Analyzing CIFAR-10 dataset using Deep Learning Models

ABSTRACT

The CIFAR-10 dataset includes 60000 32x32 colour photos in 10 classes, consisting of 6000 photos per class. 50000 photos are used for training purposes whilst 10000 photos are used for test purposes. The 10 different classes include airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships and trucks. In this paper, we attempt to use simple Neural Network and different forms of Convolutional Neural Networks (CNN) to create models and classify the images in CIFAR-10.

I. INTRODUCTION

Deep learning has been widely used in image recognition these days. Artificial neural networks, which comprises of multiple layers, can be used to drive deep learning. For instance, images of dogs would be fed into the neural network, transformed into data, which then moves through the network, with various nodes assigning different weights to different elements. The final output layer collates this information, and classifies the image as a dog.

In this paper, we will attempt to explore the various methods of Convolutional Neural Networks (CNN) to improve on the model's image recognition ability, with the aid of Tensorflow.

II. RELATED WORKS

As CIFAR-10 is a relatively popular dataset, multiple machine research learning projects have been published online, with variations to the models built to predict the images in the dataset. Most of the research included using CNN to build models, and some has achieved accuracy of as high as 80% in predicting the classes to which the images in the dataset belonged to. One of the notable works was the CNN model created by Jason B, where he combined the baseline model, together with increasing dropout, data augmentation, and batch normalization, to achieve an accuracy score of 88.62% on its test dataset.

III. DATASET

The CIFAR-10 (Canadian Institute For Advanced Research) is a collection of pictures used mainly for training of machine learning and algorithms. It is one of the most popular datasets used for machine learning research, and it consists of 60,000 32x32 colour images in 10 different classes. The 10 different classes include airplanes, cars, birds, deer, dogs, frogs, horses, ships and trucks. Each class contains 6,000 images. Examples of some of the images in the dataset are shown below:

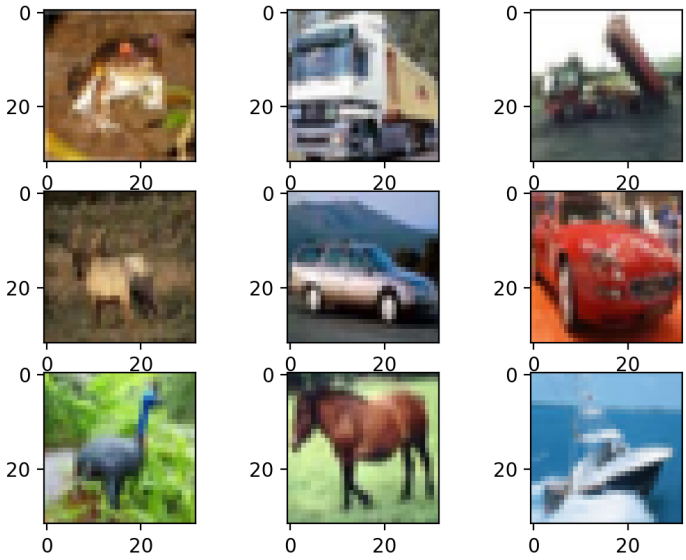

Figure 1: Preview of images generated from the CIFAR-10 dataset

The class labels and their standard associated integer values are listed below.

Class	Category
0	Airplane
1	Automobile
2	Bird
3	Cat
4	Deer
5	Dog
6	Frog
7	Horse
8	Ship
9	Truck

Figure 2: Categories and classes present in the CIFAR-10 dataset

IV. METHODOLOGY

Upon importing the CIFAR-10 dataset from the database, the y_train and y_test datasets are then encoded using 'to_categorical'. The X_train and X_test datasets are also converted to float values, and normalized by dividing the dataset values by 255. A simple model and a Convolutional Neural Network (CNN) model are then created.

Upon evaluation from the results from these 2 models, the CNN model seemed to be a better performing option with higher accuracy scores. We then attempt to tune some of the parameters and retrain the model based on the most ideal parameters provided. The results of the final CNN model built are then evaluated using metrics such as its accuracy/error scores, as well as

using confusion matrix. The final model is then used to randomly predict images and identifying the class to which the image belongs to.

V. EXPERIMENT

1. SIMPLE NEURAL NETWORK MODEL

A simple model was first built by initiating a Sequential model, followed by addition of the first hidden layer, and the output layer.

```
# create model
model = Sequential()
# add the first hidden layer
model.add(Dense(3072, input_shape=(3072,),
                kernel_initializer='normal', activation='relu'))
# add the output layer
model.add(Dense(num_classes,
                kernel_initializer='normal', activation='softmax'))
# Compile model
model.compile(loss='categorical_crossentropy',
              optimizer='adam', metrics=['accuracy'])

model.summary()
```

Figure 3: Screenshot of a simple neural network built to predict CIFAR-10 dataset images

The accuracy/Value accuracy and its Loss/Value loss were then plotted on a graph as shown below. It seems that there may be room for improvement, by building a better model to predict the CIFAR-10 dataset images.

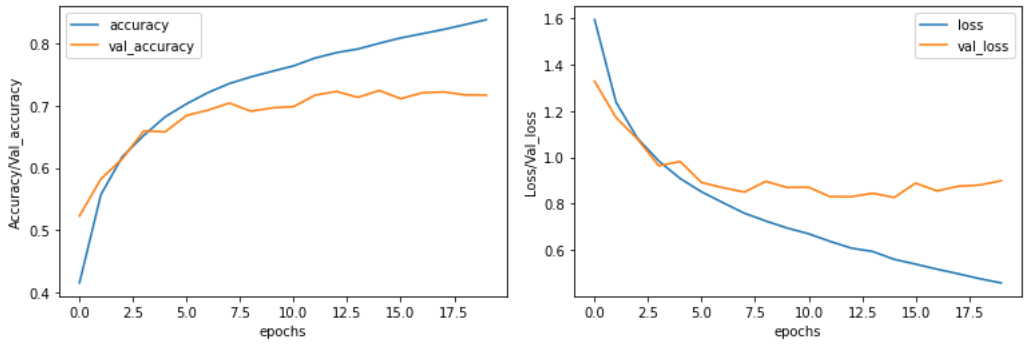

Figure 4: Charts of accuracy/value accuracy and its respective loss/value losses on the model's prediction

Overall, the model achieved an accuracy of 52.5% in predicting the categories of images present in the test dataset. The next section of the paper will then focus on construction of the CNN model to achieve better accuracy on predicting the categories of images in CIFAR-10 dataset.

2. CNN MODEL

Generally, the CNN model is constructed as per the method below.

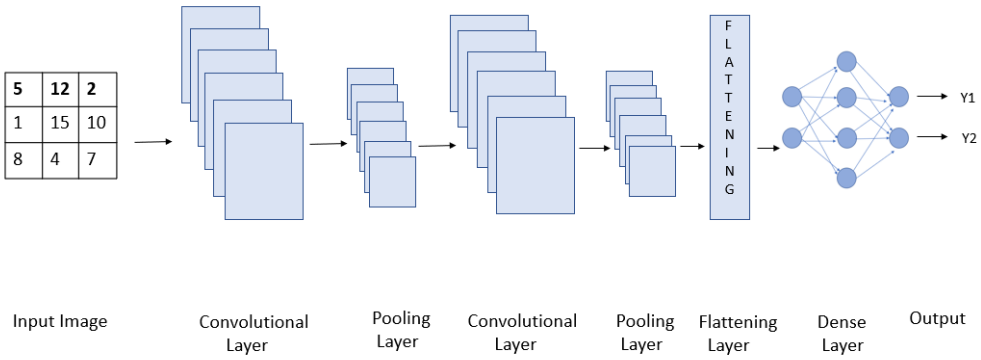

Figure 5: Steps executed in building a CNN model

The first CNN model used in this analysis was then built based on the steps prescribed in the screenshot below.

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(10))

# compile model
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

Figure 6: Steps executed in building a CNN model for this analysis

The accuracy/Value accuracy and its Loss/Value loss were then plotted on a graph as shown below. It seems that the discrepancy between accuracy and loss against its respective value accuracy and losses increased as the epochs becomes higher. This may signify that overfitting is present in this model.

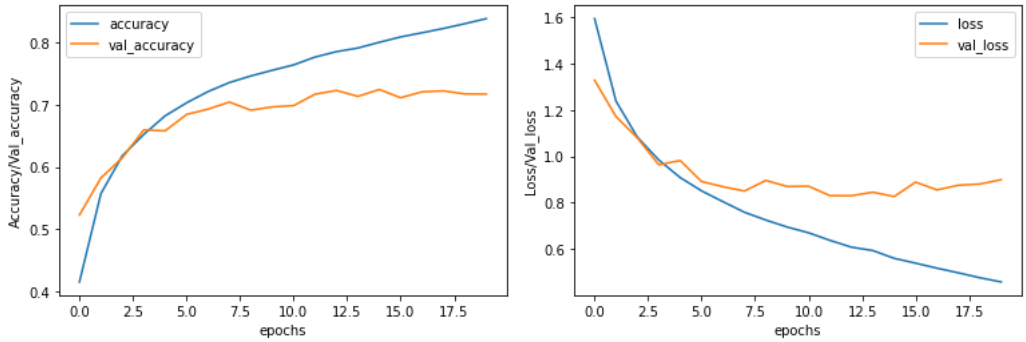

Figure 7: Charts of accuracy/value accuracy and its respective loss/value losses on the model's prediction

Nevertheless, the CNN model above achieved an accuracy score of 71.74% on test datasets. The CNN model will then be fine-tuned, with its results shared in the remaining section(s) of this research paper.

3. FINE-TUNING THE CNN MODEL

The CNN model was then fine-tuned by adding more layers, as per the screenshot below.

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape=(32, 32, 3)))
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.2))
model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.2))
model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
model.add(Dropout(0.2))
model.add(Dense(10, activation='softmax'))

# compile model
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

Figure 8: Additional steps/ layers added to fine tune the CNN model

Next, the model was then fitted and compiled, and it yielded the below accuracy/loss against its respective value accuracy and value losses below.

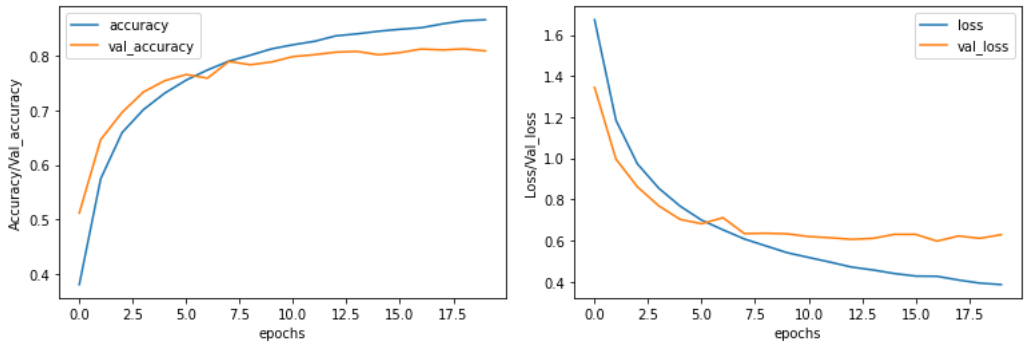

Figure 9: Charts of accuracy/value accuracy and its respective loss/value losses on the model's prediction

The fine-tuned CNN model achieved an accuracy score of 80.80%.

VI. DISCUSSION

Overall, the CNN model achieved the best accuracy score. As seen from the confusion matrix below, the model seems to have the lowest accuracy in predicting images of birds, whilst it had the highest accuracy in predicting ships.

This is also backed up by the research done by Qun Liu, whom's CNN model had the lowest accuracy scores in predicting birds and the highest in predicting ships. Its model was able to predict 708 images of birds, and 933 images of ships accurately, similar to the results below.

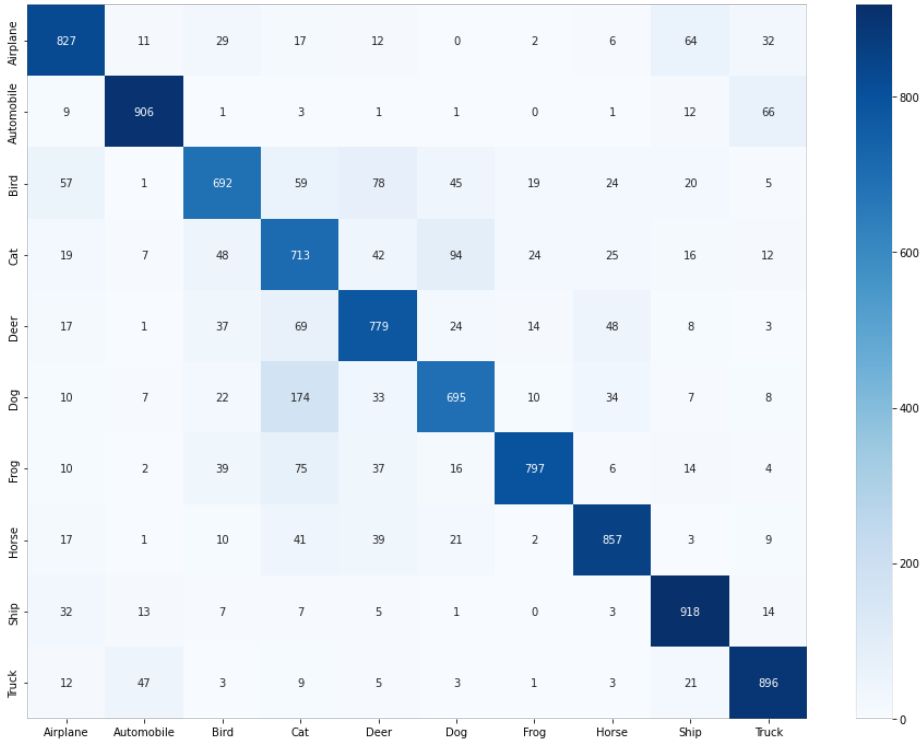

Figure 10: Additional steps/ layers added to fine tune the CNN model

Perhaps, in this dataset, animals were harder to predict due to its similarities in appearances, whilst images of ships were easier to predict as their appearances were more distinct, and hence easily recognised and categorised accurately by the model.

VII. CONCLUSION

The CNN model has proven to be a more useful model in predicting the classes to which the CIFAR-10 dataset images belong to. The accuracy score is further enhanced when additional layers are added to the model, ultimately achieving an accuracy score of 80.80%.

VIII. REFERENCES

1. Jason B. “How to Develop a CNN From Scratch for CIFAR-10 Photo Classification” Retrieved, August 4, 2022 from https://machinelearningmastery.com/how-to-develop-a-cnn-from-scratch-for-cifar-10-photo-classification/

2. Qun Liu. “Confusion Matrix for the CIFAR-10 dataset“ Retrieved, August 12, 2022 from https://www.researchgate.net/figure/Confusion-matrix-for-the-CIFAR-10-dataset_fig4_328400178