ITRI626 ARTIFICIAL INTELLIGENCE II

Assignment 2

34292748
MNISI G

Contents

1 Introduction

Machine learning (ML) drives the digital age, from internet searches to posts on social media, making recommendations for users on Netflix, Spotify, and many more. It has become prevalent in our everyday goods such as the digital devices that we use. ML algorithms are utilized to identify features in photos, text-to-speech, and appropriate search outcomes [3]. All these areas make use of Deep Learning (DL) techniques. DL is an aspect of machine learning that is about the advancement of systems capable of learning [6].

A convolutional neural network (CNN) has been the most important network aspect in the field of deep learning [4]. It has received a great deal of media coverage in recent years due to its significant breakthroughs in a variety of fields, which include image processing, and computer vision. In this assignment, a CNN will be utilized to analyze the CIFAR-10 dataset, which will be further discussed below. After training, and observing the dataset, will delve into the results and analyze them, and make conclusions based on these.

2 CIFAR-10 dataset

The dataset utilized for identifying objects is (Canadian Institute For Advanced Research) CIFAR-10, which is labeled as a subset of 80 million small pictures [2]. The collection contains 60,000 32x32 pixel color photos, the '10' in the CIFAR-10 presents ten categories or classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck).

There are 6000 photos in each class. The training batch has 50,000 photos, whereas the test batch contains 10,000 images. Each class's test batch has 1000 photos chosen at random [2]. The algorithms of computers that recognize things in pictures frequently learn by example. CIFAR-10 is a collection of photographs intended to teach a computer to recognize items. The figure below shows a sample of the CIFAR-10 dataset:

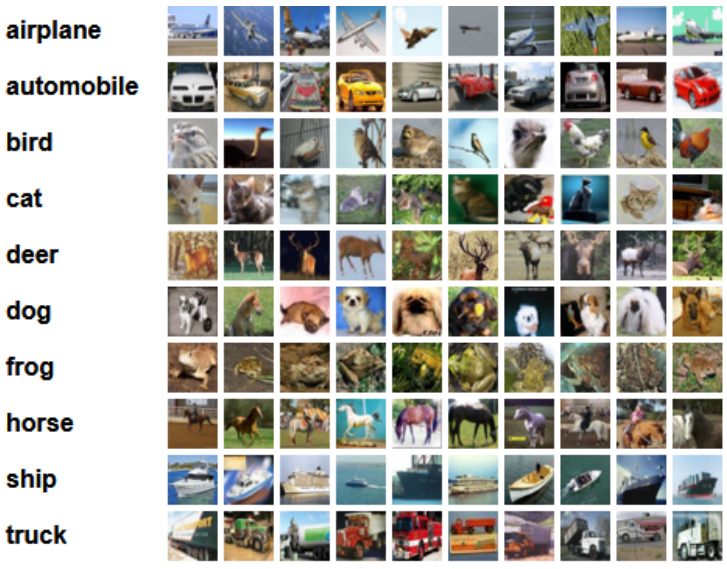

Figure 1: CIFAR-10 dataset

3 Convolutional Neural Network (CNN)

Deep learning techniques rely on neural networks, which are an aspect of machine learning [1]. They are made up of node layers, each of which has a layer for input, one or more hidden levels, and a layer for output. Each node is linked to another and has its own weight and threshold. If the output of any particular node exceeds the given threshold value, that node is activated and begins transferring data to the network's next tier. Otherwise, no data is sent to the next network layer.

Before the current CNNs, there were manual techniques for feature extraction and identifying objects which took a lot of time. Convolutional neural networks, on the other hand, now give a broader range of applications to image categorization and identification of object tasks, employing linear algebra concepts, notably matrix multiplication, to find patterns inside an image [1]. CNN is said to be computationally intensive, so it is advisable to use GPUs (Graphical Processing Units) to train models. A CNN model is made up of input, hidden, and output layers, with the hidden layers being made up of a convolutional layer, a Pooling layer, and Fully-connected layer [5].

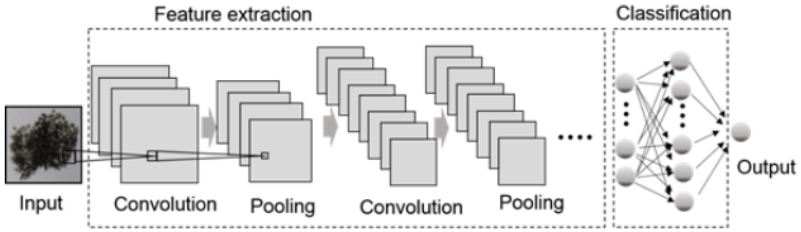

Figure 2: A convolutional neural network (CNN) architectural illustration

A CNN model normally involves two processes: feature extraction and classification. The above illustration is represented as a matrix of M $\times$ N in the feature extraction process, and the image properties (or features) are extracted in the convolution (convolutional) and pooling layers.

3.1 Convolutional layer

The convolutional layer is the foundation of a CNN model, it is where most of the processing takes place. It necessitates several elements, including a feature map, filter, and input map [3]. Assume the input is a color image composed of a 3D matrix of pixels. The input will consist of three aspects depth, width, and height which match the RGB color space of an image [1].

There is a feature detecter (known as filter or kernel) that traverses through the image's fields of reception, checking for the presence of the feature. This is referred to as a convolution. Filters and kernels can be used interchangeably. A filter is a collection of kernels, each of which interacts with a single channel of the input layer [1]. Each channel in the input layer has its own kernel, and a collection of these channels forms a filter. The weights of the kernels are not equal; some kernels are heavier than others.

3.2 Pooling layer

Pooling layers reduce the amount of parameters in the input by performing a reduction in dimensionality, it is sometimes referred to as downsampling. The pooling process, like the convolutional layer, spreads a filter through the whole input, but this filter does not contain any weights. Pooling may be classified into two types:

- Max pooling - maximizes the spatial size of a feature map while simultaneously giving translation invariance to the network. There are no parameters to modify which makes it a straightforward method and widely used [7]. Max pooling is used so that there's less computational load and reduces overfitting.
- Average pooling - computes the average value of pixels inside a filter's reception field as it scans the input, offering an alternate approach of reducing spatial dimensions by averaging values rather than picking the maximum. Essentially average pooling takes the average of all the pixel values.

3.3 Fully-connected layer

After the stack of convolutional layers and pooling layers, the Flattening Layer is applied. To identify the photos, the flattened matrix is passed through a fully connected layer. The flattening layer reduces the 3D picture vector to 1D. Since following the previously indicated layer stack, a last completely linked Dense layer is added. The dense layer is the fully-connected layer.

Each node in the output layer links directly to a node in the preceding layer in the fully-connected layer. This layer conducts categorization based on the characteristics retrieved by the preceding layers and their various filters. While convolutional and pooling layers often utilize ReLu functions to categorize inputs, Fully-connected layers typically use a softmax activation function to provide a probability ranging from 0 to 1.

4 Method

In the assignment, the following libraries were imported and utilized:

Library	**Description**
numpy	Multi-dimensional arrays and matrices
matplotlib	Data visualization
tensorflow	Building and training machine learning and deep learning models
keras	Building models
sklearn	Data analysis

Table 1: CNN libraries

The initial step in any Machine Learning or Deep Learning model is to pre-process the data that is being utilized. The keras library already consists of the CIFAR-10 dataset so it was first loaded using tensoflow.keras. Before the dataset was normalized it was reshaped

and explored, what this means is that the type of data, the number of training and testing data were explored, and many more. In order to make all the pixels the same, they should be normalized. When the data is explored, it was found that they range from 0 - 255, 255 is the maximum number of pixels, so in order for the values of the pixels to range from 0 - 1 they need to be normalized (divided by 255). This is done so that the model can be able to train fast (learn easier). Furthermore, the data was converted to binary using the to_categorical() function, this is for the machine to understand the type of data that it is presented with.

When the data was ready, it was time to build the Convolutional Neural Network (CNN) model. When building a Neural Network model, two APIs are commonly used: Sequential API and Functional API. The Sequential API enables us to build a model layer by layer and apply it to the Sequential Class. The disadvantage of Sequential API is that it cannot be used to develop models with many input sources and outputs at separate locations. We utilize Functional API to bypass this limitation. We may develop many input and output models by utilizing the Functional API. However, Sequential API is utilized in the majority of scenarios. For our CNN model, we will use the Sequential API.

Convolution or Conv2D was utilized to extract characteristics from an image. The Conv2D entails that the convolution occurs on 2 axes. The Con2D layers include the following parameters:

- *Filters* - the amount of filters that will be used in the convolution (32 or 64).
- *Kernel size* - specifies the length of the convolution window (3,3) or (4,4)
- *Padding* - is classified into two types: *(i) SAME and (ii) VALID*. There's no padding of zeros on the image's border in VALID padding. As a result, when convolution occurs, data is lost since some characteristics cannot be convolved. There is a layer of zeros padded on all picture boundaries in the SAME padding, therefore there is no data loss.
- *Activation* - to achieve a better result, we need to fit the model in ways that are too complicated, this is done by the activation function. It adds a non-linearity to the model. The activation function consists of four predominant functions (Sigmoid, TahH, ReLu, Softmax).

Sigmoid function is used for models that require us to anticipate the probability as an output that occurs between (0 and 1).

TanH (Tangent Hyperbolic) function is similar to logistic sigmoid, but better since it has a range of (-1 to 1).

ReLU (Rectified Linear Unit) Activation Function is currently the widely utilized activation function in the world. It has been employed in practically all convolutional neural networks and deep learning algorithms.

Softmax function is a more refined version of the Sigmoid function. It is used to classify data into many categories. The function computes the probability of a specific class. As a result, the function's output value range is 0 to 1. The fundamental distinction between the Sigmoid function and the Softmax function is that the former may be used for binary classification while the latter can be utilized for multi-class classification.

After adding the Conv2D layers, the size of the photos was scaled down using MaxPooling2D to keep the size of the photo (2,2) for the pooling layer. The model further converted the n-dimensional array to a 1-dimensional array. In the fully connected layer, all the neurons in the current layer are connected to the next layer, in this model, the first layer consists of 128 neurons and the second consists of 10 neurons. The compile) function was utilized to compile the model, it includes three parameters:

- *Loss function* - evaluate how well the model trains (the following might be selected depending on the dataset - 'Categorical cross entropy', 'Binary cross entropy', 'sparse categorical cross entropy')
- *Optimizer* - for changing the attributes of the neural network (learning rate and weights)
- *Metrics* - evaluating the performance of the model

The fit() function was utilized to train and test the model and the summary() function was used to see an overview of all the layers and parameters of the model. Which will be displayed below. When fitting the model, two arguments were utilized which are the ephocs and verbose. Ephocs entail the number of times the entire dataset is sent forward and backward through the neural network. In each epoch the number of parameters is adjusted so that the loss function can be minimized, when the number of epochs is a hyperparameter tuning when it is less, the model can underfit and when they are more the model can overfit. In the context of CNN, verbose entails the amount of output the training process generates. It is utilized as an argument when fitting the model

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 29, 29, 32)        1568

 max_pooling2d (MaxPooling2  (None, 14, 14, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 11, 11, 32)        16416

 max_pooling2d_1 (MaxPoolin  (None, 5, 5, 32)          0
 g2D)

 flatten (Flatten)           (None, 800)               0

 dense (Dense)               (None, 128)               102528

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 121802 (475.79 KB)
Trainable params: 121802 (475.79 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 3: Summary of the base model

5 Discussion and Results

5.1 Base Model

The base model was run with an epoch of 17, after running the model, it got a training performance of accuracy of **94.44%** and a loss function of **0.1545** which is good. The validation performance of the model is as follows: validation accuracy of approximately **65%** and a loss function of **2.410**. This shows that the model is overfitting. Meaning that it outputs good training results but poor validation (testing) results.

```
Epoch 17/17
1563/1563 [==============================] - 58s 37ms/step - loss: 0.1545 - accuracy: 0.9444 - val_loss: 2.4210 - val_accuracy: 0.6498
```

Figure 4: Base model

Below is a ROC curve for the base model for all the ten classes:

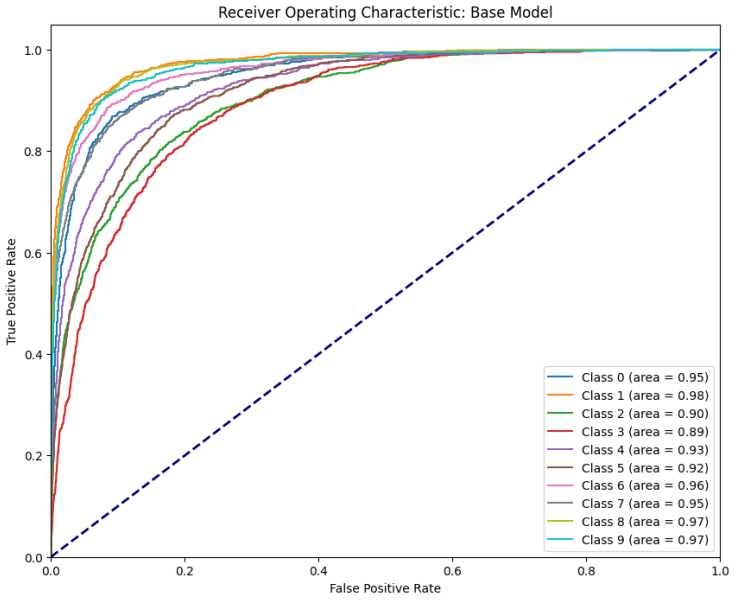

Figure 5: ROC: Base model

From the illustration above, the model demonstrates proficiency in predicting automobiles and is less accurate in predicting cats.

5.2 Model 1

To achieve better results, the model needs to be adjusted by adding dropout layers. This model will be dropped by **25%**. After running Model 1 twenty-five times by passing the dataset forward and backward in the model, the following observations were made: the validation accuracy increased from **65%** to **72%** and the loss function decreased from **2.4210** to **0.9133**, this means that the current model parameters are good at predicting the target variable.

```
Epoch 25/25
1563/1563 [==============================] - 100s 64ms/step - loss: 0.4861 - accuracy: 0.8290 - val_loss: 0.9133 - val_accuracy: 0.7180
```

Figure 6: Model 1 with a dropout of 25%

During data training, certain neurons are inhibited at random. The number supplied to neurons is the proportion of neurons that should be dropped throughout an iteration. Thus, after training, the weights of other neurons have little effect on the neurons.

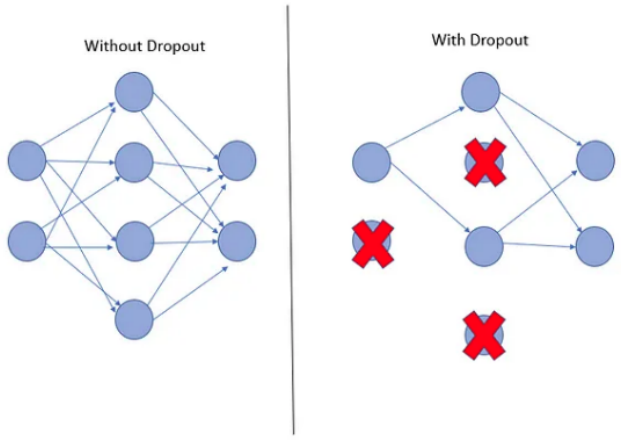

Figure 7: Droupout layer

Below is a ROC curve for Model 1 for all the ten classes:

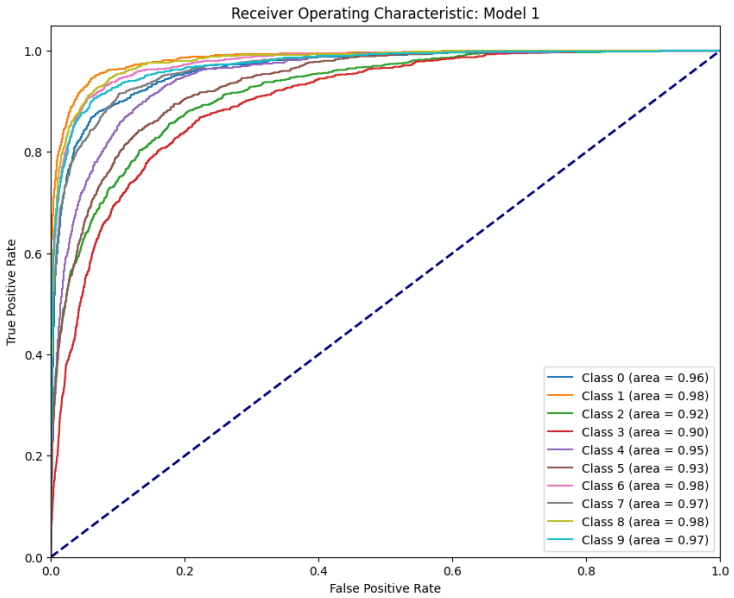

Figure 8: ROC: Base model

From the illustration above, this model still demonstrates proficiency in predicting automobiles and is less accurate in predicting dogs.

5.3 Model 2

The model can be further improved by adding more filter layers. Increasing the number of layers in a neural network can help it perform better. Increasing extra layers can assist the neural network in learning more complicated patterns in the input, resulting in improved predictions.

```
Epoch 50/50
1563/1563 [==============================] - 14s 9ms/step - loss: 0.1618 - accuracy: 0.9413 - val_loss: 0.7453 - val_accuracy: 0.8030
```

Figure 9: Model 2

After adding two more filter layers to the model, it has improved further by the following: the validation accuracy has improved from **72%** to **80%**, and the loss function had a decrease of **1,6757** from the base model. This shows that it has been reasonably trained.

This model has excellent training performance accuracy of **94%**, and an overall validation accuracy of the model is **80%**, we can deduce that the model has been reasonably trained.

Below is a ROC curve for Model 2 for all the ten classes:

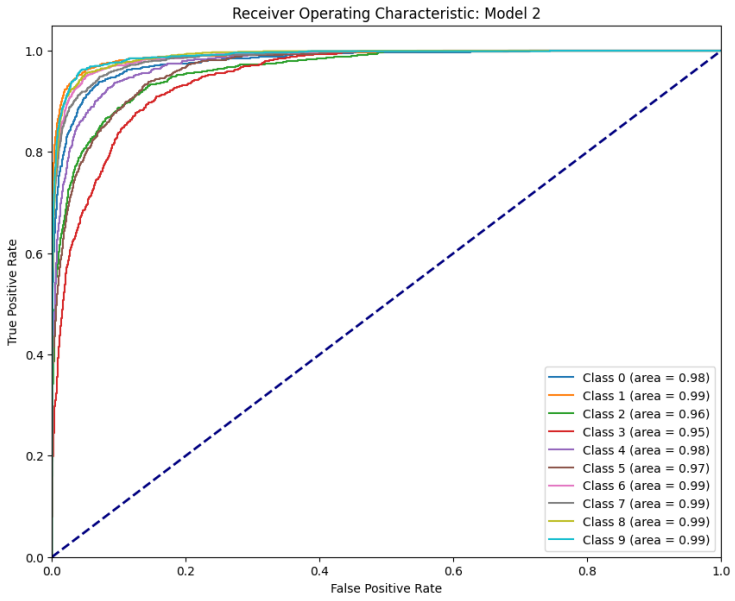

Figure 10: ROC: Model 2

In Model 2, the learning rate was employed in the "Adam" optimizer. The learning rate = 0.0001 input is supplied to the Adam optimizer, and it determines the optimizer's starting rate of learning. This value (0.0001) indicates that the optimizer will make relatively tiny modifications to the weights depending on the gradient of the loss function for each weight update. From the illustration above, this model demonstrates proficiency in predicting automobile, frog, horse, ship, and truck classes. However, its performance in predicting cats is less accurate.

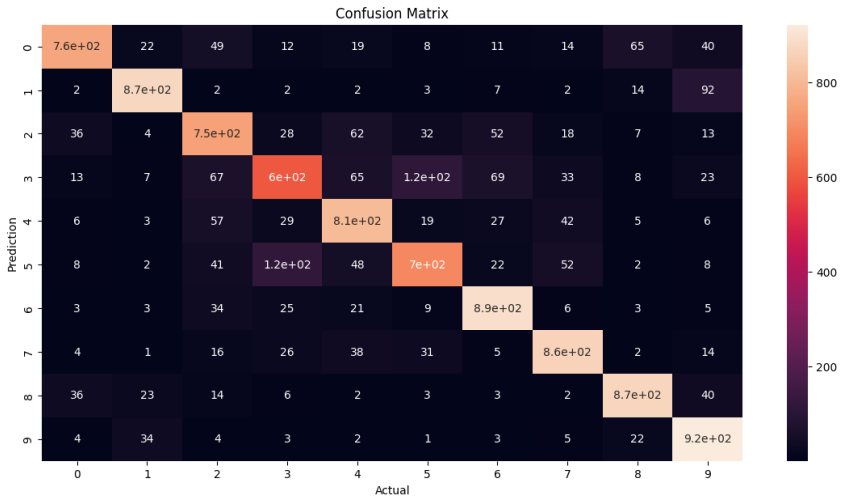

Figure 11: Confusion matrix

Above is the Illustration of a confusion matrix of Model 3. A confusion matrix is a table that summarizes a classification model's performance by comparing predicted labels against true labels. It shows the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) predicted by the model.

A confusion matrix visualizes the frequency of each of them in one location. This allows you to keep track of the amount of right predictions and mistakes in each category at the same time.

	precision	recall	f1-score	support
0	0.87	0.76	0.81	1000
1	0.90	0.87	0.89	1000
2	0.72	0.75	0.74	1000
3	0.70	0.60	0.65	1000
4	0.76	0.81	0.78	1000
5	0.76	0.69	0.73	1000
6	0.82	0.89	0.85	1000
7	0.83	0.86	0.85	1000
8	0.87	0.87	0.87	1000
9	0.79	0.92	0.85	1000
accuracy			0.80	10000
macro avg	0.80	0.80	0.80	10000
weighted avg	0.80	0.80	0.80	10000

Figure 12: Classification report

The classification report above is meant for a clearer overview of the results obtained by Model 3. These results are for the validation accuracy, this shows that the validation accuracy shows that the validation model is proficient at predicting automobiles.

6 Conclusion

In this assignment, the CNN model was employed in the CIFAR-10 dataset. CNN is one of the best models for predicting images and can be utilized in other several uses. Three different CNN models were employed using different epochs, dropouts, and many more. The base model was good in predicting but the only problem was a higher loss function.

The base model was adjusted into Model 1 by adding a 25% dropout and it resulted in a lower loss function but the validation accuracy was not sufficient. In the last model (Model 2), the learning rate was employed and the model performed well. One can conclude that the model has been reasonably employed in terms of the loss function, training accuracy, and validation accuracy.

References

[1] IBM. Convolutional neural networks, 2023. Accessed: 22 October 2023.

[2] A. Krizhevsky. The cifar-10 dataset. `https://www.cs.toronto.edu/~kriz/cifar.html`, 2009.

[3] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[4] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 2021.

[5] J. Park, H. Lee, C. Y. Park, S. Hasan, T.-Y. Heo, and W. H. Lee. Algal morphological identification in watersheds for drinking water supply using neural architecture search for convolutional neural network. *Water*, 11(7):1338, 2019.

[6] A. W. Trask. *Grokking deep learning*. Simon and Schuster, 2019.

[7] A. Zafar, M. Aamir, N. Mohd Nawi, A. Arshad, S. Riaz, A. Alruban, A. K. Dutta, and S. Almotairi. A comparison of pooling methods for convolutional neural networks. *Applied Sciences*, 12(17):8643, 2022.