Introduction to Machine Learning

Assignment 2 report

By:

Name : Nour Mohamed Mahmoud

ID : 7591

Introduction

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

Here are the classes in the dataset, as well as 10 random images from each:

Neural Network Image Classifiers

Architecture of ANN

We utilize an Artificial Neural Network (ANN) with three hidden layers:

- First hidden layer: 4096 units
- Second hidden layer: 2048 units
- Third hidden layer: 512 units

#Note

Batch normalization : is applied after each layer to ensure stable and efficient training.

Activation function: we used (reLU) as an activation function in the forward pass .

Data Loaders: Data loaders are created for training and test sets, using a batch size of 100.

Model Initialization: The ANN model is instantiated with an input size of 32x32x3 and output size of 10 classes.

Loss Function and Optimizer: Cross-Entropy loss is used for multi-class classification. The SGD optimizer is employed with a learning rate of 0.001 and momentum of 0.9.

ANN Design

-With the previous specifications the ANN design looks like this:

```python
# Step 1: Build ANN
class ANN(nn.Module):
    def __init__(self, input_size, num_classes):
        super(ANN, self).__init__()
        self.fc1 = nn.Linear(input_size, 4096)
        self.fc2 = nn.Linear(4096, 2048)
        self.fc3 = nn.Linear(2048, 512)
        self.fc4 = nn.Linear(512, num_classes)

    def forward(self, x):
        x = x.view(-1, 32 * 32 * 3)  # Flatten the input images
        x = torch.relu(self.fc1(x))
        x = torch.relu(self.fc2(x))
        x = torch.relu(self.fc3(x))
        x = self.fc4(x)
        return x
```

ANN Train

-In the ANN training stage among 10 epochs , the accuracy and loss every 200 batches was printed in the next figure.
-The loss calculated per batch was not smoothly but overall it does decrease well with training until it reached 0.65 on the last epoch.
- The train accuracy was increasing until it reached 77% on the last epoch.

-The total train accuracy was 82.1% , here is some figures about the performance of the ANN training stage and the overall accuracy.

```
# Training ANN model
training(ann_model, ann_lossFunction, ann_optimizer)
```

```
Epoch [1/10], Batch [200/500], Loss: 1.9799, Accuracy: 0.29
Epoch [1/10], Batch [400/500], Loss: 1.6659, Accuracy: 0.41
Epoch [2/10], Batch [200/500], Loss: 1.4740, Accuracy: 0.48
Epoch [2/10], Batch [400/500], Loss: 1.4438, Accuracy: 0.49
Epoch [3/10], Batch [200/500], Loss: 1.3160, Accuracy: 0.54
Epoch [3/10], Batch [400/500], Loss: 1.3066, Accuracy: 0.54
Epoch [4/10], Batch [200/500], Loss: 1.1915, Accuracy: 0.58
Epoch [4/10], Batch [400/500], Loss: 1.1920, Accuracy: 0.58
Epoch [5/10], Batch [200/500], Loss: 1.0811, Accuracy: 0.62
Epoch [5/10], Batch [400/500], Loss: 1.1098, Accuracy: 0.61
Epoch [6/10], Batch [200/500], Loss: 0.9843, Accuracy: 0.65
Epoch [6/10], Batch [400/500], Loss: 1.0089, Accuracy: 0.65
Epoch [7/10], Batch [200/500], Loss: 0.9040, Accuracy: 0.68
Epoch [7/10], Batch [400/500], Loss: 0.9113, Accuracy: 0.67
Epoch [8/10], Batch [200/500], Loss: 0.7908, Accuracy: 0.72
Epoch [8/10], Batch [400/500], Loss: 0.8385, Accuracy: 0.70
Epoch [9/10], Batch [200/500], Loss: 0.7031, Accuracy: 0.75
Epoch [9/10], Batch [400/500], Loss: 0.7440, Accuracy: 0.74
Epoch [10/10], Batch [200/500], Loss: 0.6074, Accuracy: 0.79
Epoch [10/10], Batch [400/500], Loss: 0.6579, Accuracy: 0.77
```

The loss and accuracy in each 200 batch in the training stage.

```
# train accuracy
eval(ann_model, train_dl)
```

```
train accuracy : 0.821
```

The overall accuracy in the training stage

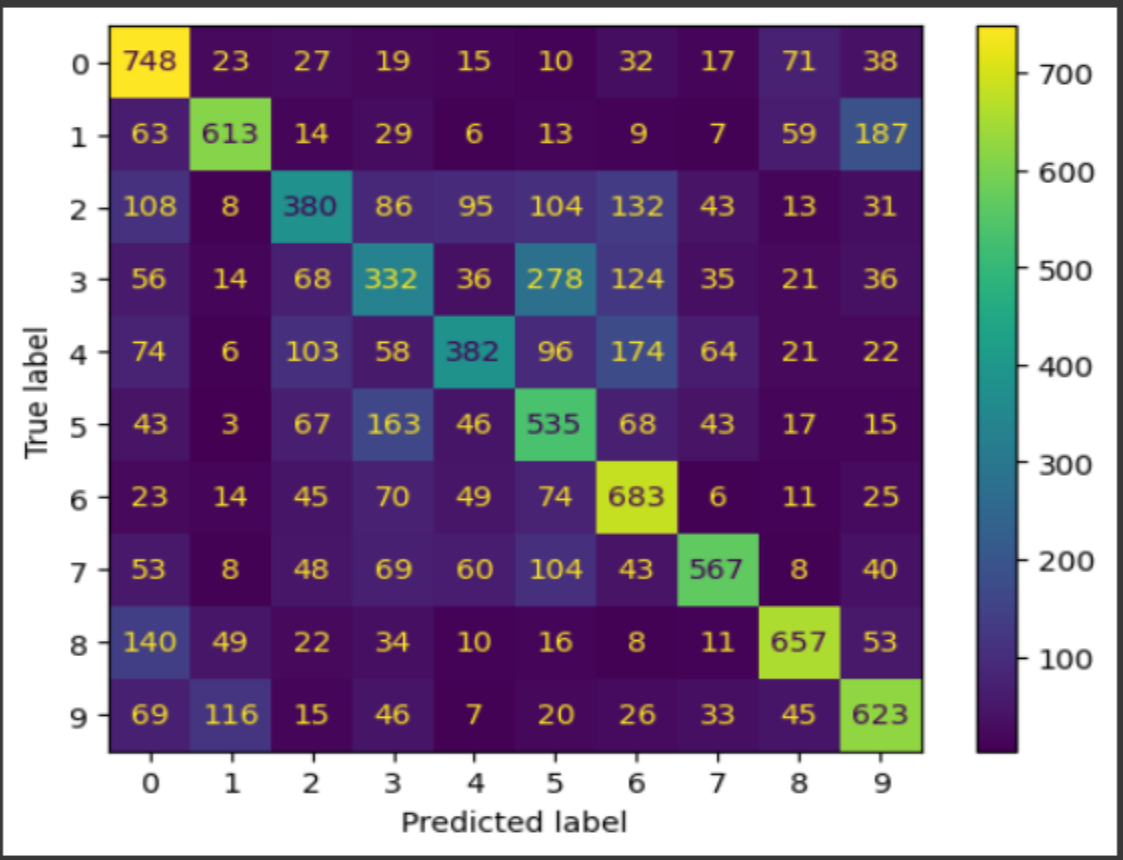
ANN Test

1- Performance measures

On the ANN test stage, we obtain some performance measures like test accuracy, precision, recall and the confusion matrix in the following figure.

```
#test ANN
eval(ann_model, test_dl)
```

```
Results for testing:
Accuracy: 0.552
Precision: 0.558
Recall: 0.552
```



2-Test images for ANN model

- In the following figure we passed the test data loader throw the network and printed some test images , here some of the printed plots:

Conclusion: the test accuracy was 55% , so we can say that it was not complex enough to capture the patterns in our data.

Transfer learning models

Models Used:

We employ three pretrained Convolutional Neural Networks (CNNs) for transfer learning:

- VGG16
- GoogleNet
- ResNet50

Transfer Learning Approach:

Transfer learning involves initializing the CNNs with weights pretrained on large datasets and fine-tuning them on CIFAR-10. This process leverages the prelearned features of the models.

Evaluation approach:

The transfer learning models are evaluated on the CIFAR-10 test set using the same metrics as the ANN.

1- VGG16:

Model Initialization: The VGG16 model is instantiated with output size of 10 classes and the pretrained = True.

Loss Function and Optimizer: Cross-Entropy loss is used for multi-class classification. The SGD optimizer is employed with a learning rate of 0.001 and momentum of 0.9.

- **VGG16 train:**

-In the VGG16 training stage among 10 epochs , the accuracy and loss every 200 batches was printed in the next figure.

-The loss calculated per batch decreased well with training until it reached 0.04 on the last epoch.

- The train accuracy was increasing until it reached 98% on the last epoch.

-The total train accuracy was 99.3% , here is some figures about the performance of the VGG16 training stage and the overall accuracy.

```
# Fine-tune the VGG16 model
training(vgg16_model, vgg16_lossFunction, vgg16_optimizer)
```

```
Epoch [1/10], Batch [200/500], Loss: 0.9976, Accuracy: 0.66
Epoch [1/10], Batch [400/500], Loss: 0.6262, Accuracy: 0.80
Epoch [2/10], Batch [200/500], Loss: 0.4164, Accuracy: 0.86
Epoch [2/10], Batch [400/500], Loss: 0.4297, Accuracy: 0.86
Epoch [3/10], Batch [200/500], Loss: 0.2698, Accuracy: 0.91
Epoch [3/10], Batch [400/500], Loss: 0.2891, Accuracy: 0.90
Epoch [4/10], Batch [200/500], Loss: 0.1799, Accuracy: 0.94
Epoch [4/10], Batch [400/500], Loss: 0.2029, Accuracy: 0.93
Epoch [5/10], Batch [200/500], Loss: 0.1406, Accuracy: 0.95
Epoch [5/10], Batch [400/500], Loss: 0.1421, Accuracy: 0.95
Epoch [6/10], Batch [200/500], Loss: 0.1034, Accuracy: 0.97
Epoch [6/10], Batch [400/500], Loss: 0.1166, Accuracy: 0.96
Epoch [7/10], Batch [200/500], Loss: 0.0839, Accuracy: 0.97
Epoch [7/10], Batch [400/500], Loss: 0.0914, Accuracy: 0.97
Epoch [8/10], Batch [200/500], Loss: 0.0561, Accuracy: 0.98
Epoch [8/10], Batch [400/500], Loss: 0.0735, Accuracy: 0.98
Epoch [9/10], Batch [200/500], Loss: 0.0529, Accuracy: 0.98
Epoch [9/10], Batch [400/500], Loss: 0.0699, Accuracy: 0.98
Epoch [10/10], Batch [200/500], Loss: 0.0522, Accuracy: 0.98
Epoch [10/10], Batch [400/500], Loss: 0.0486, Accuracy: 0.98
```

The overall accuracy in the training stage

```
[14] # Train accuracy
     eval(vgg16_model, train_dl)
```

```
train accuracy : 0.993
```

The overall accuracy in the training stage

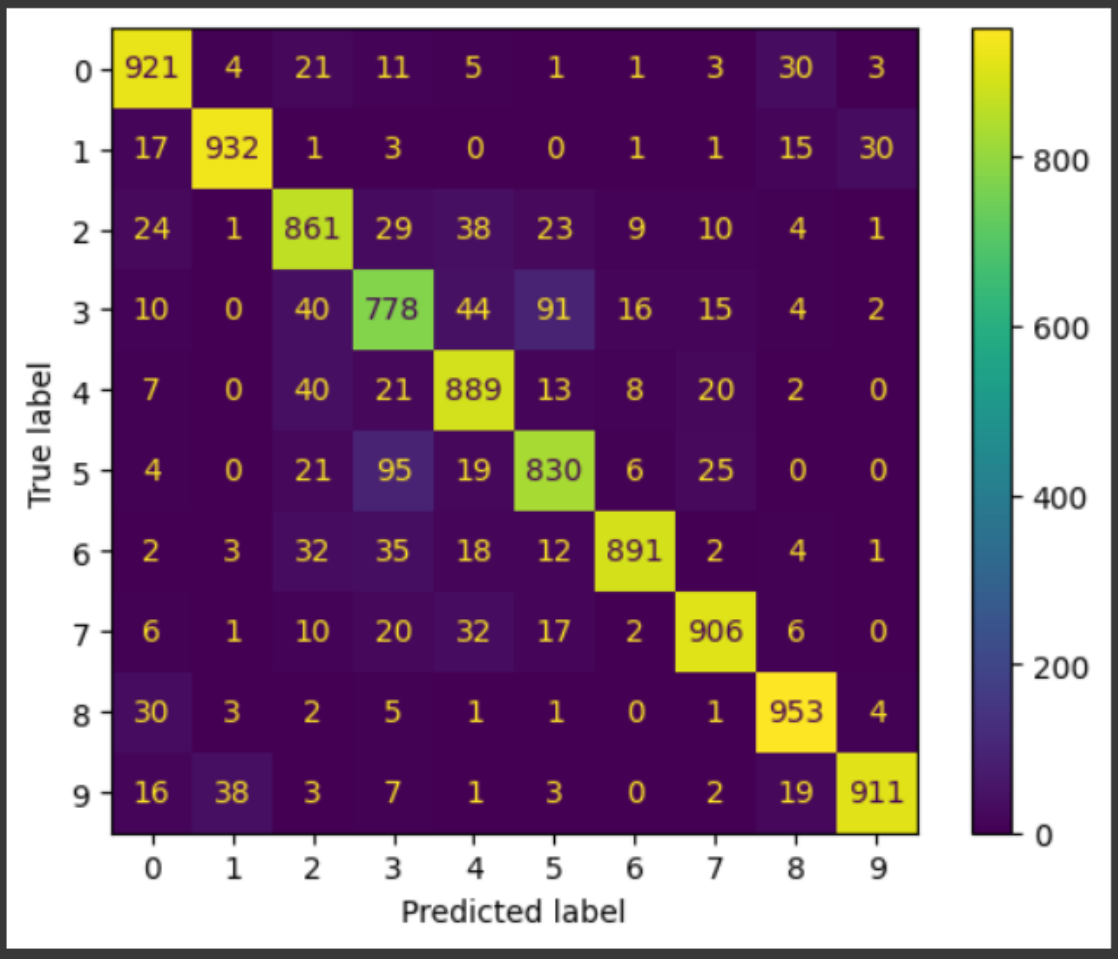
- ***VGG16 Test:***

1- Performance measures

On the VGG16 test stage, we obtain some performance measures like test accuracy, precision, recall and the confusion matrix in the following figure.

```
#Test vgg16_model
eval(vgg16_model, test_dl)
```

```
Results for testing:
Accuracy: 0.887
Precision: 0.888
Recall: 0.887
```



2-Test images for VGG16 model

- In the following figure we passed the test data loader throw the network and printed some test images , here some of the printed plots:

Conclusion: the test accuracy was 88.7%, the overall performance was much better than the ANN model.

3-GoogleNet:

Model Initialization: The GoogleNet model is instantiated with output size of 10 classes and the pretrained = True.
Loss Function and Optimizer: Cross-Entropy loss is used for multi-class classification. The SGD optimizer is employed with a learning rate of 0.001 and momentum of 0.9.

- **GoogleNet train:**

-In the GoogleNet training stage among 10 epochs , the accuracy and loss every 200 batches was printed in the next figure.

-The loss calculated per batch decreased well with training until it reached 0.114 on the last epoch.

- The train accuracy was increasing until it reached 96% on the last epoch.

-The total train accuracy was 97.5% , here is some figures about the performance of the GoogleNet training stage and the overall accuracy.

```
# Training googlenet_model
training(googlenet_model, googlenet_model_lossFunction, googlenet_model_optimizer)
```

```
Epoch [1/10], Batch [200/500], Loss: 1.3610, Accuracy: 0.52
Epoch [1/10], Batch [400/500], Loss: 0.8965, Accuracy: 0.69
Epoch [2/10], Batch [200/500], Loss: 0.6304, Accuracy: 0.78
Epoch [2/10], Batch [400/500], Loss: 0.6130, Accuracy: 0.79
Epoch [3/10], Batch [200/500], Loss: 0.4228, Accuracy: 0.85
Epoch [3/10], Batch [400/500], Loss: 0.4577, Accuracy: 0.84
Epoch [4/10], Batch [200/500], Loss: 0.3320, Accuracy: 0.89
Epoch [4/10], Batch [400/500], Loss: 0.3557, Accuracy: 0.88
Epoch [5/10], Batch [200/500], Loss: 0.2574, Accuracy: 0.91
Epoch [5/10], Batch [400/500], Loss: 0.2897, Accuracy: 0.90
Epoch [6/10], Batch [200/500], Loss: 0.1942, Accuracy: 0.93
Epoch [6/10], Batch [400/500], Loss: 0.2369, Accuracy: 0.92
Epoch [7/10], Batch [200/500], Loss: 0.1581, Accuracy: 0.94
Epoch [7/10], Batch [400/500], Loss: 0.1974, Accuracy: 0.93
Epoch [8/10], Batch [200/500], Loss: 0.1394, Accuracy: 0.95
Epoch [8/10], Batch [400/500], Loss: 0.1536, Accuracy: 0.95
Epoch [9/10], Batch [200/500], Loss: 0.1174, Accuracy: 0.96
Epoch [9/10], Batch [400/500], Loss: 0.1308, Accuracy: 0.95
Epoch [10/10], Batch [200/500], Loss: 0.0944, Accuracy: 0.97
Epoch [10/10], Batch [400/500], Loss: 0.1144, Accuracy: 0.96
```

The loss and accuracy in each 200 batch in the training stage.

```
[18] # Train accuracy
     eval(googlenet_model, train_dl)
```

```
train accuracy : 0.975
```

The overall accuracy in the training stage

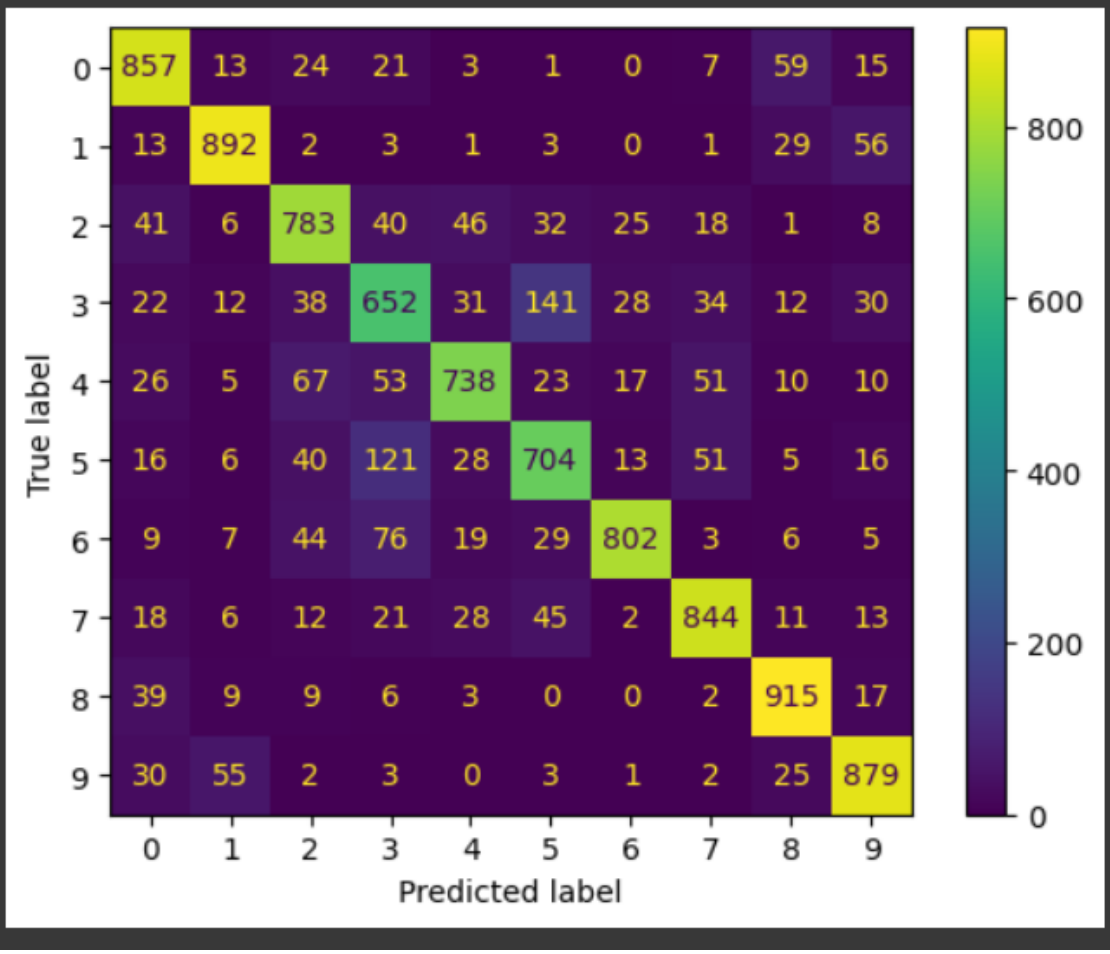
- ***GoogleNet Test:***

1-Performance measures

On the GoogleNet test stage, we obtain some performance measures like test accuracy, precision, recall and the confusion matrix in the following figure.

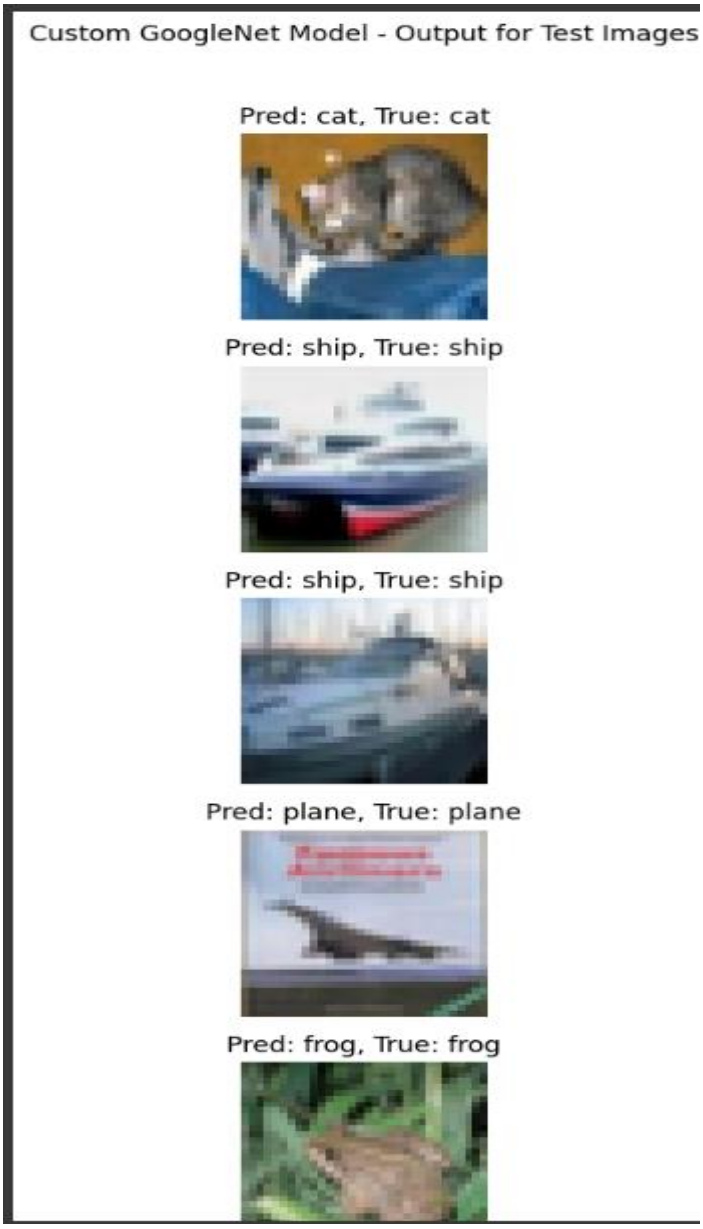
```
#Test googlenet_model
eval(googlenet_model, test_dl)
```

```
Results for testing:
Accuracy: 0.807
Precision: 0.807
Recall: 0.807
```



2-Test images for GoogleNet model

- In the following figure we passed the test data loader throw the network and printed some test images , here some of the printed plots:



Conclusion: the test accuracy was 80.7%, the overall performance was less than the VGG16, but much better than the ANN model.

4-ResNet50:

Model Initialization: The ResNet50 model is instantiated with output size of 10 classes and the pretrained = True.
Loss Function and Optimizer: Cross-Entropy loss is used for multi-class classification. The SGD optimizer is employed with a learning rate of 0.001 and momentum of 0.9.

- **ResNet50 train:**

-In the ResNet50 training stage among 10 epochs , the accuracy and loss every 200 batches was printed in the next figure.

-The loss calculated per batch decreased well with training until it reached 0.062 on the last epoch.

- The train accuracy was increasing until it reached 98% on the last epoch.

-The total train accuracy was 98.1% , here is some figures about the performance of the ResNet50 training stage and the overall accuracy.

```
# Training resnet_model
training(resnet_model, resnet_model_lossFunction, resnet_model_optimizer)
```

```
Epoch [1/10], Batch [200/500], Loss: 1.1595, Accuracy: 0.61
Epoch [1/10], Batch [400/500], Loss: 0.6940, Accuracy: 0.76
Epoch [2/10], Batch [200/500], Loss: 0.5297, Accuracy: 0.82
Epoch [2/10], Batch [400/500], Loss: 0.8314, Accuracy: 0.73
Epoch [3/10], Batch [200/500], Loss: 0.4213, Accuracy: 0.85
Epoch [3/10], Batch [400/500], Loss: 0.4272, Accuracy: 0.85
Epoch [4/10], Batch [200/500], Loss: 0.2490, Accuracy: 0.91
Epoch [4/10], Batch [400/500], Loss: 0.2656, Accuracy: 0.91
Epoch [5/10], Batch [200/500], Loss: 0.1601, Accuracy: 0.94
Epoch [5/10], Batch [400/500], Loss: 0.2054, Accuracy: 0.93
Epoch [6/10], Batch [200/500], Loss: 0.1127, Accuracy: 0.96
Epoch [6/10], Batch [400/500], Loss: 0.1406, Accuracy: 0.95
Epoch [7/10], Batch [200/500], Loss: 0.0765, Accuracy: 0.97
Epoch [7/10], Batch [400/500], Loss: 0.1132, Accuracy: 0.96
Epoch [8/10], Batch [200/500], Loss: 0.0680, Accuracy: 0.98
Epoch [8/10], Batch [400/500], Loss: 0.0902, Accuracy: 0.97
Epoch [9/10], Batch [200/500], Loss: 0.0721, Accuracy: 0.98
Epoch [9/10], Batch [400/500], Loss: 0.0760, Accuracy: 0.97
Epoch [10/10], Batch [200/500], Loss: 0.0499, Accuracy: 0.98
Epoch [10/10], Batch [400/500], Loss: 0.0620, Accuracy: 0.98
```

The loss and accuracy in each 200 batch in the training stage.

```
[22] # Train accuracy
     eval(resnet_model, train_dl)
```

```
train accuracy : 0.981
```

The overall accuracy in the training stage.

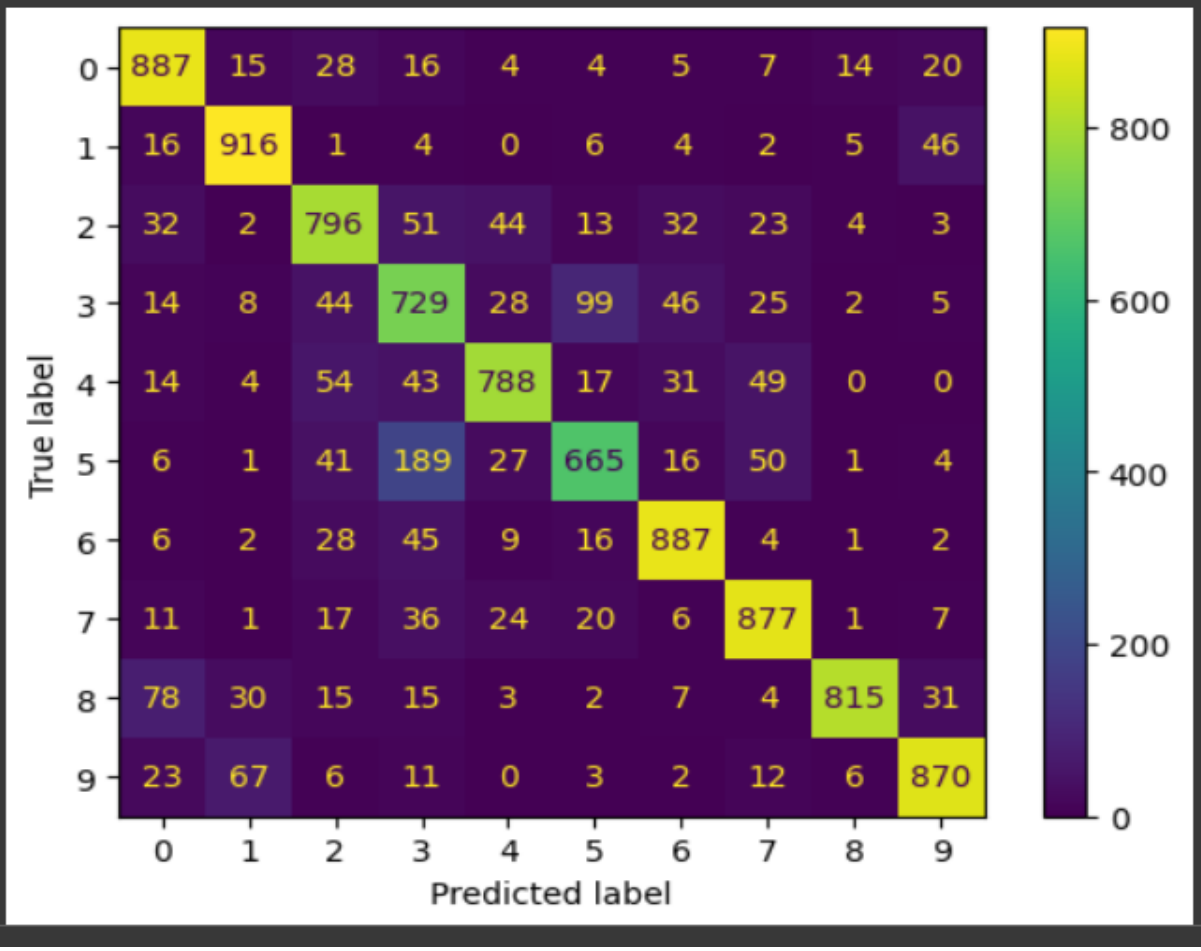
ResNet50 Test:

1-Performance measures

On the ResNet50 test stage, we obtain some performance measures like test accuracy, precision, recall and the confusion matrix in the following figure.

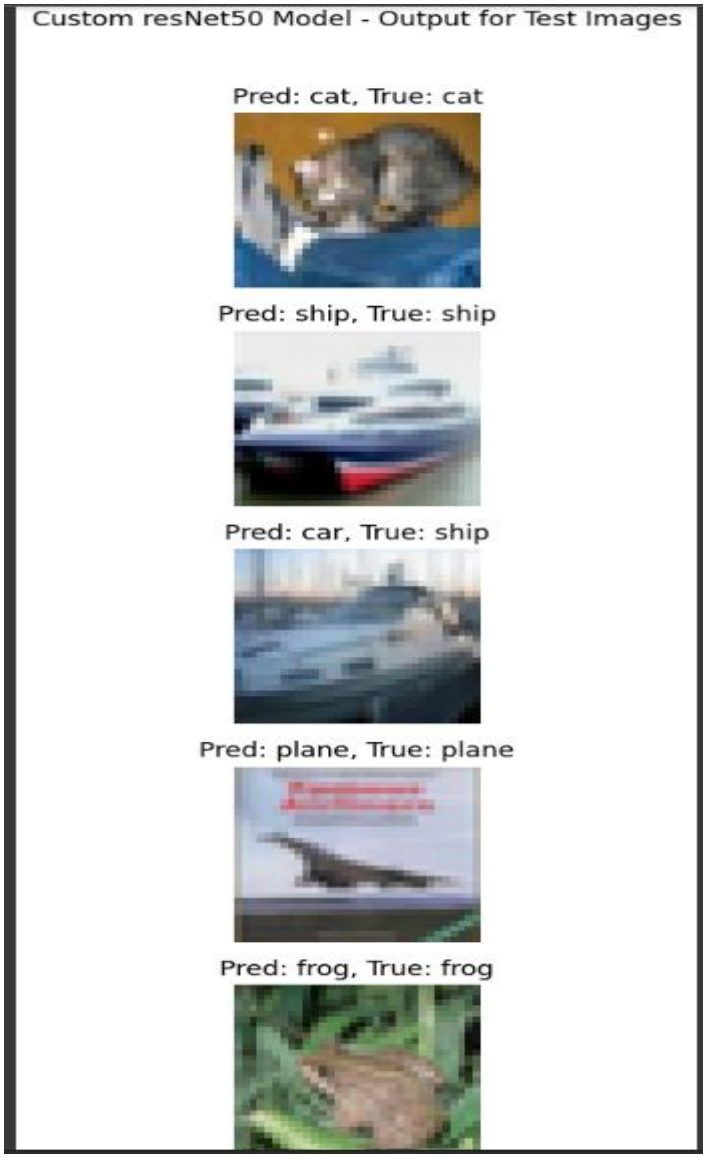
```
#Test resnet_model
eval(resnet_model, test_dl)
```

```
Results for testing:
Accuracy: 0.823
Precision: 0.827
Recall: 0.823
```



2-Test images for ResNet50 model

- In the following figure we passed the test data loader throw the network and printed some test images , here some of the printed plots:



Conclusion: the test accuracy was 82.3%, the overall performance was less than the VGG16, better than the GoogleNet and but much better than the ANN model.

Final Conclusion

the VGG16 model preformed best where it was the fastest and most accurate model followed by ResNet then GoogleNet and lastly our model (ANN) which was not complex enough to capture the patterns in our data.