IMPROVING ON CIFAR-10 IMAGE CLASSIFICATION ACCURACY USING CONVOLUTIONAL NEURAL NETWORK (C N N)

Folorunso, O. David
MSc. Computer Science, University of East London
dofoll@yahoo.com

Abstract: Image processing in recent years has become part of human daily lives, especially with the exploding breakthrough in mobile technology with such computing power never before accessed. With AI enabled devices image can be recognised. This interest is the motivation for the classification of image dataset. The study sought to improve classification accuracy of CIFAR-10 dataset using Convolutional Neural Network (CNN). The default parameters of the classification model were tweaked for better accuracy. While the training dataset achieved 96% accuracy, the model achieved 74% on the validation dataset, which is still an acceptable performance. Among the 10 classes in the dataset, the model performed best in classifying automobile but struggled with cat. This observation is a baseline for another research.

1.0 Introduction

This study aims to improve classification accuracy of the CIFAR-10 Dataset. A fundamental job with several applications, such as object recognition, picture captioning, face and emotion detection, is classifying an input image into one or more categories. The work may often be completed manually using a set of algorithms created by a digital image processing specialist or automatically using a machine learning classifier. The picture is pre-processed before being given into a classifier in many instances, combining both methods. To solve the classification challenge, increasingly complicated approaches and models are needed as the amount and complexity of the data grow rapidly. Because of this, many data scientists are turning to deep learning to address the categorization challenge. (Chen & Ran, 2019). Convolutional neural networks are the most often utilized neural network classification technique (CNN). Self-optimizing artificial neurons are used in convolutional neural networks, which function similarly to artificial neural networks (ANNs). Convolutional layer, pooling, and fully linked layers make up CNN's three layers.

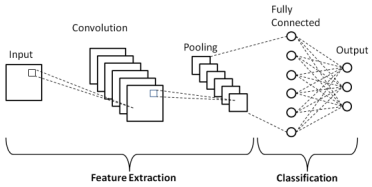

Figure 1: A typical convolutional neural network

Convolutional layers' primary objective is to create feature maps for an image by sliding a smaller matrix (a filter or kernel) over the entire image. The most important data features were preserved after the feature maps were reduced. By connecting the bottom-most neurons in the previous layer to the top-most neurons in the following layer, the Softmax layer, that will output the prediction, flattens the input matrix from the previous layer (Vijayaraj et al., 2022). Because of the model's design, it may be trained with less datasets, which requires less parameter learning. Shared weights are used by CNNs to expedite processing. The accuracy and effectiveness of applications are improved by CNN, which has excelled at machine learning applications

In this study, some of the numerous models that have been used to CIFAR-10 categorization are briefly summarized. However, CNN will make a substantial contribution by addressing the multi-class categorization's accuracy. The table below,

gotten from Wikipedia website contains the paper titles of previous neural networks.

Paper title	Error rate (%)	Publication date
Convolutional Deep Belief Networks on CIFAR-10[6]	21.1	August, 2010
Maxout Networks[7]	9.38	February 13, 2013
Wide Residual Networks[8]	4.0	May 23, 2016
Neural Architecture Search with Reinforcement Learning[9]	3.65	November 4, 2016
Fractional Max-Pooling[10]	3.47	December 18, 2014
Densely Connected Convolutional Networks[11]	3.46	August 24, 2016
Shake-Shake regularization[12]	2.86	May 21, 2017
Coupled Ensembles of Neural Networks[13]	2.68	September 18, 2017
ShakeDrop regularization[14]	2.67	Feb 7, 2018
Improved Regularization of Convolutional Neural Networks with Cutout[15]	2.56	Aug 15, 2017
Regularized Evolution for Image Classifier Architecture Search[16]	2.13	Feb 6, 2018
Rethinking Recurrent Neural Networks and other Improvements for Image Classification[17]	1.64	July 31, 2020
AutoAugment: Learning Augmentation Policies from Data[18]	1.48	May 24, 2018
A Survey on Neural Architecture Search[19]	1.33	May 4, 2019
GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism[20]	1.00	Nov 16, 2018

Table 1: Research papers claiming state-of-the-art results on CIFAR-10

Maxout networks (Goodfellow, Ian et al., 2013) take into account the challenge of developing models that make use of the dropout approximation model averaging approach. It provides a straightforward new model called maxout—so named because it is a natural partner to dropout and because its output is the maximum of a set of inputs—that is intended to both speed up optimization by dropout and increase the precision of its quick approximation model averaging method. The model successfully completes both of these objectives, as demonstrated by the researchers' empirical tests.

Wide Residual Networks attempt to address the issue of declining feature reuse (Zagoruyko & Komodakis, 2016). It has been demonstrated that deep residual networks may scale up to thousands of layers while still performing better. However, each tenth of a percent increase in accuracy requires roughly doubling the number of layers, making extremely deep residual networks difficult to train because feature reuse is decreasing. To address these issues, a thorough experimental analysis of the ResNet block design was carried out, and on the basis of this research, a novel architecture was presented in which the depth and width of residual networks were both enhanced.

A recurrent network is used to produce model descriptions for neural networks in Neural Architecture Search with Reinforcement Learning (Zoph and Le, 2016). This RNN is then trained with reinforcement learning to increase the predicted accuracy of the generated structures on a testing set. Beginning from scratch and using the CIFAR-10 dataset, the approach created a unique network architecture that, in terms of accuracy of the test set, is at par with the best human-invented architecture. The approach can create a unique recurrent cell on the Penn Treebank dataset that beats the commonly used LSTM cell and other cutting-edge baselines. The cell outperforms the prior state-of-the-art model by 3.6 perplexities, achieving a test set perplexity of 62.4 on the Penn Treebank. The cell may also be used for the character language modeling problem on PTB, where it obtains a modern perplexity of 1.214.

In densely connected convolutional networks, each layer is connected to every other layer in a feed-forward manner (Huang, Liu et al., 2016). Our network features L(L+1)/2 direct connections as opposed to standard convolutional networks with L layers having L connections, one between each layer and its succeeding layer. The feature-maps of all layers before it are utilized as inputs for each layer, and its own feature-maps are used as inputs for all levels after it. DenseNets offer a variety of appealing benefits, including the elimination

of the vanishing-gradient issue, improved feature propagation, promoted feature reuse, and much fewer parameters.
This study seeks to improve the accuracy ever achieved in the classification of CIFAR-10 dataset using Convolutional Neural Network (CNN).

2.0 Architecture of Convolutional Neural Network (CNN)

Convolutional Neural Networks, often known as CNNs, are a subset of artificial neural networks used in deep learning and are frequently employed for object and picture identification and categorization. CNNs are being used extensively in a variety of tasks and activities, including voice recognition in natural language processing, video analysis, and difficulties with image processing, computer vision, and self-driving car obstacle detection. Components of a neural network are input layer, hidden layers, and an output layer.
CNNs draw inspiration from the structure of the human brain. Artificial neurons found in CNNs, receive inputs like brain cortex, process the input it receives, and then present the result as output, this process is typical of how neurons in the brain process information throughout the body. The image serves as the input. As an input, the input layer receives arrays of image pixels.

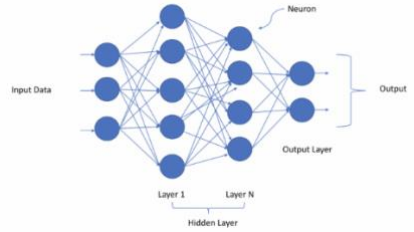

Figure 2: A Simple Neural Network

CNNs may have numerous hidden layers that utilize calculations to extract information from images; examples of these layers are convolution, pooling, rectified linear units, and fully connected layers. Convolution is the initial layer used to extract features from an input image. (Rawat and Wang, 2017). When a n x n convolution kernel convolutions a m x m picture with the same depth, we obtain a new image of size ((m + n)/l) +1, which is the feature map. The item is classified and identified in the output layer by the fully connected layer. Both CNNs and artificial neural networks (ANN) are based on biological principles. Their architecture is influenced by the visual cortex of the brain, which is composed of alternating layers of basic and sophisticated cells.

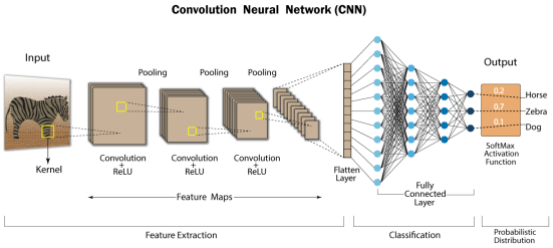

Figure 3: Convolutional Neural Network

Although there are several CNN designs, they all have convolutional and pooling layers that are grouped into modules. Following these modules, one or more fully linked layers are present, similar to a normal feedforward neural network. (Yamashita, Nishio, Do et al., 2018)
Due to CNNs' ability to see patterns and interpret them, our approach to image identification has been fundamentally altered. They are recognized as the most efficient architecture for image classification, retrieval, and detection applications due to the high degree of accuracy in their output. CNNs have a variety of applications in real-world testing, where they provide high-quality results and successfully locate and identify objects in images such as people, automobiles, birds, and so on. They are now the preferred technique for predictions using any picture as an input because of this characteristic. The capacity of CNNs to accomplish "spatial invariance," which denotes that they can learn to detect and extract visual information from any location in the image, is a crucial quality. Since CNNs automatically learn features from the image/data and carry out extraction from pictures, manual extraction is not required. As a result, CNNs are an

effective Deep Learning technique for producing precise results.
According to the study published in "Neural Computation" (Orchard & Phillips, 1991), the purpose of the pooling layers is to limit the spatial resolution of the feature maps and therefore achieve spatial invariance to input distortions and translations. Processing speed rises as the pooling layer reduces the number of parameters needed to process the image while also reducing memory utilization and computational price. Although CNNs have typically been used for image analysis, they may also be used for other types of data analysis and categorization. As a result, they may be utilized to produce reliable findings in a range of businesses, including feature detection, video classification, street sign recognition, galaxy categorization, and interpretation and diagnosis/analysis of medical images, among others.

2.1 Dataset

CIFAR-10 dataset was used in this research, which is a labeled dataset containing 60,000 images from Canadian Institute for Advance Research. These 60,000 images are a subset of the larger dataset of 80 million small images (Alex, 2010), which is a collection of 32 32 color images collected by searching multiple web image search engines for all non-abstract English nouns and noun phrases in the WordNet lexical database (Miller, 1995).
CIFAR-10 (60,000 images subset used in this research project) contains ten classes: deer, ship, bird, cat, dog, automobile, frog, truck, airplane and horse. Each class contains exactly 6,000 images and the classes are completely mutually exclusive. It was collected by Alex Krizhevsky (Alex, 2010), Vinod Nair, and Geoffrey Hinton. The class labels and their standard associated integer values are listed below.

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

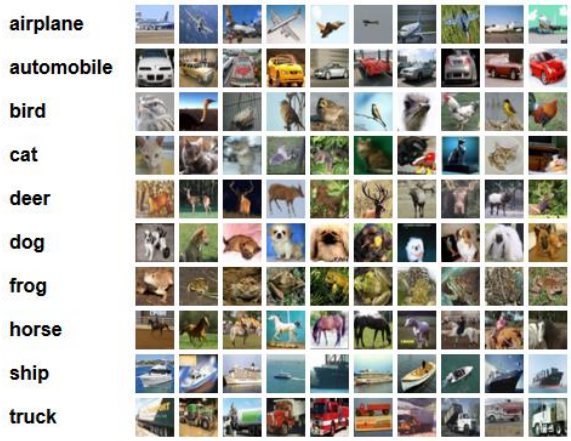

Figure 4: CIFAR-10 class names with sample pictures

3.0 Implementation

The training of the deep learning model was carried out on Windows PC with the following configurations.

Operating System: Windows 11 Pro 64-bit (10.0, Build 22000) (22000.co_release.210604-1628)
Processor: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz (8 CPUs), ~1.8GHz
Memory: 16384MB RAM
IDE Environment: Jupyter Notebook running on Anaconda.

3.1 The CNN Algorithm

This research was carried out on the CIFAR-10 image dataset using Convolutional Neural Network (CNN) for classification.
Convolutional layer: Convolution layers, which combine convolution operation with activation function to conduct feature extraction, are a crucial part of the CNN design.

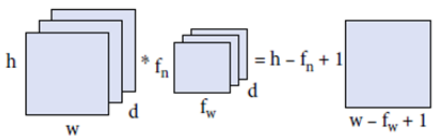

Figure 5: Image kernel or filter kernel

Utilizing a little input square, the convolutional layer retains the association between pixels. It involves two steps: An

image matrix must be created first. Second, a kernel or filter is used to process this picture matrix. The output's length, width, and height are (h-fh+1) (w-fw+1) 1, and the image matrix's width, height, and depth are h w d, fh fw d, and respectively. " This is known as “Features Map". This study's convolutional layer was set up with the following parameters: input shape is 32, filters are 512 (multiple of 64), kernel size is 3 x 3, pooling is 2 x 2, and activation is ReLU (Rectified Linear Unit).

Padding: The CNN kernel's padding specifies how many pixels are added to an image as it processes it. In order to build a convolutional neural network, padding is necessary. We will obtain a tiny, filtered image if we use a neural network with several layers and compress the original image. The padding used in this CIFAR-10 study is (2,2).

Max Pooling: From the input feature maps, patches are extracted using max pooling. Each patch outputs the largest value while discarding all other values. Feature maps' depth dimension does not vary, in contrast to height and breadth.

Activation function: A nonlinear activation function is then applied to the results of a linear process, such as convolution. Smooth nonlinear functions, such as the sigmoid or hyperbolic tangent (tanh) function, have traditionally been utilized as mathematical representations of the activity of actual neurons. The function is computed by ReLU (Rectified Linear Units): f(x) = max(0, x) (Yamashita et al, 2018).

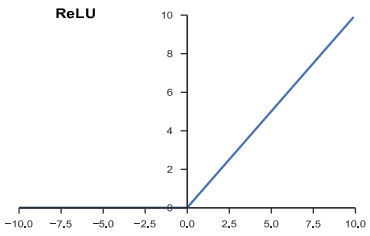

Figure 6: ReLU activation function

Output Layer Activation Function: The last completely linked layer often receives a different activation function than the others. Each work needs a unique activation function, which must be selected properly.. A SoftMax function is utilized as the activation function for the multiclass classification task, normalizing the output actual values from the final fully linked layer to aim class probabilities, in which each value varies between 0 and 1 and all the values amount to 1.

$$\sigma(x_j) = \frac{e^{x_j}}{\sum_i e^{x_i}}$$

Figure 7: Formula for SoftMax function

3.3 The Codes

```
import keras

from keras.models import Sequential
from keras.layers import Dense

#Checking the backend that keras is using
keras.backend.backend()
 'tensorflow'

import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import numpy as np
import matplotlib.pyplot as plt

#Importing data
from keras.datasets import cifar10

#Reading the imported data
(X_train, y_train), (X_test, y_test) = datasets.cifar10.load_data()

#No of images in the train dataset
X_train.shape
(50000, 32, 32, 3)

#No of images in the test dataset
X_test.shape
(10000, 32, 32, 3)

y_train.shape
(50000, 1)

y_test.shape
(10000, 1)

#Checking type of array of the dataset
X_train[5:]
array([[[[159, 102, 101],
        [150,  91,  95],
        [153,  95,  97],
        ...
        ...
        [179, 177, 173],
        [164, 164, 162],
        [163, 163, 161]]]], dtype=uint8)

y_train[:5]
array([[6],
```

```
      [9],
      [9],
      [4],
      [1]], dtype=uint8)

#Converting y_train and y_test from 2d array to 1d array, since 1d array is good enough for the classification
y_train = y_train.reshape(-1,)
y_train[:5]
array([6, 9, 9, 4, 1], dtype=uint8)

y_test = y_test.reshape(-1,)
y_test[:5]
array([3, 8, 8, 0, 6], dtype=uint8)

#Visualizing the first image
plt.matshow(X_train[0])
<matplotlib.image.AxesImage at 0x202145dde80>

#Introducing the classes of the dataset for proper labelling
classes = ["airplane","automobile","bird","cat","deer","dog","frog","horse","ship","truck"]

def plot_preview(X, y, index):
    plt.figure(figsize = (15,2))
    plt.imshow(X[index])
    plt.xlabel(classes[y[index]])

plot_preview(X_train, y_train, 3)
plot_preview(X_train, y_train, 8)

#Normalising dataset (x_train and x_test), so that the values of the variable would range between 0 and 1
x_train = x_train / 255
x_test =x_test / 255

#Checking out the resut of normalisation
x_train[0]
array([[[0.23137255, 0.24313725, 0.24705882],
        [0.16862745, 0.18039216, 0.17647059],
        [0.19607843, 0.18823529, 0.16862745],
        ...,
        ...,
        [0.84705882, 0.72156863, 0.54901961],
        [0.59215686, 0.4627451 , 0.32941176],
        [0.48235294, 0.36078431, 0.28235294]]])

#Building convolutional neural network for image classification
cnn = models.Sequential([
    layers.Conv2D(filters=512, kernel_size=(3, 3), activation='relu', input_shape=(32, 32, 3)),
    layers.MaxPooling2D((2, 2)),

    layers.Conv2D(filters=512, kernel_size=(3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),

    layers.Conv2D(filters=512, kernel_size=(3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),


    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])

cnn.summary()
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 30, 30, 512)       14336

 max_pooling2d_3 (MaxPooling (None, 15, 15, 512)       0
 2D)

 conv2d_4 (Conv2D)           (None, 13, 13, 512)       2359808

 max_pooling2d_4 (MaxPooling (None, 6, 6, 512)         0
 2D)

 conv2d_5 (Conv2D)           (None, 4, 4, 512)         2359808

 max_pooling2d_5 (MaxPooling (None, 2, 2, 512)         0
 2D)

 flatten_1 (Flatten)         (None, 2048)              0

 dense_2 (Dense)             (None, 64)                131136

 dense_3 (Dense)             (None, 10)                650

=================================================================
Total params: 4,865,738
Trainable params: 4,865,738
Non-trainable params: 0
_________________________________________________________________

#Setting up the compiler
cnn.compile(optimizer='adam',
            loss='sparse_categorical_crossentropy',
            metrics=['accuracy'])

#Training the dataset
cnn.fit(X_train, y_train, epochs=20)
Epoch 1/20
1563/1563 [==============================] - 1109s 709ms/step - loss: 1.5184 - accuracy: 0.4469
Epoch 2/20
1563/1563 [==============================] - 1228s 786ms/step - loss: 1.0812 - accuracy: 0.6209
Epoch 3/20
1563/1563 [==============================] - 1219s 780ms/step - loss: 0.8791 - accuracy: 0.6956
Epoch 4/20
1563/1563 [==============================] - 1220s 780ms/step - loss: 0.7380 - accuracy: 0.7422
Epoch 5/20
1563/1563 [==============================] - 4597s 3s/step - loss: 0.6244 - accuracy: 0.7831
Epoch 6/20
1563/1563 [==============================] - 1218s 780ms/step - loss: 0.5305 - accuracy: 0.8150
Epoch 7/20
1563/1563 [==============================] - 1244s 796ms/step - loss: 0.4448 - accuracy: 0.8442
Epoch 8/20
1563/1563 [==============================] - 1253s 802ms/step - loss: 0.3742 - accuracy: 0.8693
Epoch 9/20
1563/1563 [==============================] - 1229s 787ms/step - loss: 0.3128 - accuracy: 0.8889
Epoch 10/20
1563/1563 [==============================] - 1221s 781ms/step - loss: 0.2627 - accuracy: 0.9073
Epoch 11/20
1563/1563 [==============================] - 1240s 793ms/step - loss: 0.2254 - accuracy: 0.9200
Epoch 12/20
1563/1563 [==============================] - 1237s 791ms/step - loss: 0.2009 - accuracy: 0.9300
Epoch 13/20
1563/1563 [==============================] - 1243s 796ms/step - loss: 0.1775 - accuracy: 0.9368
Epoch 14/20
1563/1563 [==============================] - 1240s 793ms/step - loss: 0.1596 - accuracy: 0.9433
Epoch 15/20
```

```
1563/1563 [==============================] - 1198s
766ms/step - loss: 0.1453 - accuracy: 0.9476
Epoch 16/20
1563/1563 [==============================] - 1151s
736ms/step - loss: 0.1338 - accuracy: 0.9532
Epoch 17/20
1563/1563 [==============================] - 1148s
734ms/step - loss: 0.1285 - accuracy: 0.9553
Epoch 18/20
1563/1563 [==============================] - 1162s
743ms/step - loss: 0.1247 - accuracy: 0.9569
Epoch 19/20
1563/1563 [==============================] - 1151s
736ms/step - loss: 0.1123 - accuracy: 0.9612
Epoch 20/20
1563/1563 [==============================] - 1136s
727ms/step - loss: 0.1057 - accuracy: 0.9632
<keras.callbacks.History at 0x20214811730>
```

```
#Running classification report
from sklearn.metrics import confusion_matrix , classification_report
import numpy as np
y_pred = cnn.predict(X_test)
y_pred_classes = [np.argmax(element) for element in y_pred]

print("Classification Report: \n", classification_report(y_test, y_pred_classes))
```

```
313/313 [==============================] - 39s 125ms/step
Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.73      0.74      1000
           1       0.88      0.85      0.86      1000
           2       0.55      0.69      0.61      1000
           3       0.62      0.47      0.53      1000
           4       0.69      0.66      0.67      1000
           5       0.65      0.62      0.63      1000
           6       0.77      0.80      0.79      1000
           7       0.76      0.74      0.75      1000
           8       0.81      0.83      0.82      1000
           9       0.75      0.83      0.79      1000

    accuracy                           0.72     10000
   macro avg       0.72      0.72      0.72     10000
weighted avg       0.72      0.72      0.72     10000
```

```
#Evaluating the model
cnn.evaluate(X_test,y_test)
313/313 [==============================] - 7s 22ms/step - loss: 1.7730 - accuracy: 0.7402
[1.7730391025543213, 0.7401999831199646]
```

4.0 Experimental Result

4.1 Training

Locating the convolution layer kernels and the fully connected layer weights on a training dataset that reduce disparities between output predictions and supplied test dataset labels constitutes the process of training a network. In the back propagation method, which is frequently used to train neural networks, the loss function and the gradient descent optimization methodology are important factors. Using a loss function that uses forward propagation on a training dataset and trainable parameters, such as kernels and weights, the accuracy of a model under specific kernels and weights is measured. The CIFAR-10 image collection was categorized using a convolutional neural network (CNN). It was done using the IDE environment for Jupyter Notebook.

The following was the setting for the convolution layer in the model.

```
cnn = models.Sequential([
    layers.Conv2D(filters=512, kernel_size = (3, 3), activation='relu', input_shape=(32, 32, 3)), layers.MaxPooling2D((2, 2)),

    layers.Conv2D(filters=512, kernel_size=(3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),

    layers.Conv2D(filters=512, kernel_size=(3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
```

The dataset was trained using CNN (Convolutional Neural Network), a deep learning framework, to provide the experimental findings listed below. The model was trained, and it now has a weighted average accuracy of 72%; the precision value of each of the 10 classes in the CIFAR-10 dataset, together with recall and f1-score, are listed in the table below.

Classification Report:				
class	**precision**	**recall**	**fl-score**	**support**
0	0.75	0.73	0.74	1000
1	0.88	0.85	0.86	1000
2	0.55	0.69	0.61	1000
3	0.62	0.47	0.53	1000
4	0.69	0.66	0.67	1000
5	0.65	0.62	0.63	1000
6	0.77	0.8	0.79	1000
7	0.76	0.74	0.75	1000
8	0.81	0.83	0.82	1000
9	0.75	0.83	0.79	1000
accuracy			0.72	10000
macro avg	0.72	0.72	0.72	10000
weighted avg	0.72	0.72	0.72	10000

Table 2: Classification Report on the CIFAR-10 training

Precision: Precision is the ratio of correctly predicted positive outcomes to all positive outcomes. The capacity of a classifier to not categorize a case that is genuinely negative as positive. It is described for each class as the proportion of true positives to the total of true positives and false positives.
Precision = TP/(TP + FP)
TP = True Positives, FP = False Positives
Recall: Percentage of correct positive predictions relative to total actual positives.
Recall = TP/(TP + FN).
FN = False Negatives, TP = True Positives
F1 Score: A weighted harmonic mean of precision and recall. The closer to 1, the better the model.

4.2 The Model Predictions

In a precision performance, 75% of predictions for an airplane were accurate, 88% of predictions for an automobile were accurate, 55% of predictions for a bird were accurate, 62% of predictions for a cat were accurate, 69% of predictions for a deer were accurate, 65% of predictions for a dog were accurate, 77% of predictions for a frog were accurate, 76% of predictions for a horse were accurate, and 81 percent of predictions for a ship were accurate.
In Recall performance of the model, out of the 5,000 true samples of each class used for training, the model predicted correctly 73% of the airplane sample, 85% of the automobile sample, 69% of the bird sample, 47% of the cat sample, 66% of the deer sample, 62% of the dog sample, 80% of the frog sample, 74% of the horse sample, 83% of the ship sample and 83% of the truck sample. The model performed well in classifying vehicles with a score of 0.86 and badly in categorizing cats, according to the f1-score; the f1-score is best when it is near to 1. The remaining classes' f1-scores are in this range (0.53 and 0.86). The average computational time for every training and validation cycle was 5 hours.

4.3 Testing

Accuracy of the model was evaluated after the training of the dataset. This was done using the test dataset of the CIFAR-10. The test dataset contains 10,000 images and it represented the 10 classes in the CIFAR-10 dataset. Each of the 10 classes has 1,000 images in the test dataset that was not part of the training dataset.

```
cnn.evaluate(X_test,y_test)
313/313 [==============================] - 7s 22ms/step - loss: 1.7730 - accuracy: 0.7402
[1.7730391025543213, 0.7401999831199646]
```

The accuracy of the CNN trained model in this research, from the evaluation computation above is 74%. The model classifies correctly 7 out of every 10 images.

4.4 Performance

In this research, the performance of Convolutional Neural Network (CNN) architectures was assessed for the CIFAR-10 dataset classification. The model was trained with 50,000 images. The images were of 10 classes and training report, classification report and validation report were generated for the model. The training report showed the increase of the training accuracy from 0.4469 on the first epoch to 0.9632 on the last epoch. From figure 11 the accuracy climbs fast between epoch 1 and 10 but plateaued from epoch 11 to 20. The accuracy started 0.4469 on epoch 1 and climbed to 0.9073 on epoch 10; but ended on 0.9632 on epoch 20.
The model performed well on training dataset, ending on 96.32% accuracy in correctly classifying images it was trained on with just 20 epochs. On the loss side, the value dropped from 1.5184 to 0.1057 on the last epoch. The drop in loss value also slowed down at epoch 10, reflecting what was happening to the training accuracy of the model at the same time. Classification report recorded accuracy of 0.72 on f1-score. The macro and weighted averages are also 0.72. According to Joshi, P. (2017, pg69),

"The performance of a classifier is characterized by precision, recall, and f1-scores. Precision refers to the accuracy of the classification and recall refers to the

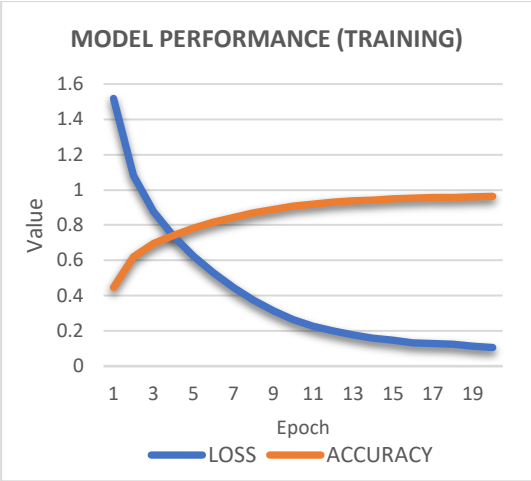

Figure 8: Model Performance on Training Dataset

number of items that were retrieved as a percentage of the overall number of items that were supposed to be retrieved. A good classifier will have high precision and high recall, but it is usually a tradeoff between the two. Hence, we have f1-score to characterize that. F1 score is the harmonic mean of precision and recall, which gives it a good balance between precision and recall values".

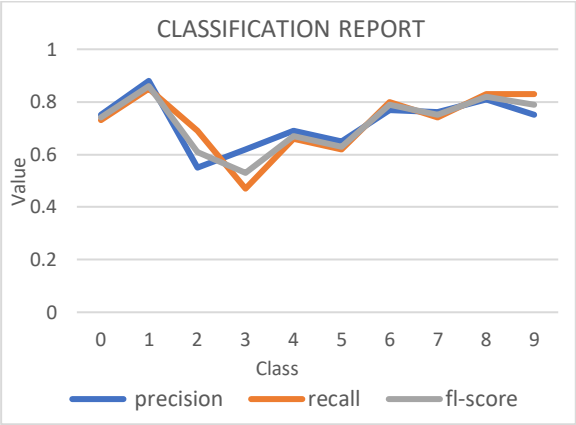

Figure 9: CNN Classification Report Chart

A model is at its finest after it has demonstrated good performance in both the training and test datasets. This research CNN model recorded 96% accuracy on training dataset and 74% accuracy on test/validation report.

```
cnn.evaluate(X_test,y_test)
313/313 [==============================] - 7s 22ms/step - loss: 1.7730 - accuracy: 0.7402
[1.7730391025543213, 0.7401999831199646]
```

5.0 Analysis

Decisions: In setting up the parameters for the convolutional layer, I had input shape of 32. 512 filters were used for the layer. Kernel size was (3,3), the poling layer was Max Pooling with (2,2) parameters and the activation layer for the Conv2D was ReLU. Three convolutional layers were set up with the above parameters. The activation layer for the fully connected layer was ReLU and output layer was Softmax. This configuration brings the total number of model trainable parameters to 4, 865,738.

```
layers.Conv2D(filters=512, kernel_size=(3, 3), activation='relu'', input_shape=(32,

layers.MaxPooling2D((2, 2)),
layers.Conv2D(filters=512, kernel_size=(3, 3), activation='relu'), layers.MaxPooling2D((2, 2)),

layers.Conv2D(filters=512, kernel_size=(3, 3), activation='relu'), layers.MaxPooling2D((2, 2)),

layers.Flatten(),
layers.Dense(64, activation='relu'),
layers.Dense(10, activation='softmax')
```

Justifications: The convolutional layer was setup with filter 512 been multiple of 64. The input size was 32 because the CIFAR-10 dataset has dimension of 32 x 32 coloured images. Max pooling was (2, 2) so that the corner pixels of the images can contribute to the feature extraction multiple times like the other parts of the images. ReLU was the activation function in other to avoid the complex computation involved in Sigmoid function activation; with ReLU the forward and backward propagations are just 'if' statement. It simplifies the computation unlike Sigmoid that computes an exponent. Another reason is model that is trained with ReLU converges quickly, thus it takes much less time compared to models that are trained

with Sigmoid. On the fully connected layer, Softmax activation function was used because of the multi-classification nature of the research and also Softmax output probabilities sum to one unlike Sigmoid.

$$\text{softmax}(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ for j = 1,...,K}$$

The training was compiled using Sparse categorical crossentropy for the classification. One advantage of sparse categorical cross entropy is that it saves memory and calculation time by employing a single integer for a class rather than an entire vector.

Observations & Explanations: The model learned the training dataset, demonstrating an enhancement on the training dataset which lasted 20 epochs. This is a favorable indicator since it indicates that the problem can be learned and also that the model has enough ability to learn the dataset. It was observed that the training accuracy rose rapidly from 0.4469 on the 1st epoch to 0.9073 on the 10th epoch but plateaued from epoch 11 to 20. The accuracy started 0.4469 on epoch 1 and climbed to 0.9073 on epoch 10; but ended on 0.9632 on epoch 20. The model performed well on the training dataset, achieving 96.32 percent accuracy in accurately categorizing photos after just 20 epochs. On the loss side, the value fell from 1.5184 to 0.1057 on the most last epoch. The decline in loss value likewise slowed around epoch 10 up until epoch 20, mirroring what was going on with the model's training accuracy at the same time. Model performance was average for cat class with f1-ratio of 0.53 while it performed best in predicting automobile with 0.86 f1-ratio. It may be that the cat images were blurred or bad beyond recognition; it could also be that the parameters used did not well in classifying the cat class, though this is a rare situation.

6.0 Conclusions

This research work proposed an improvement on the Convolutional Neural Network (CNN) model to classify CIFAR-10 image dataset. The proposed model was trained on a dataset that included the 60,000 images of 10 classes. This model performs slightly lower than the best of research papers known and listed on Wikipedia. The validated accuracy of the model was 74.01% while the error stood at 1.77. Therefore, a new CNN architecture is proposed for future research. Such research would run on higher configuration neural machine than the one used in this research. And also run hundreds of epoch in place of 20 epochs in this research. The Convolutional layer would run with filter 1024 (32 x 32) against 512 used in this model. More broadly, the findings indicate that it may be worthwhile to examine approaches that slow the pace of learning of the cat class in the model. This might involve approaches like data augmentation, learning rate plans, batch size modifications, and possibly more.

7.0 References

Abbasnejad, Ehsan & Shi, Qinfeng & Abbasnejad, Iman & Hengel, Anton & Dick, Anthony. (2017). Bayesian Conditional Generative Adverserial Networks.

Alex Krizhevsky https://www.cs.toronto.edu/~kriz/conv-cifar10-aug2010.pdf

A. Vijayaraj, P. T. Vasanth Raj, R. Jebakumar, P. Gururama Senthilvel, N. Kumar, R. Suresh Kumar, R. Dhanagopal, "Deep Learning Image Classification for Fashion Design", Wireless Communications and Mobile Computing, vol. 2022, Article ID 7549397, 13 pages, 2022. https://doi.org/10.1155/2022/7549397

Chen, J., & Ran, X. (2019). Deep learning with edge computing: A review. Proceedings of the IEEE, 107(8), 1655-1674.

https://doi.org/10.1109/JPROC.2019.2921977
Convolutional Networks with Dense Connectivity – Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/A-5-layer-dense-block-with-a-growth-rate-of-k-4-Each-layer-takes-all-preceding_fig1_333337231
Goodfellow, Ian J.; Warde-Farley, David; Mirza, Mehdi; Courville, Aaron; Bengio, Yoshua (2013-02-13). "Maxout Networks". arXiv:1302.4389 [stat.ML].
https://www.semanticscholar.org/paper/Wide-Residual-Networks-Zagoruyko-Komodakis/1c4e9156ca07705531e45960b7a919dc473abb51
Huang, Gao & Liu, Zhuang & Pleiss, Geoff & van der Maaten, Laurens & Weinberger, Kilian. (2019). Convolutional Networks with Dense Connectivity. IEEE Transactions on Pattern Analysis and Machine Intelligence. PP. 1-1. 10.1109/TPAMI.2019.2918284.
Huang, Gao; Liu, Zhuang; Weinberger, Kilian Q.; van der Maaten, Laurens (2016-08-24). "Densely Connected Convolutional Networks". arXiv:1608.06993 [cs.CV].
Joshi, P., & Joshi, P. 2017, Artificial Intelligence with Python, Packt Publishing, Limited, Birmingham. Available from: ProQuest Ebook Central. [30 July 2022].
Miller, G. A., 1995, WordNet: a lexical database for English
Orchard, G. A., & Phillips, W. A. (1991). Neural Computation. Psychology Press. https://doi.org/10.1604/9780863772351
Wikipedia contributors. (2022, June 19). CIFAR-10. In Wikipedia, The Free Encyclopedia. Retrieved 06:32, July 30, 2022, from https://en.wikipedia.org/w/index.php?title=CIFAR-10&oldid=1093822012
W. Rawat and Z. Wang, "Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review," in Neural Computation, vol. 29, no. 9, pp. 2352-2449, Sept. 2017, doi: 10.1162/neco_a_00990.
Yamashita, R., Nishio, M., Do, R.K.G. et al. Convolutional neural networks: an overview and application in radiology. Insights Imaging 9, 611–629 (2018). https://doi.org/10.1007/s13244-018-0639-9
Zagoruyko, Sergey; Komodakis, Nikos (2016-05-23). "Wide Residual Networks". arXiv:1605.07146 [cs.CV]
Zoph, Barret; Le, Quoc V. (2016-11-04). "Neural Architecture Search with Reinforcement Learning". arXiv:1611.01578 [cs.LG].