The Project of Machine Learning

Name: Samaun Jahan Rodro

Date: 06/08/2021

Version: 1

Course Name: Machine Learning

Course Code: EE4108

Word Count: 1399

• Abstract

This project aims to execute the convolutional neural network for allocating pictures in the data set of the CIFAR-10. In addition, the implementation of this network consists of the improvements for the architectural performance which include Dropout, Batch-Normalization, and Image augmentation.

• Introduction

Neural networks are programmable modes that help solve complex problems and provide the best possible result. As it is well known that deep learning is one step ahead of machine learning and helps train neural networks to get solutions to unanswered questions or improve solutions. In this article, the project is developed a deep learning model using the CFAR-10 dataset where the dataset is frequently used in deep learning to test images classification algorithms. It has 60000 color images divided into ten categories. The images are 32 x 32 pixels in measurement containing 50000 plus 10000 training and test images respectively.

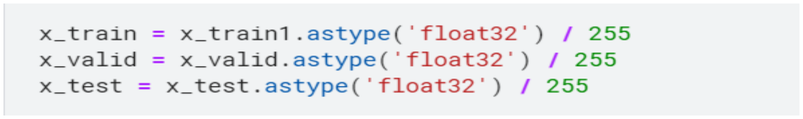
The implementation of this project based on the AlexNet neural network architecture which is well known for winning the ImageNet contest. AlexNet is an 8-layer convolutional neural network that can load a pretrained version of the network that has been trained on over a million photos from the ImageNet database and can categorize images into 1000 object categories such as keyboard, mouse, pencil, and a variety of animals [1]. Furthermore, Keras is a high-level deep learning API utilised in this project to make the building of neural networks simple, and it runs on top of the TensorFlow machine learning platform [2]. In specifically, the goal of this project is to first develop a convolutional neural network based on AlexNet, and then to create variants by improving the core architecture to avoid overfitting and increase performance [3]. These improvements consisting of Dropout, BatchNorm and Image Augmentation.

In this article, the main work is consisting of 5 steps to improvise the neural network where the total model accuracy will increase by moving from step by step (Step 1: "Model accuracy is 0.702600002") where total model loss will decrease by moving from step by step (Step 1: "Model accuracy is 1.928587914"). Comparison between accuracy to accuracy and loss to loss for different steps will be described briefly in the forthcoming sections.

• Methods

Dataset: This section provides a high-level description of a typical Convolutional Neural Network (CNN) workflow for justifying images in CIFAR-10 dataset machine learning process. Pre-processing data is the initial stage in every Machine Learning, Deep Learning, or Data Science project. Convolutional Neural Networks will be used to create the model.

Pre-Processing: Images are made up of pixel value matrices. Normally, pixels should have a value between 0 and 255. These numbers must be normalized to a range of 0 to 1.



```
x_train = x_train1.astype('float32') / 255
x_valid = x_valid.astype('float32') / 255
x_test = x_test.astype('float32') / 255
```

One-hot code are a set of bits in which the only permissible value combinations are those that have a single high bit, and all the rest are low.

```
from keras.utils.np_utils import to_categorical
N = 10
y_train = to_categorical(y_train1, N)
y_valid = to_categorical(y_valid, N)
y_test = to_categorical(y_test, N)
```

The matrices will be converted into binary matrices with a width of ten. The categorical data cannot just hand over to the computer to process. As a result, the actions must take outlined above to make the data more machine-processable.

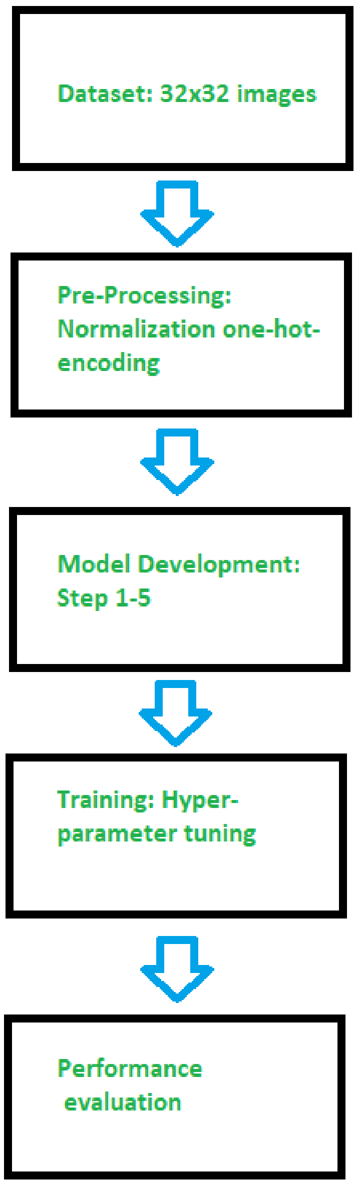
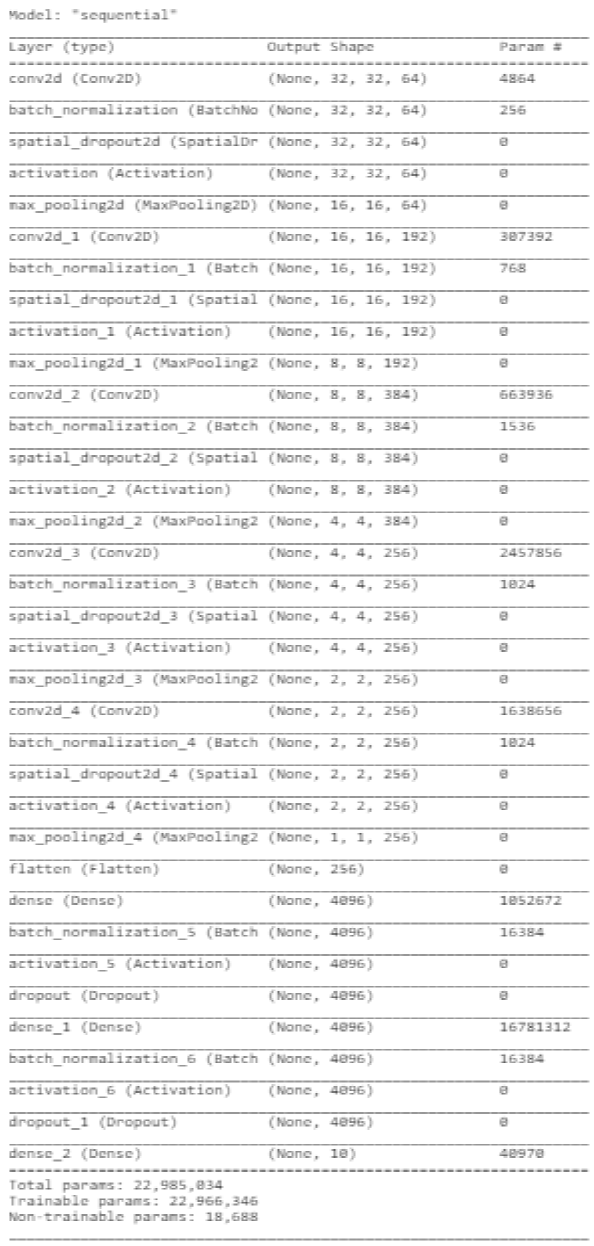

Dataset: 32x32 images
Pre-Processing: Normalization one-hot-encoding
Model Development: Step 1-5
Training: Hyper-parameter tuning
Performance evaluation

Figure 1: End-to-end machine learning process

Model Development and Hyper parameter tuning: To develop the model set, convolutional neural network is initializing by using sequential model of keras. The activation functions ReLu and softmax are employed in this model. In the feature map, ReLU eliminates any negative pixel values with 0. Softmax takes a vector of real numbers as input and converts it to a probability distribution made up of real number probabilities proportional to the input numbers' exponentials.

This is a representation of our CNN model



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 64)        4864
_________________________________________________________________
batch_normalization (BatchNo (None, 32, 32, 64)        256
_________________________________________________________________
spatial_dropout2d (SpatialDr (None, 32, 32, 64)        0
_________________________________________________________________
activation (Activation)      (None, 32, 32, 64)        0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 16, 16, 192)       307392
_________________________________________________________________
batch_normalization_1 (Batch (None, 16, 16, 192)       768
_________________________________________________________________
spatial_dropout2d_1 (Spatial (None, 16, 16, 192)       0
_________________________________________________________________
activation_1 (Activation)    (None, 16, 16, 192)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 192)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 8, 8, 384)         663936
_________________________________________________________________
batch_normalization_2 (Batch (None, 8, 8, 384)         1536
_________________________________________________________________
spatial_dropout2d_2 (Spatial (None, 8, 8, 384)         0
_________________________________________________________________
activation_2 (Activation)    (None, 8, 8, 384)         0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 384)         0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 4, 4, 256)         2457856
_________________________________________________________________
batch_normalization_3 (Batch (None, 4, 4, 256)         1024
_________________________________________________________________
spatial_dropout2d_3 (Spatial (None, 4, 4, 256)         0
_________________________________________________________________
activation_3 (Activation)    (None, 4, 4, 256)         0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 2, 2, 256)         0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 2, 2, 256)         1638656
_________________________________________________________________
batch_normalization_4 (Batch (None, 2, 2, 256)         1024
_________________________________________________________________
spatial_dropout2d_4 (Spatial (None, 2, 2, 256)         0
_________________________________________________________________
activation_4 (Activation)    (None, 2, 2, 256)         0
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 1, 1, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 256)               0
_________________________________________________________________
dense (Dense)                (None, 4096)              1052672
_________________________________________________________________
batch_normalization_5 (Batch (None, 4096)              16384
_________________________________________________________________
activation_5 (Activation)    (None, 4096)              0
_________________________________________________________________
dropout (Dropout)            (None, 4096)              0
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              16781312
_________________________________________________________________
batch_normalization_6 (Batch (None, 4096)              16384
_________________________________________________________________
activation_6 (Activation)    (None, 4096)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                40970
=================================================================
Total params: 22,985,034
Trainable params: 22,966,346
Non-trainable params: 18,688
_________________________________________________________________
```

Figure 2: CNN model for Task 3

This project is consisting of 5 tasks. Where first step is to implement AlexNet which is consisting of total of seven hidden layers: five convolutional layers and two fully connected layers. In this step most potential steps are performed such as 5x5@64 convolution with stride 1, ReLU activation, 2x2 max pooling with stride 2, 5x5 kernel size, 3x32x32 CHW format input image, and 64x16x16 dimensional feature. In task two, first step has been modified by adding add dropout layer to minimize the over-fitting. Spatial dropout has been placed in all 7 hidden layers where for convolutional and fully connected layers probability is 0.2 and 0.5 respectively. To increase the network's performance in task3, BatchNorm layers added to all layers where convolution layers used BatchNorm2D and BatchNorm1D for fully connected layers. In task 4, maxpooling layers removed and replaced it with fully connected layers. To accomplish the same scaling, the level of the stride on the convolutional layers was increased from 1 to 2 at the time of eliminating the max-pooling layers. In the last task, task 4 structure used and to boost generalization performance, applied additional image augmentation during training.

Performance evaluation: By performing above mentioned tasks, the evaluation of graphs, training dataset accuracy, validation dataset accuracy, and testing dataset accuracy measured where the comparison between the training and validation accuracy curves aided in the investigation of system overfitting.

- **Results**

Task 1:

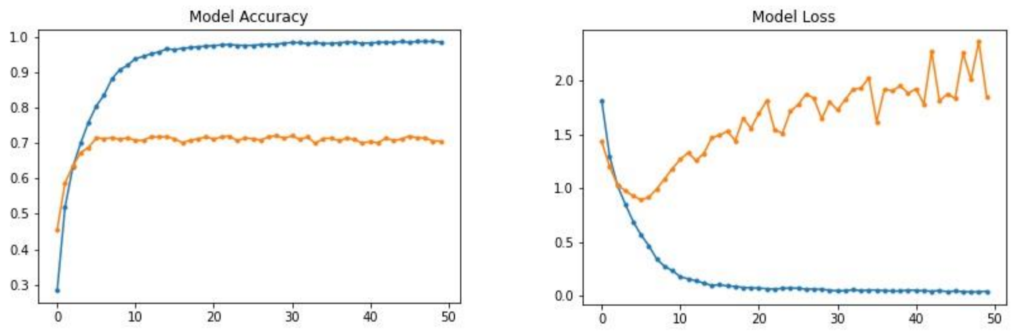


Figure 3. Model accuracy and loss of first task

Task	Train. loss	Train. Accu.	Valid. Loss	Valid. Accu.	Test loss	Test Accu.
1.	0.0354	0.9896	1.8559	0.7065	1.928587914	0.702600002

Table 1. Loss and Accuracy statistics of first task.

So, according to the model accuracy graph it can be clearly observed that, the training accuracy to 99% while validation saturates at around 70%, there is overfitting.

Task 2:

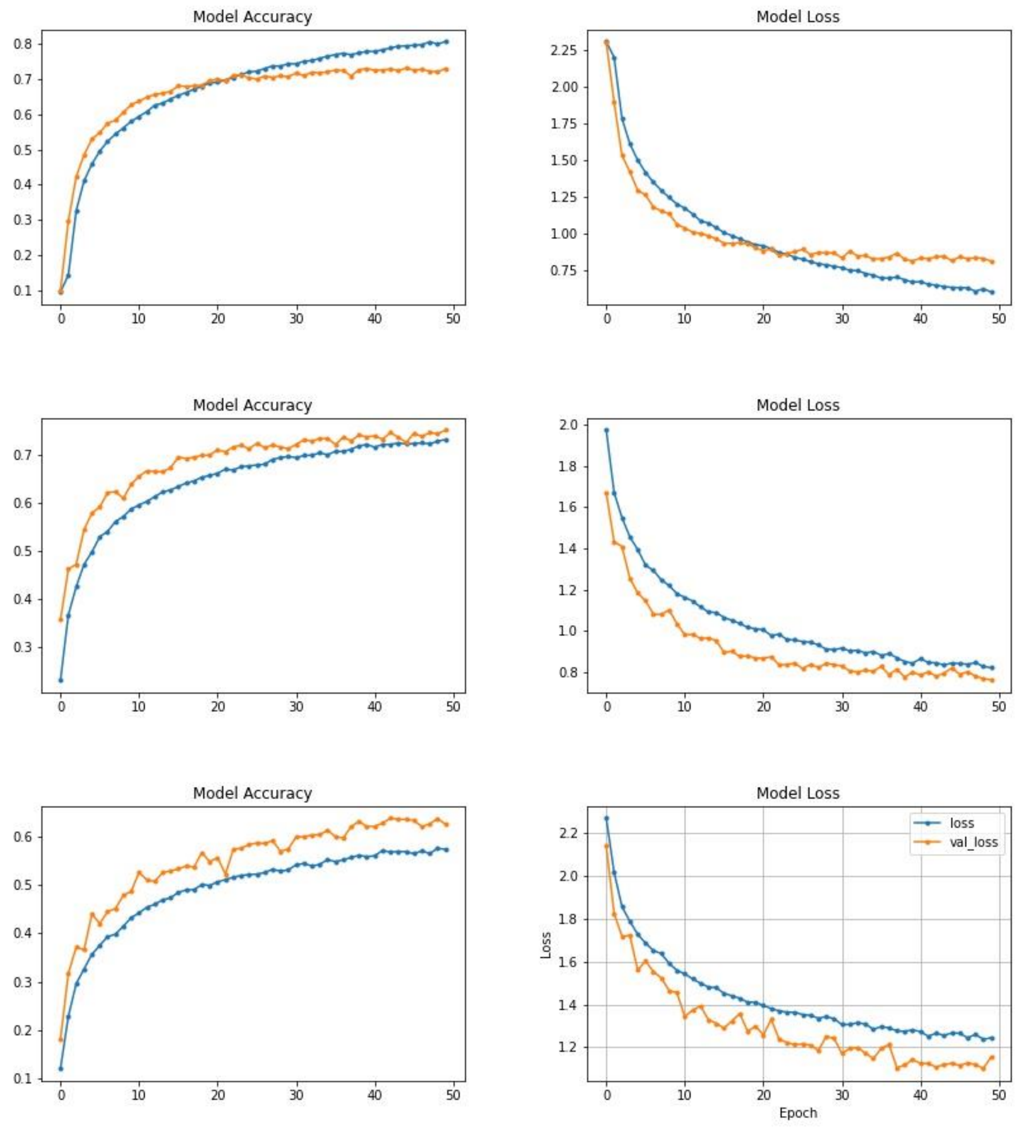


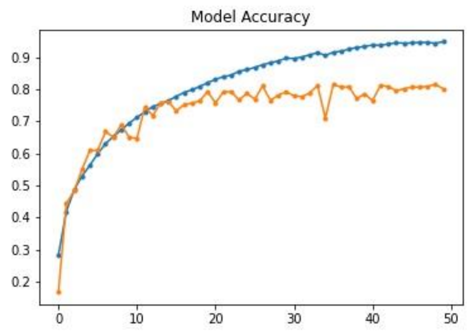
Figure 4. Model accuracy and loss of second task for 0.2, 0.3 & 0.4 respectively

Task	Drop out	Train. loss	Train. Accu.	Valid. Loss	Valid. Accu.	Test loss	Test Accu.
2.	0.2	0.6013	0.8035	0.8109	0.7291	0.802514434	0.733900011
	0.3	0.8202	0.7326	0.7601	0.7508	0.769365847	0.748600006
	0.4	1.2387	0.5758	1.1545	0.6257	1.150945663	0.62620002

Table 2. Loss and Accuracy statistics of second task.

Because we are deleting 30% of the initial neurons and the remaining 70% adjusts and the weights of the neurons rescale to fit the space, the accuracy has improved by around 4% relative to the earlier scenario, reducing overfitting.

Task 3:



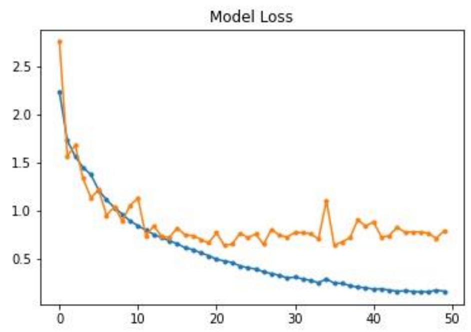


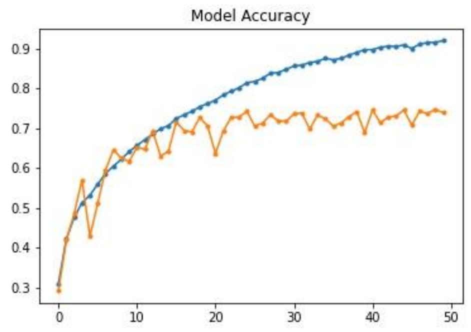
Figure 5. Model accuracy and loss of third task

Task	Train. loss	Train. Accu.	Valid. Loss	Valid. Accu.	Test loss	Test Accu.
3.	0.1629	0.9502	0.7918	0.8014	0.819568455	0.797699988

Table 3. Loss and Accuracy statistics of third task.

Due to batchNormalization, the accuracy has improved. This method solves the problem of fading away gradients by allowing the model to learn the best scale and mean for each of the layer's inputs.

Task 4:



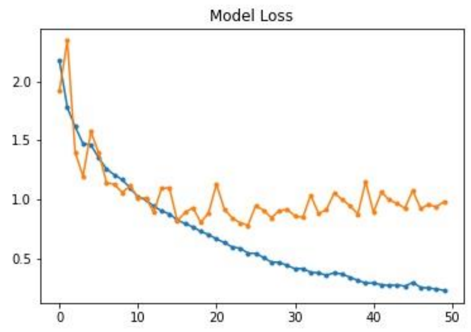


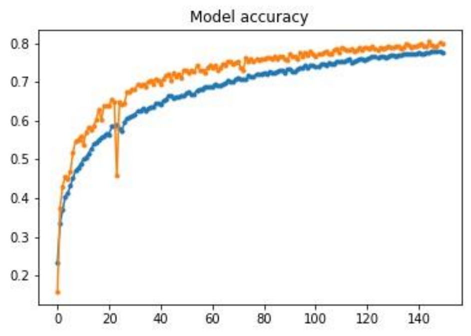
Figure 6. Model accuracy and loss of fourth task

Task	Train. loss	Train. Accu.	Valid. Loss	Valid. Accu.	Test loss	Test Accu.
4.	0.2176	0.9241	0.9824	0.7395	1.014063239	0.734499991

Table 4. Loss and Accuracy statistics of fourth task.

The elimination of Max-pooling reduced the accuracy; this function served as a filter, preserving the major features of the images, and so enhancing the accuracy.

Task 5:



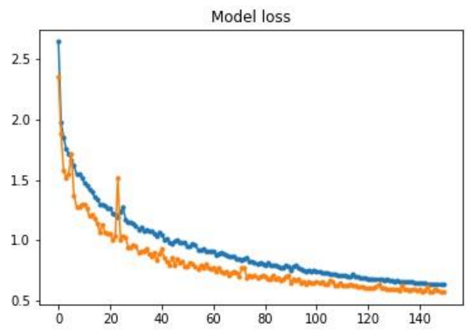


Figure 7. Model accuracy and loss of fifth task

Task	Train. loss	Train. Accu.	Valid. Loss	Valid. Accu.	Test loss	Test Accu.
5.	0.6397	0.7764	0.5746	0.8008	0.599996567	0.791100025

Table 5. Loss and Accuracy statistics of fifth task.

By altering the images and creating extra samples to use, augmentation expands the training set and improves the model's accuracy.

• Conclusion

To classify the CIFAR-10 dataset, a convolutional neural network was used. First, the AlexNet structure was developed, and then improvements like Dropout, BatchNorm, and image augmentation were introduced to the neural network. Finally, the network's performance was verified and improved. The CNN model was built with numerous layers and trained. Finally, the classification model was put to the test by presenting some random photos. With an accuracy of 79.11 percent, the image classifier predicted the outcomes. The neural network performed well after adding image augmentation, with test accuracy increasing to about 0.79 and graphs becoming less underfitting. However, the performance might be enhanced further by increasing the amount of training data.

• Reference

[1]. "Receiver operating characteristic - MATLAB roc", *Mathworks.com*, 2021. [Online]. Available: https://www.mathworks.com/help/deeplearning/ref/roc.html. [Accessed: 13- Aug-2021].

[2]. "Receiver operating characteristic - MATLAB roc", *Mathworks.com*, 2021. [Online]. Available: https://www.mathworks.com/help/deeplearning/ref/roc.html. [Accessed: 13- Aug-2021].

[3]. A. Krizhevsky, I. Sutskever and G. Hinton, "ImageNet classification with deep convolutional neural networks", *Communications of the ACM*, vol. 60, no. 6, pp. 84-90, 2017. Available: 10.1145/3065386.