The University of Western Ontario

Faculty of Engineering

CS 4442 – Artificial Intelligence 2

Assignment – 4 Report

By

Mohammad Ali Sarfraz: (250860782, msarfra2@uwo.ca)

Instructor: Yalda Mohsenzadeh

Date Submitted: April 11th, 2021

Question 1

a) The data has been downloaded and imported using the *torchvision* library in *Python* as seen in the attached notebook. The training batch has been randomized for the purposes of removing bias from the model training. This training data of 50,000 images is then split into a random 40,000 images for the testing batch and another 10,000 for the validation batch. To verify that the data has been imported properly, the first image and label of the testing batch was printed to the console as shown in *Figure 1* below.

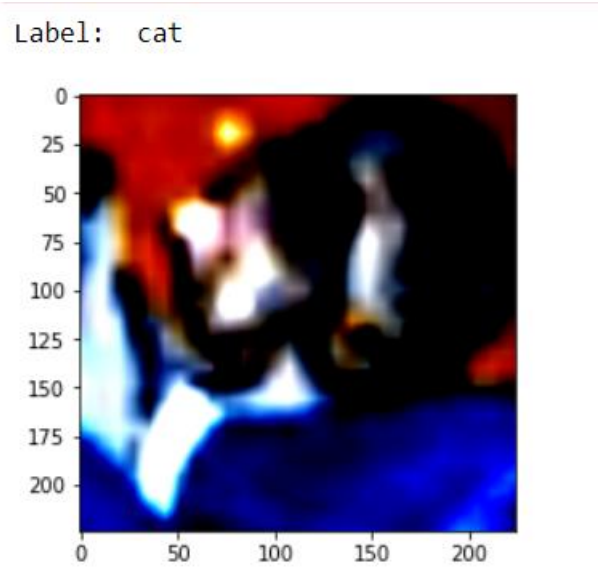

Figure 1: Verification image and label retrieved from the test data.

b) The test data was run through the downloaded pre-trained *AlexNet* model for analyzing the input image and returning the possibilities for all 1000 classes. The maximum value of these probabilities was inserted into a list and the process was repeated for all 10,000 test images. Each instance in this list was then counted to accumulate the total predictions for all 1000 possible ImageNet classes, and sorted in descending order for obtaining the top 10 most frequently predicted classes. These top 10 classes and the number of times they were predicted are then plotted against the 10 *CIFAR* classes in a confusion matrix. The resulting plot is shown in *Figure 2* below, where the top ten label predictions, in descending order, by *AlexNet* were:

```
[675, 335, 339, 510, 167, 653, 856, 152, 194, 491]
```

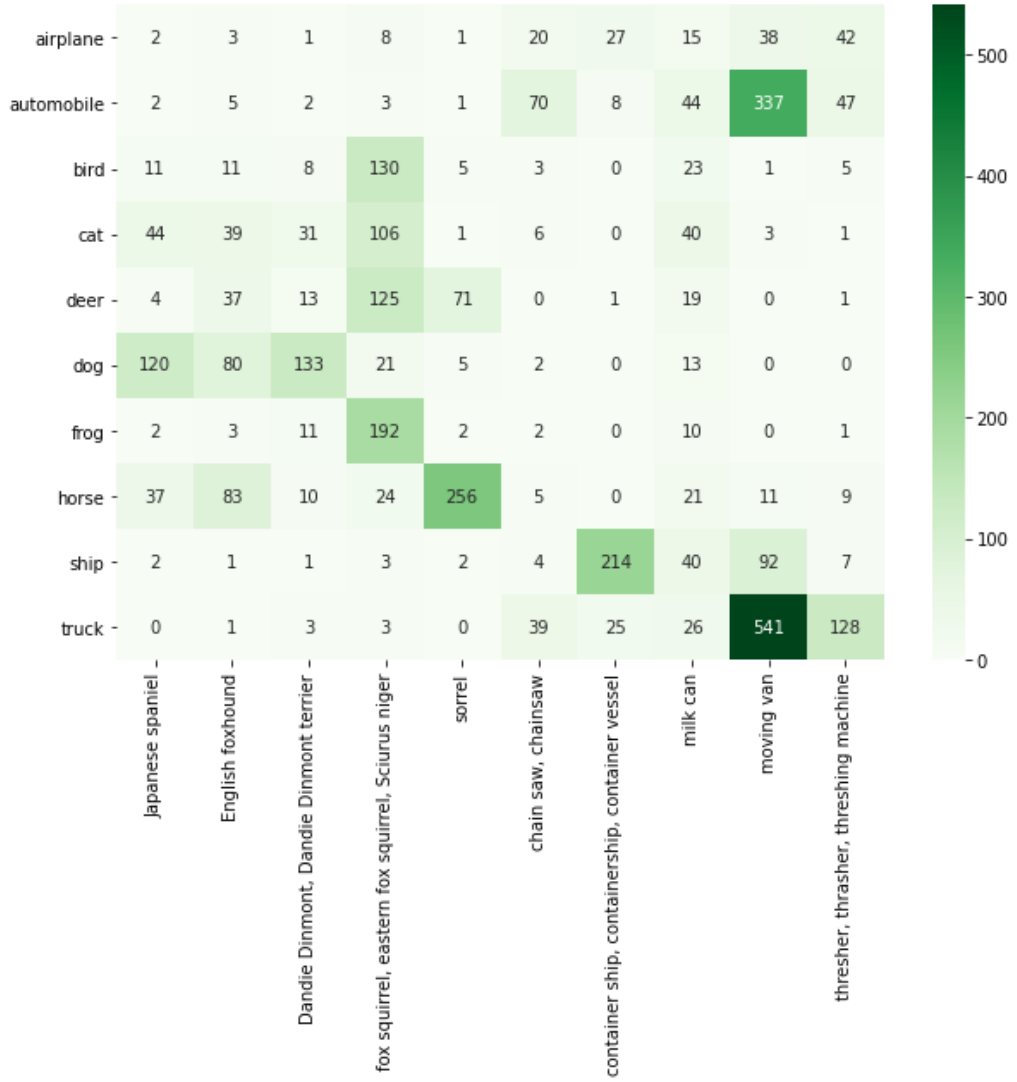

Figure 2: Confusion matrix relating the CIFAR10 classes with the 10 most frequent classes from ImageNet predicted by the model

c) The FC-6 layer output was obtained by chopping off all the proceeding layers from the base *AlexNet* model, hereby giving an output tensor of size 4096 instead of the usual 1000. To train the logistic regression model, two separate lists were created. The first list only contained the first 5000 elements of the training data, and the second list contained all 40,000. Each list was populated with the outputs of the FC-6 model and reduced to a one-dimensional *numpy* array.

The 5000-element list was used a means to tune the logistic regression for finding out what the best hyperparameter was with an L1-regularized model. This was done in an attempt to minimize the computation time required to train and fit the model, since fitting with all 40,000 elements is very expensive for five iterations at a time given the fact that higher hyperparameter values penalize the model more for making errors, hereby taking longer to find the perfect decision

boundary. Once each model was trained using the 5000 training elements, the accuracy was determined using the 10,000 validation data elements. Based on these scores, the best hyperparameter value was selected, since this value will be the best regardless of how many training elements are parsed into the model. A depiction of the accuracy scores for each iteration of the tuning loop is shown in *Figure 3* below.

```
For C =  0.01  accuracy =  0.7467
For C =  0.1  accuracy =  0.7864
For C =  1.0  accuracy =  0.7569
For C =  10.0  accuracy =  0.7518
For C =  100.0  accuracy =  0.7514
```

Figure 3: Accuracy scores obtained from the hyperparameter tuning of the FC-6 layer.

From the observations made in the 5000-element list, the appropriate hyperparameter value of 0.1 was chosen for training the model once again using all 40,000 elements and validating using the validation sets. Finally, the trained logistic regression model was fed the 10,000 test data elements for computing its performance and prediction accuracy, which was a value of **82.04%** overall.

d) The same process was repeated to extract the features of the FC-7 layer, which was deeper down the base *AlexNet's* model, but still output a tensor of the same size of 4096, allowing for the reuse of code snippets from the previous section. Hence, two lists of size 5000 and 40,000 respectively were populated using the data from the FC-7 layer and used to tune and verify the second logistic regression model. The resulting hyperparameter tuning with the 5000 images is shown in *Figure 4* below.

```
For C =  0.01  accuracy =  0.7517
For C =  0.1  accuracy =  0.7905
For C =  1.0  accuracy =  0.7744
For C =  10.0  accuracy =  0.7656
For C =  100.0  accuracy =  0.7671
```

Figure 4: Accuracy scores obtained from the hyperparameter tuning of the FC-7 layer.

Once again it can be seen that the hyperparameter value of 0.1 results in the best accuracy score on the verification data. Hence, the second logistic regression model was trained with 40,000 images using this hyperparameter value and validated using the 10,000 elements from the validation set. Finally, the trained logistic regression model was fed the 10,000 test data elements for computing its performance and prediction accuracy, which was a value of **83.73%** overall.

Discussion

The predictions made by the complete *AlexNet* show that the top ten most frequently predicted classes using the *CIFAR10* data are as follows:

- Japanese spaniel
- English foxhound
- Dandie Dinmont, Dandie Dinmont terrier
- fox squirrel, eastern fox squirrel, Sciurus niger
- sorrel
- chain saw, chainsaw
- container ship, containership, container vessel
- milk can
- moving van
- thresher, thrasher, threshing machine

The accuracy score obtained for the FC6 trained model of the *AlexNet* was slightly lower than the FC7 trained model. This was unexpected because the FC6 layer extracts features that are less detailed than the features extracted from the FC7 layer, meaning FC6 will have more "raw" features than FC7. This means that the model trained using FC7 features will have an overfit towards data received from the *ImageNet* bank, and hence should have a difficult time classifying images from the *CIFAR10* bank. Consequently, since the FC6 layer outputs more generic/raw image features like edges and shapes, the model trained using this data will be able to better classify images in the *CIFAR10* bank because of the similarity in the raw features of the data.

It should be noted that the accuracies for both models are very close, and so the results may fall on either side of the problem, i.e. FC6 accuracies may have been higher than FC7 accuracies as expected, and it all depends on the training and validation data randomization. Another possible explanation for why the results are not what was expected is that the test data itself is biased toward the more prominent features extracted by the FC7 model, or there may be a slight error in the building of the models themselves.

Appendix

FC-6 Model Structure

```
AlexNet(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(11, 11), stride=(4, 4), padding=(2, 2))
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(64, 192, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU(inplace=True)
    (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU(inplace=True)
    (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(6, 6))
  (classifier): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=9216, out_features=4096, bias=True)
  )
)
```

FC-7 Model Structure

```
AlexNet(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(11, 11), stride=(4, 4), padding=(2, 2))
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(64, 192, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU(inplace=True)
    (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU(inplace=True)
    (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(6, 6))
  (classifier): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=9216, out_features=4096, bias=True)
    (2): ReLU(inplace=True)
    (3): Dropout(p=0.5, inplace=False)
    (4): Linear(in_features=4096, out_features=4096, bias=True)
  )
)
```