

TRANSFER LEARNING FOR IMAGE CLASSIFICATION

ANTONY ROY NELLANGARA

























































Stud ID : 22019057

Machine Learning and Neural Networks

Colab link :

https://colab.research.google.com/drive/1TmNe8D6XjY7EBuf9hgL_rC40YgGERCHI?usp=sharing

Introduction to the CIFAR10 dataset

- The CIFAR10 dataset has been chosen to implement image classification using different techniques and is a very famous dataset for Machine Learning Algorithms and Computer Vision.
 - The CIFAR10 dataset is huge with 60000 images with 32*32 dimensions that can be classified into 10 different classes with each class containing 6000 images.
 - CIFAR10 dataset's diversity and the vast amount of distinctive images makes it a perfect choice to implement image classification.
 - The dataset is split into 50000 images for training and 10000 images for testing.
- | | | | | | | | |
|------------|---|---|---|---|---|---|---|
| airplane |  |  |  |  |  |  |  |
| automobile |  |  |  |  |  |  |  |
| bird |  |  |  |  |  |  |  |
| cat |  |  |  |  |  |  |  |
| deer |  |  |  |  |  |  |  |
| dog |  |  |  |  |  |  |  |
| frog |  |  |  |  |  |  |  |
| horse |  |  |  |  |  |  |  |

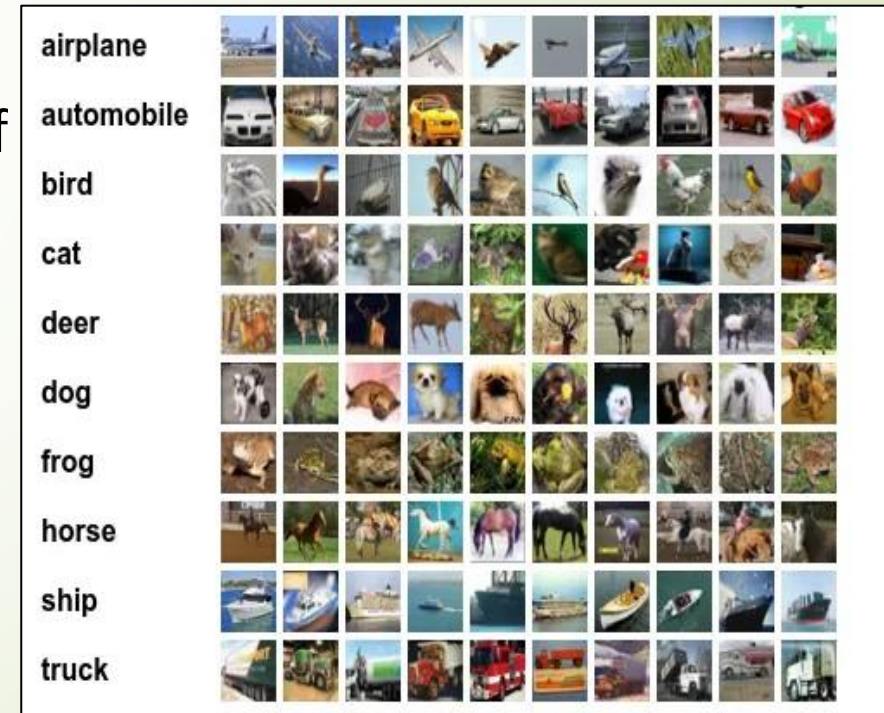


Fig 1 : CIFAR10 dataset

Image classifn. from scratch using CNN and ANN

- Image classification of the CIFAR10 dataset was initially done using the ANN (Artificial Neural Networks) Model and CNN (Convolutional Neural Networks) Model to study the performance of the Models.
- ANN does not specialize in image classification. and is a more general neural network for various other tasks and performs poorly.
- CNN classifier for image classification. makes use of multiple specialized convolutional layers and pooling layers and performs far more efficiently than the ANN model. Fine tuning the CNN model by data augmentation, increasing the number of convolution and pooling layers, modifying the number of filters and the number of output neurons were done to achieve a higher accuracy.

```
#Performing the data augmentation for CNN model
data_augmentation = keras.Sequential([
    layers.experimental.preprocessing.RandomFlip("horizontal", input_shape = (32,32,3)),
    layers.experimental.preprocessing.RandomRotation(0.1),
    layers.experimental.preprocessing.RandomZoom(0.1)
])
```

```
#The first set of convolution and pooling layer.
layers.Conv2D(filters = 64, padding = 'same', input_shape = (32,32,3), activation = 'relu', kernel_size = (3,3)),
layers.MaxPooling2D((2,2)),

#The second set of convolution and pooling layer
layers.Conv2D(filters = 128, padding = 'same', activation = 'relu', kernel_size = (3,3)),
layers.MaxPooling2D((2,2)),
```

```
#Adding a drop layer to drop 20% of the neurons
layers.Dropout(0.2),
layers.Flatten(),
layers.Dense(200, activation = 'relu'),
layers.Dense(10, activation = 'softmax')
```

```
#Compiling the CNN model
cnn.compile(optimizer = 'adam',
            loss = 'sparse_categorical_crossentropy',
            metrics = ['accuracy'])
```

Fig 2 : CNN layers

Comparison between ANN and CNN

ANN

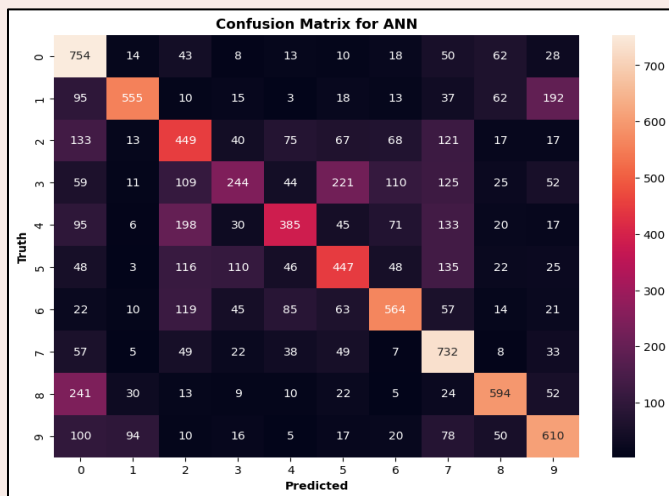
Accuracy : 51%

Classification Report:

Classification Report for ANN model :				
	precision	recall	f1-score	support
0	0.47	0.75	0.58	1000
1	0.75	0.56	0.64	1000
2	0.40	0.45	0.42	1000
3	0.45	0.24	0.32	1000
4	0.55	0.39	0.45	1000
5	0.47	0.45	0.46	1000
6	0.61	0.56	0.59	1000
7	0.49	0.73	0.59	1000
8	0.68	0.59	0.63	1000
9	0.58	0.61	0.60	1000
accuracy			0.53	10000
macro avg	0.55	0.53	0.53	10000
weighted avg	0.55	0.53	0.53	10000

3 Dense layers were created in the model and the SGD optimizer was used. After 15 epochs, there is no significant increase in the accuracy.

Confusion Matrix:



CNN

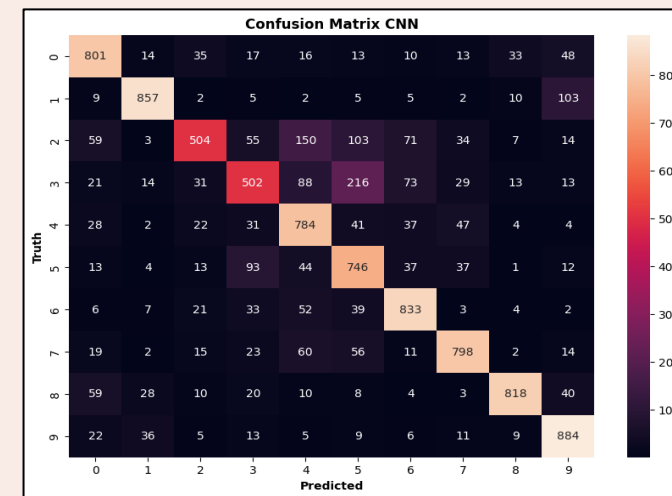
Accuracy : 76%

Classification Report:

Classification Report :				
	precision	recall	f1-score	support
0	0.78	0.79	0.78	1000
1	0.83	0.90	0.86	1000
2	0.65	0.71	0.68	1000
3	0.62	0.51	0.56	1000
4	0.79	0.63	0.70	1000
5	0.68	0.64	0.66	1000
6	0.75	0.83	0.79	1000
7	0.81	0.82	0.81	1000
8	0.86	0.85	0.85	1000
9	0.78	0.88	0.83	1000
accuracy			0.76	10000
macro avg	0.75	0.76	0.75	10000
weighted avg	0.75	0.76	0.75	10000

Data augmentation, 2 sets of convolution and pooling layers, a drop out layer and dense layers and 'adam' optimizer was used.

Confusion Matrix:



Transfer learning

- Transfer learning is a technique used in Machine Learning where a pre trained model that has been developed and trained for a specific task is reused as the starting point for another task.
- While training a Neural Network from scratch, it can be computationally expensive as it takes a lot of time to train a complex Neural Network for a large dataset even with high performing computers. Transfer learning saves time and resources by utilising the knowledge the pre trained model gained from its training.
- Transfer learning utilizes the feature learning layers of the pre trained model to classify a task for another dataset than one it was created for.
- Transfer learning allows both adaptability and generalization as it quickly adapts to the new data just by fine tuning the parameters.
- VGG (VGG16 OR VGG19), GoogLeNet (Inception), Residual Network (ResNet50) are some famous transfer learning models.

Transfer learning model – VGG16

- VGG16 is a convolutional neural network that is one of the best computer vision models to date and was trained on the ImageNet dataset which contains more than 14 million high resolution images belonging to 1000 classes.
- The 16 in VGG16 represents the 16 layers that have weights and this include 13 convolutional layers , 5 max pooling layers and 3 dense layers which sums up to 21 layers out of which 16 weight layers are learnable.
- VGG16 has 224,224 with 3 RGB channel as input tensor. The convolution layers have 3*3 filter with stride 1 and has maxpool layer of 2*2 filter with stride 2.

VGG16 Architecture

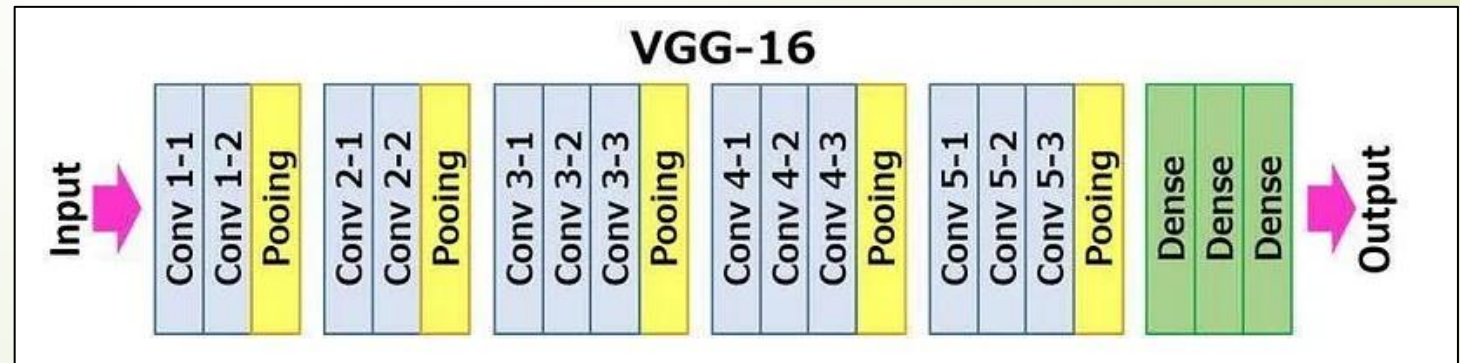
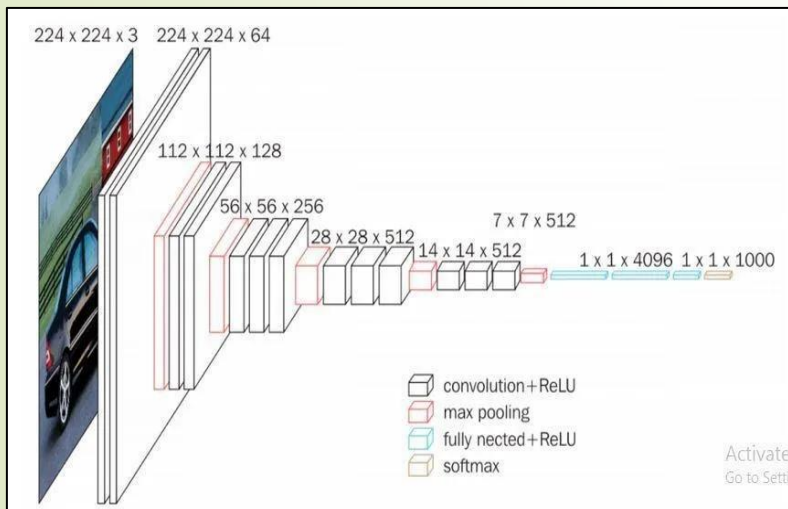


Fig 3 : VGG16 Architecture

Fine tuning steps in VGG16

- The base model from VGG16 architecture has been extracted excluding the top layer of the model and the last layer of the base model is set to be 'block3_pool'.
- On top of the 'block3_pool' layer, the following classification and dense layers were stacked to fine tune the model for the CIFAR10 dataset :
 - *GlobalAveragePooling2D()* : calculates the average value of each feature map in the input tensor and outputs a tensor that is smaller in size.
 - *BatchNormalization()* : normalizes the intermediate outputs of each layer within a batch during training.
 - Three dense layers were implemented with 400, 256 and 256 neurons respectively with 'relu' activation function.
 - A Dropout layer where 20% of neurons will be dropped will be stacked on the above layers and will be placed below the final dense layers with 10 output classes for the CIFAR10 dataset.
- The base layers of the VGG16 layers were frozen and the parameters of the hidden layers were not trained. Only the parameters of the final layers were trained and was compiled using the 'adam' optimizer and 'BinaryCrossentropy' as the loss function for 50 epochs. Early stopping was implement for effective regularization to avoid overfitting and best weights from the epochs were restored.
- All the fine tuning processes were carried out after several trial and error methods by considering the accuracy score, performance and efficiency of the model.

Results of VGG16

- The performance of VGG16 was very good with a high training accuracy of 96% in just 50 epochs.

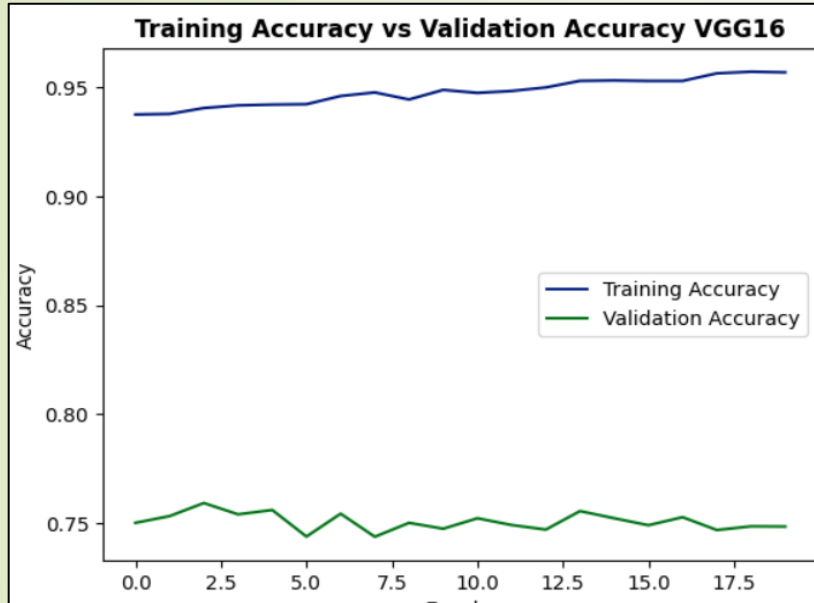


Fig 6 : Training vs Validation (Accuracy)

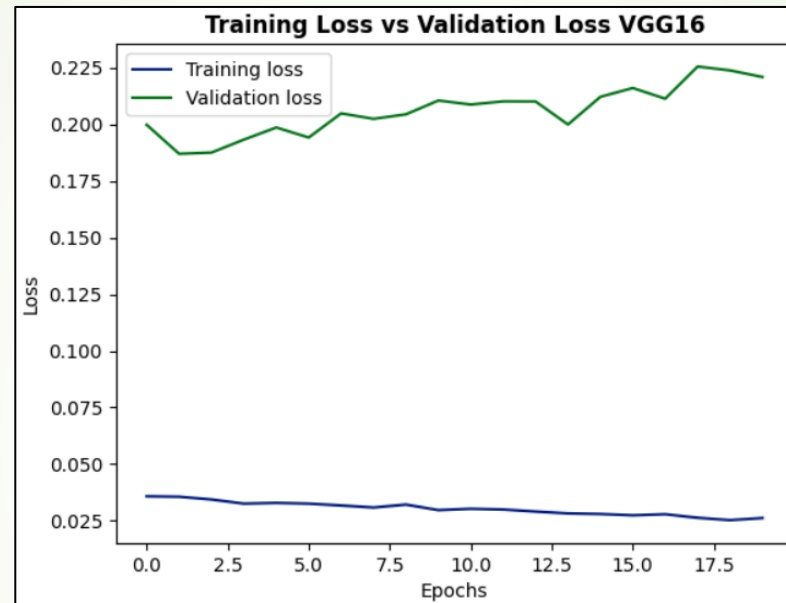


Fig 6 : Training vs Validation (Loss)

- The training accuracy was 96% and the testing accuracy was 99%. The training loss was 0.0211 and testing loss was 0.0050.
- The classification report shows the results of 10 classes and their metrics. Confusion matrix represents the performance of 10 classes.
- The training accuracy improves over iterations while validation accuracy remains the same. Training loss improves over iterations while validation loss decreases.

Classification Report for VGG16 transfer learning model :				
	precision	recall	f1-score	support
0	0.98	1.00	0.99	5000
1	1.00	1.00	1.00	5000
2	1.00	0.97	0.98	5000
3	0.98	0.99	0.99	5000
4	0.99	0.99	0.99	5000
5	0.99	1.00	0.99	5000
6	0.99	0.99	0.99	5000
7	1.00	1.00	1.00	5000
8	1.00	0.99	1.00	5000
9	1.00	1.00	1.00	5000
accuracy			0.99	50000
macro avg	0.99	0.99	0.99	50000
weighted avg	0.99	0.99	0.99	50000

Fig 4 : Classfn. Report

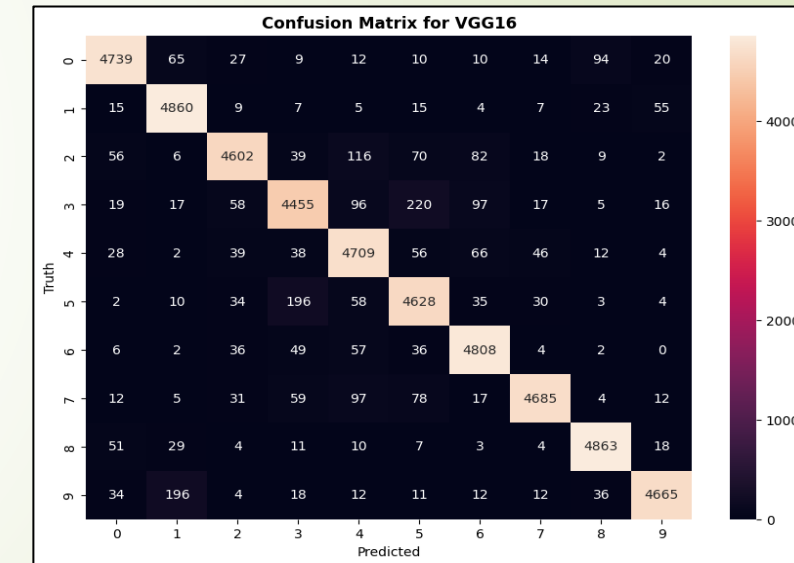


Fig 5 : Confusion Matrix

Conclusion

- VGG16 transfer learning model clearly outperforms the CNN model and ANN model for image classification.
- VGG16 model easily gave a high accuracy after some fine tuning which the CNN model could not achieve even after multiple epochs and implementing multiple layers and fine tuning techniques.

Factors	CNN	VGG16
Training Accuracy	82%	96%
Testing Accuracy	77%	99%
Training Loss	0.5024	0.0211
Testing Loss	0.7772	0.0050

Table 1 : Comparison. Between CNN and VGG16

- The VGG16 model has a higher accuracy and lower loss rate in both training and testing.
- CNN model was trained for 15 epochs and the VGG16 model was trained for 50 epochs to achieve the same.

Limitations & Potential Areas of improvement

Limitations :

- One of the major drawbacks of the VGG16 model is that it is very large in size and this can lead to higher running times to train the parameters. This makes this particular model more time consuming than other transfer learning architectures. Since the size is large and the performance is better for larger datasets, when used for smaller datasets, this might lead to poor performance.
- VGG16 is also a very deep complex model and this leads to the usage of higher computational resources to train the model for better results.
- Other transfer learning models which are smaller in size also has similar performance to VGG16 model.

Areas of improvement :

- Improved data augmentation and hyper tuning techniques can be done to bring down the number of iterations and the training time of the model and increase the performance.
- Implementing different regularization methods like L2 regularization can be done in order to prevent overfitting of the model.
- Tuning the batch size and the learning rate can also result in better performance of the model.