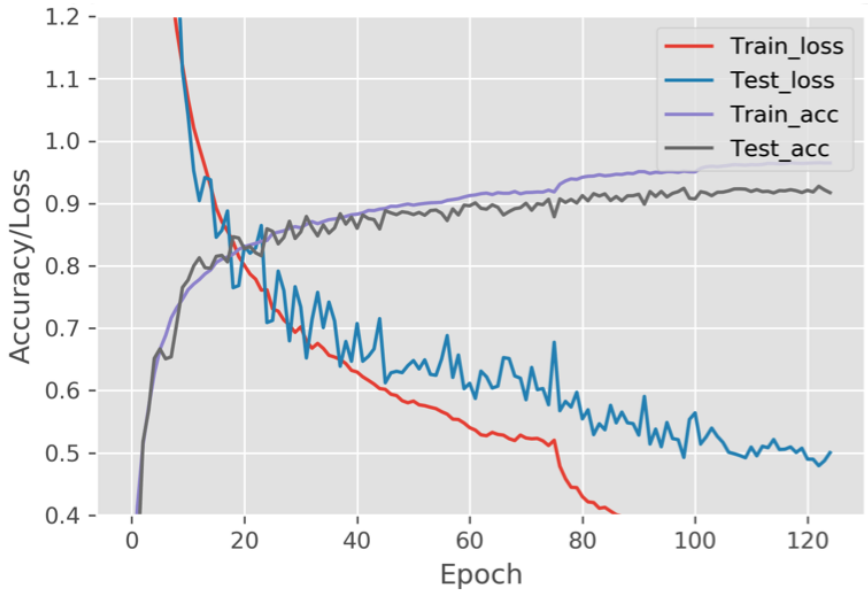

Fig. 29. Traces of train and test accuracy and loss in model VGG16-1 with optimizer *Adamax* and activation argument *elu*. The data augmentation techniques for VGG16-1 model are shown in Table 5, and the model performance results are in Table 6.

Table 7. Classification report for model VGG16-1 with data augmentation

Class	Precision	Recall	F1score	Support
Horse	0.95	0.92	0.94	1000
Truck	0.95	0.97	0.96	1000
Dog	0.88	0.93	0.90	1000
Frog	0.86	0.81	0.84	1000
Airplane	0.91	0.94	0.92	1000
Ship	0.92	0.83	0.87	1000
Cat	0.89	0.97	0.93	1000
Deer	0.96	0.94	0.95	1000
Automobile	0.95	0.97	0.96	1000
Bird	0.95	0.95	0.95	1000
Micro avg	0.92	0.92	0.92	10000
Macro avg	0.92	0.92	0.92	10000
Weighted avg	0.92	0.92	0.92	10000

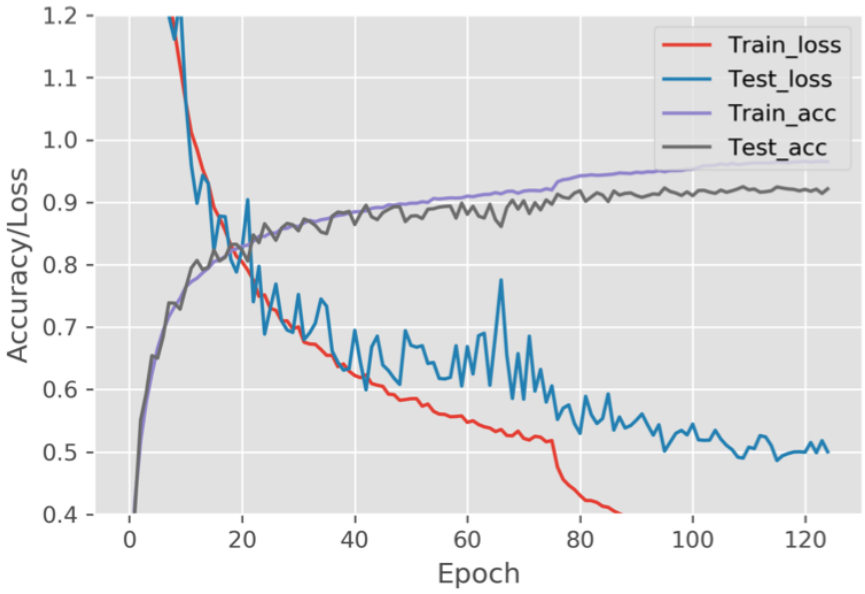

Fig. 30. Traces of train and test accuracy and loss in model VGG16-2 with optimizer *Adamax* and activation argument *elu*. The VGG16-2 model is different from the model VGG16-1 in that extra 10% Gaussian noises are added. The data augmentation techniques are shown in Table 5, and the model performance results are in Table 6.

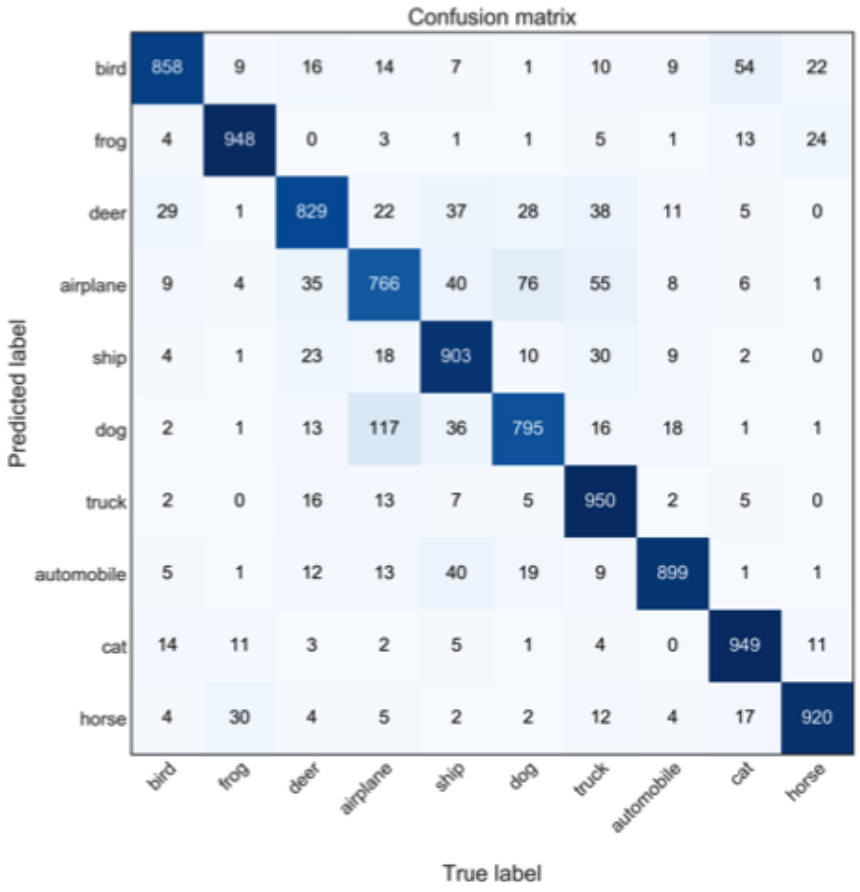

Fig. 31. Confusion matrix table for 10-class image classification with model VGG16-0 in the absence of data augmentation. Columns represent true classes, and rows represent the classifier's predictions. The square matrix puts all correct classifications along the upper-left to lower right diagonal.

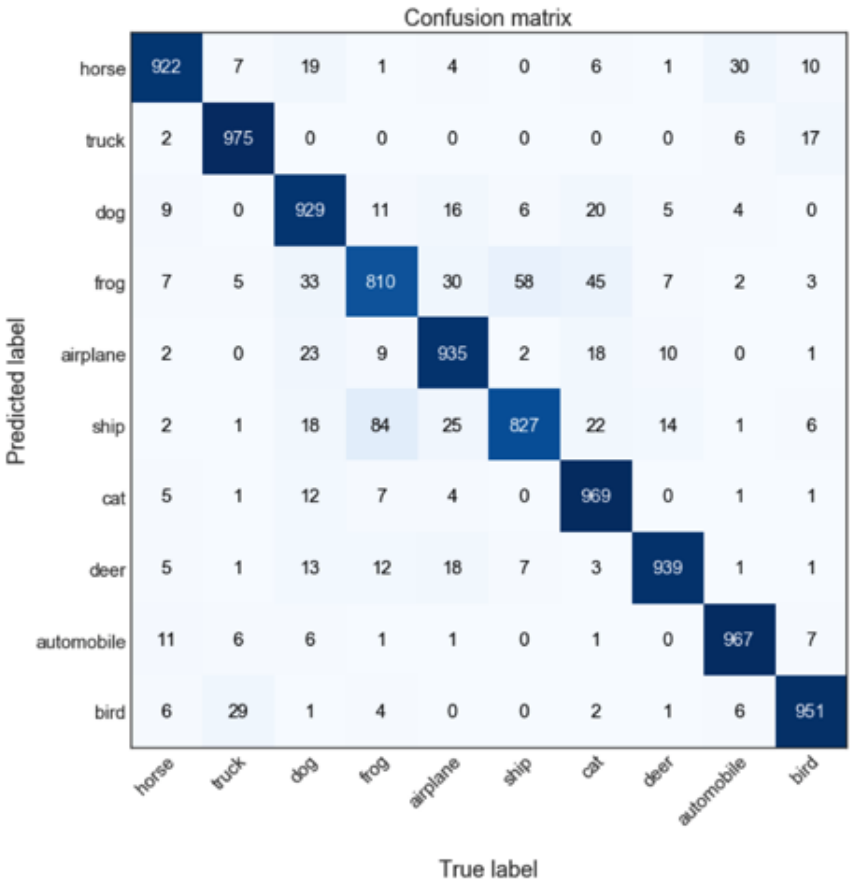

Fig. 32. Confusion matrix table for 10-class image classification with VGG16-1 model in the presence of data augmentation. The details about data augmentation techniques are shown in Table 5. Columns represent true classes, and rows represent the classifier's predictions. The square matrix puts all correct classifications along the upper-left to lower right diagonal.

5.9 Visualization of Image Classification Using t-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) was developed by van der Maaten and Hinton in 2008 [17], and used to visualize high dimensional data into low dimensional space. t-SNE is based on probability distribution for non-linear mapping. t-SNE visualization measures the effective number of neighbors with minimized relative entropy. For visualizing neural network model outputs, features should be extracted first from the binary outputs for all classes. The complex feature matrices are then transformed into two dimensional points by using t-SNE algorithm, and finally displayed in effective clusters to match the corresponding labels.

In Figure 33, t-SNE is used to visualize four representative model outputs. Model M0 is the basic CNN model with only two convolutional layers, model M-DA-0 is the standard model with six convolutional layers, model M-DA-1 is to add general data augmentation techniques to model M-DA-0, and VGG16-1 is the advanced VGG16 model with data augmentation. From the model output visualizations by using t-SNE (Figure 33), the unclassified or misclassified regions shrink from model M0 to model M-DA-0, and even more significantly to model M-DA-1 and model VGG16-1 after applying image data augmentation techniques. The model output visualizations in these four models are consistent with the results and plots of accuracy and loss, and the derived information from confusion matrices for models M0, M-DA-0, M-DA-1, and VGG16-1.

6 Result Summary and Analysis

The major model development results are summarized in Figure 34. The aims are to compare these eight models in three different CNN model architectures and use the completed model for image classification implementation.

6.1 Data Augmentation

Image preprocessing is to transform the raw data before being trained with neural networks. This step is important to speed up training process and improve classification performance. There are several technologies that can be applied for preprocessing, such as: min-max normalization and data augmentation. In this paper, the images are preprocessed in the absence or presence of data augmentation techniques (Table 2 and Table 5) by using Keras ImageDataGenerators.

6.2 Convolutional Layer

In the sequential CNN models (Figure 4), the convolutional layers are the core building blocks which systemically apply learned filters to input images and extract the important image features. VGG16 models are built with 13 convolutional layers (Figure 27), more than standard M-DA models (six layers, Figure 10) and basic M0 model (two layers, Figure 7).

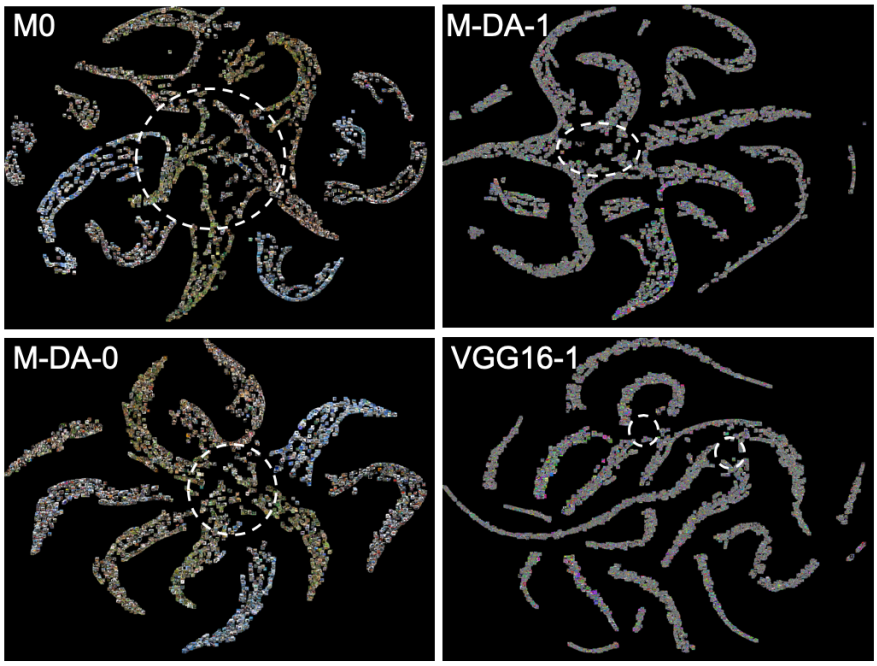

Fig. 33. Visualization of model outputs from four constructed models by using t-SNE. t-SNE dimensionality reduction technique is used to transform the high dimensional features extracted from the binary output of all ten classes to two-dimensional map points. Similar images are normally clustered close to each other. Image classification is significantly improved with data augmentation as shown from the unclassified images concentrated in the central region of each model. (A) t-SNE visualization of model output from model M0, unclassified samples are marked with large dotted circle, (B) t-SNE visualization of model output for model M-DA-0. The unclassified samples are marked with smaller dotted circle. (C) t-SNE visualization of model output for model M-DA-1 after applying data augmentation. The dotted circle is to show even less unclassified images after image data augmentation. (D) t-SNE visualization of model output for model VGG16-1 after applying data augmentation.

6.3 Optimizer

Optimizer is the hyper-parameter used for calculating loss function to update weight and bias with back-propagation. There are seven optimizers in Keras neural network library (*SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax*, and *Nadam*). *Adamax* is a variant of *Adam* in its infinity form. It is the selected optimizer from model M-DA-0 training (Table 1) and be used in VGG16 models.

6.4 Activation Argument

The activation argument is typically applied after each convolutional layer. In the Keras neural network library, there are ten available activation types. *elu* and *selu* are the two variants of the nonlinear rectified linear unit *relu*. The activation argument *elu* performs the best among ten activation arguments due to its simplicity and ability to enable fast training process. The train accuracy for activation argument *elu* is 90% and the test accuracy is 89% (Table 3).

6.5 Model Overfitting

Model overfitting is one of the most common problems when using neural networks. When it happens, the model train accuracy is higher than the test accuracy, and the test loss is higher than the train loss. Comparatively for the optimized model, the train and test curves are very close to each other in accuracy and loss plots. These trends can be seen better during more iterations or epochs. VGG16 models have less overfitting concerns than basic and standard CNN models (compare basic M0 model in Figure 8, standard M-DA-0 model in Figure 11, M-DA-1 model in Figure 20, M-DA-2 model in Figure 21, VGG16-0 model in Figure 28, VGG16-1 model in Figure 29 and VGG16-2 model in Figure 30). The model overfitting reductions are confirmed with the improved test accuracy and loss results for VGG16 models.

6.6 Train and Test Accuracy

Accuracy is defined as the ratio of correctly predicted observation to the total observations. The train accuracy for the best performance VGG16-1 model is 96% and test accuracy is 92%. There are no accuracy paradox problems since the test class sizes are equally 1,000.

6.7 Train and Test Loss

Loss is defined as the categorical cross-entropy to predict class probabilities (e.g. the difference between the predicted value and the true value). Model performance can be improved with minimizing the loss function. For the best performance VGG16-1 model, the test loss is only 0.48 which is less than 0.5.

6.8 Confusion Matrix

The confusion matrix is a table to report the number of false positives, false negatives, true positives, and true negatives. This allows more detailed analysis than mere proportion of correct classifications (accuracy), especially when accuracy paradox problems exist. The confusion matrix of VGG16-1 model in the presence of data augmentation shows better image classification than other models (Figure 32, Table 5, Table 6) .

6.9 Classification F1 Score

The F1 score is helpful in the setting of multi-class prediction, particularly when the class sizes are not balanced. Applying data augmentation techniques to target specific classes is an alternative way to improve model performance. The confusion matrix and the F1 score in classification report show model VGG16-1 in the presence of data augmentation (Figure 32) has better performance than model VGG16-0 (Figure 31).

6.10 t-SNE Visualization

Dimensionality reduction techniques such as t-SNE [17] can help map the last hidden layer outputs in two dimensions. t-SNE provides another useful technique to provide supplementary information for the classified spectra. The t-SNE visualization is used to compare four model outputs in clusters (Figure 33). The well-defined clusters for the VGG16-1 model output are consistent with the results from accuracy, loss and confusion matrix.

7 Ethics

Image classification using neural networks has been applied in many fields, such as artificial intelligence, automotive industries and medical diagnostics. In this paper, these techniques are specifically used to classify physical objects (e.g. frog and automobile) with extracted image features.

Neural networks have the potentials to learn things that the data scientists unlikely want the model to train initially. For example, the model can unintentionally produce a result that discriminates against certain sexes, races, or abilities. This could happen because of some inherent bias collected in the dataset. We need to handle these data cautiously to ensure that the data is unbiased and these neural networks or deep learning algorithms to be trained cautiously to produce a reliable and unbiased output.

The various components in a machine learning model are certainly complex, and deep learning was once regarded as a black box [19]. Most shallow machine learning models can be explained, such as analyzing model feature importance and correlations. However, deep learning models are much more complicated and difficult to explain. Because of higher capabilities and greater range of impacts,

	Basic M0	Standard M-DA Model	Advanced VGG16 Model
1. Data Augmentation	No DA	M-DA-0: No DA M-DA-1: DA M-DA-2: DA+GN M-DA-3: DA+RSZ	VGG16-0: No DA VGG16-1: DA VGG16-2: DA+GN
2. Convolutional Layers	2	6	13
3. Optimizer	*SGD*	select '*Adamax*' from 7 optimizers in M-DA-0	*Adamax*
4. Activation Argument	*relu*	select '*elu*' from 10 arguments in M-DA-1	*elu*
5. Model Overfitting	overfitting	reduced (M-DA-1, -2)	reduced (VGG16-1, -2)
6. Train and Test Accuracy		0.90 and 0.89 (M-DA-1) 0.90 and 0.89 (M-DA-2)	0.96 and 0.92 (VGG16-1) 0.97 and 0.92 (VGG16-2)
7. Train and Test Loss		0.37 and 0.45 (M-DA-1) 0.34 and 0.42 (M-DA-2)	0.30 and 0.48 (VGG16-1) 0.30 and 0.50 (VGG16-2)
8. Confusion Matrix		[718, 928] for M-DA-0 [733, 956] for M-DA-1 [726, 963] for M-DA-2	[829, 949] for VGG16-0 [810, 975] for VGG16-1
9. Classification F1 Score			[0.84, 0.96] for single class, 0.92 for average F1 score
10. t-SNE Visualization	unclassfied	improved for M-DA-0 optimized for M-DA-1	well clustered (VGG16-1)

Fig. 34. Result summary for the eight models in three different CNN architectures. The images are preprocessed in the absence or presence of data augmentation techniques (Table 2 and Table 5). VGG16 model has 13 convolutional layers, more than standard M-DA model (six layers) and basic M0 model (two layers). The optimizer and activation argument are selected from available candidates in Keras neural network library. *Adamax* and *elu* are the selected optimizer and activation argument used in the VGG16 model. Model VGG16 has less overfitting than basic M0 and standard M-DA model. The overfitting reductions are confirmed with the improved test accuracy and loss results for model VGG16. The confusion matrix and F1 score in classification report show the VGG16-1 model with data augmentation has better model performance than the VGG16-0 model. The t-SNE visualization is used to compare four model outputs in clusters. The VGG16-1 model output displays well defined clusters which are consistent with the results from accuracy, loss and confusion matrix. DA: data augmentation; GN: Gaussian noises; RSZ: resize, shear and zoom.

the decision-making process with deep learning algorithms should be explainable in terms people can understand. Explainability is the key for the end users especially in healthcare, criminal justice and other domains.

Model overfitting is also one of the concerns in applying neural network models. This happens when a model is too complex or too many features with small training dataset. For instance, such a model in disease diagnostic practice can harm trust in the health-care provider-patient relationship. We need to be cautious for balancing model efficiency and ethics in applications.

The different policies in applying neural networks worldwide is still a reality today. The potential concerns for their deployments without conferring interpretability are raised in Europe and Japan. These approaches should be revealed before practical applications. Comparatively in the United States, Amazon launched Comprehend Medical service (ACM) in the November of 2018. This ACM service uses pre-trained neural network models to understand the medical information and identify meaningful relationships in unstructured text.

In this big data era, global ethical guidance should be emphasized to protect user data and preserve the user's power over access and uses. We should recognize and adhere to applicable national and international regulations when working with machine learning algorithms. For examples, General Data Protection Regulation (GDPR)[15] which is based on European Union (EU) law and data protection and privacy for all individual citizens of the EU and European Economic Zone (EEA). In addition, individual institutions like Canadian research groups have their institutional, state-level and other related regulations to data resources and machine learning studies.

Specifically, the image dataset CIFAR-10 used in this study is from freely available public resource. All of the images do not include human subjects. Therefore, no informed consents should be collected for this study. All the machine learning tools and algorithms are open sources which are allowed to distribute or modify. All online technique notes, and publications are properly cited in footnotes and references. The results of modeling are for machine learning algorithm research only and no end user deployments at this stage.

During this study, we strictly adhere to the Association for Computing Machinery (ACM) Code of Ethics for privacy, security and intellectual property, the Professional Conduct (the Codes) [20] to ensure both privacy and security of individuals, and other codes of conducts related to good ethical practices. We also follow the general ethical principles to make no harm, be honest and trustworthy, respect the work required to produce new ideas, inventions and creative works. We keep on taking professional responsibilities to achieve high quality in professional work, maintaining high standards of professional competence, knowing and respecting present rules pertaining to professional work.

[15] GDPR.
https://eugdpr.org/the-regulation/

8 Conclusions

In machine learning, image classification is used to identify the input images to their mutually exclusive categories using neural networks. CNN model is composed of feature extraction portion in multiple hidden layers, and classification portion in the output layer. CNN can achieve higher model accuracy, but model overfitting and low performance are common problems in image classification.

In this paper, we reduce model overfitting and improve image classification to build a reliable model with three strategies: (1) refining model architectures by adjusting filter sizes and adding more convolutional, maxpooling and dropout layers, (2) optimizing hyper-parameter selection (e.g. optimizer *Adamax* and activation function *elu*), and (3) implementing data augmentation with common techniques (e.g. rotation, flip, translation, Gaussian noises) to generate new images for increasing the volume of the training dataset [7]. This is extremely important for working with large image dataset and large-scale neural networks.

Specifically, VGG16 model has more convolutional and pooling layers in a complex architecture than other models. The VGG16 model with data augmentation (i.e. the VGG16-1 model) can improve the train accuracy to 96% and the test accuracy to 92%. The superior model performance is further supported by confusion matrix and the model output as visualized in clusters by using t-SNE algorithm.

Similar to refining neural network model architectures and tuning hyper-parameters, data augmentation does not always work well with using more and extreme techniques combined together to preprocess image data. The practical approach is to start with implementing the most common data augmentation techniques and then make favorable adjustments. The model performance should be improved significantly after solving these problems like model overfitting.

9 Acknowledgements

The authors thank Dr. Robert Slater for supporting this study with his vast knowledge and experience in neural networks, Dr. Daniel W. Engels for fruitful comments and inspiration, and Dr. Jake Drew for helpful discussions.

References

1. Sze, V., Chen, YH., Yang, TJ., et al.: Efficient Processing of Deep Neural Networks: A Tutorial and Survey. `https://arxiv.org/pdf/1703.09039.pdf`
2. Hannun, AY., Rajpurkar, P., Haghpanahi, M., et al.: Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. Nature Medicine. **25** (2019) 65–69
3. Krizhevsky, A., Sutskever, I., Hinton GE.: ImageNet Classification with Deep Convolutional Neural Networks. `https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf`

4. Wu, H., Gu, X.: Towards Dropout Training for Convolutional Neural Networks. https://arxiv.org/pdf/1512.00242.pdf
5. Zhang, C., Vinyals, O., Munos, R., et al.: A Study on Overfitting in Deep Reinforcement Learning. https://arxiv.org/pdf/1804.06893.pdf
6. Hernandez-Garcia, A., Konig, P.: Do deep nets really need weight decay and dropout? https://arxiv.org/pdf/1802.07042.pdf
7. Wang, J., Perez, L.: The Effectiveness of Data Augmentation in Image Classification using Deep Learning. http://cs231n.stanford.edu/reports/2017/pdfs/300.pdf
8. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. https://arxiv.org/pdf/1409.1556.pdf
9. Ramo, K. (ed.): Hands-On Java Deep Learning for Computer Vision. Packt Publishing, Birmingham UK (2019)
10. Chekanov, SV. (ed.): Numeric Computation and Statistical Data Analysis on the Java Platform. Chapter 13: 475-504 Neural Networks. Springer International Publishing Switzerland (2016)
11. Matsugu, M., Mori, K., Mitari, Y., et al.: Subject independent facial expression recognition with robust face detection using a convolutional neural network. Neural Networks. **16** (2003) 555–559. https://doi.org/10.1016/S0893-6080(03)00115-1
12. Gu, J., Wang, Z., Kuen, J., et al.: Recent Advances in Convolutional Neural Networks. https://arxiv.org/pdf/1512.07108.pdf
13. Atienza, R. (ed.): Advanced deep learning with Keras. Packt Publishing Birmingham UK (2018)
14. Sensoy, M., Kaplan, L., Kandemir, M.: Evidential Deep Learning to Quantify Classification Uncertainty. https://arxiv.org/pdf/1806.01768.pdf
15. Krizhevsky, A.: Learning Multiple Layers of Features from Tiny Images. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf
16. Burkov, A. (ed.): The hundred-page machine learning book. Publisher: Andriy Burkov (2019)
17. van der Matten, LJP.,Hinton, GE.: Visualizing Data using t-SNE. Journal of Machine Learning Research **9** (2008) 2579–2605. http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf
18. LeCun, B., Bottou, L., Bengion, Y., et al.: Gradient-based learning applied to document recognition. Proceedings of the IEEE. **86** (1998) 2278–2324
19. He, K., Zhang, X., Ren, S., et al.: Deep Residual Learning for Image Recognition. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, (2016) 770–778. https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf
20. The ACM Code of Ethics and Professional Conduct. https://www.acm.org/binaries/content/assets/about/acm-code-of-ethics-booklet.pdf