Final Project Individual Report: Machine Learning II
Submitted to: Dr. Amir Jafari
By Manoah Farmer

**Project Title**
Image Classification on Online Stores Fashion Dataset

**Introduction.** An overview of the project and an outline of the shared work.

The aim of this project is to build a Convolution Neural Network to Identify different types of online clothing items. The model will be evaluated with the following metrics: accuracy, F1 score, precision and recall, as well as AUC and ROC curves. To achieve this we will be tuning hyperparameters such as the number of epochs, number of layers, kernel size, etc. to improve our model. We did most of this project together as a group since we were able to meet. This included the some of the report. And the data processing as well as decisions about which frameworks, and architecture to use for our model.

**Description of your individual work**.

I worked on the report heavily, as well as deciding on the structure of our models. I started this process by using the MNIST code found in the github, but using the CIFAR dataset. When I was able to get the original two-layer code running well, I then began experimenting with adding layers and dropout nodes. In this experimentation dropout nodes did not improve the model, however, adding layers did. I also used this code to observe the behavior that occurred when adding a bottleneck feature and found that for that dataset, it did perform better. This result, however, was not consistent when we made these attempts on our clothing dataset.

**Portion of work.**

As stated above, I helped build our original algorithms to practice running the data in pytorch. I also wrote a script that we were originally going to use for determining how to resize our images, but we ended up using an arbitrary number for this value instead. I am responsible for the wording of most of the report and conducted the necessary steps on github to upload our project.

**Results.**

With Channel of 3 and the same random sample overtime with set.seed (1122), our results are summarized as below.

| Epoch | Iteration | Batchsize | Layers | Accuracy | Time | Image Size | Optimizer |
|---|---|---|---|---|---|---|---|
| 1 | 4 | 100 | 2 | 65% | -- | 28x28 | Adam |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 4 | 100 | 3 | 56% | 104.281539 sec | 224x224 | Adam |
| 2 | 7 | 64 | | 65% | 11121.987646 sec | 224x224 | Adam |
| 4 | 7 | 64 | 5 (bottleneck) 4 (maxpool) | 70% | 422.83998 sec | 224x224 | Adam |
| 5 | 7 | 64 | 5 (bottleneck) 4 (maxpool) | 72% | 717.42504 sec | 224x224 | Adam |
| 5 | 7 | 64 | 5 (straight) 4 (maxpool) | 76% | 795.74 sec | 224x224 | Adam |
| 5 | 7 | 64 | 5 (bottleneck) 4 (maxpool) | 80% | 473.503 sec | 224x224 | SGD |
| 5 | 7 | 64 | 5 (bottleneck) 4 (maxpool) | 79% | 481.23 sec | 224x224 | Adagrad |
| 10 | 7 | 64 | 5 (bottleneck) 4 (maxpool) | 79% | 1542.556 sec | 224x224 | SGD |
| 10 | 7 | 64 | 5 (bottleneck) 4 (maxpool) | 81% | 972.23 sec | 224x224 | Adagrad |
| 5 | 7 | 64 | 5 (bottleneck) 4 (maxpool) | 54% | 1025.029 sec | 224x224 | Rprop |
| 10 | 7 | 64 | 7 (straight) 4 (maxpool) | 71% | 1727.82sec | 224x224 | Adam |
| 10 | 7 | 64 | 7(bottleneck) 4(maxpool) | 69% | 1511.426 sec | 224x224 | Adam |

With channel of 3 and set.seed to 1122, we tuned different hyperparameters to see changes that can cause to the results.

From the results we can interpret that the accuracy rate goes up when the layers are "straight" instead of bottleneck. Also when comparing the time it takes, we finds that the time it takes increases when the layers are straight. There seems to be about 4% differences between straight and bottleneck approach. Therefore, for this dataset, the straight approach performs better than the bottleneck approach and we find that straight approach isn't preferable in this image classification. The models that have the "straight" specification were run with identical layers with the exception of the max-pooling function which were only applied to the the four outermost layers to avoid feature map size issues.

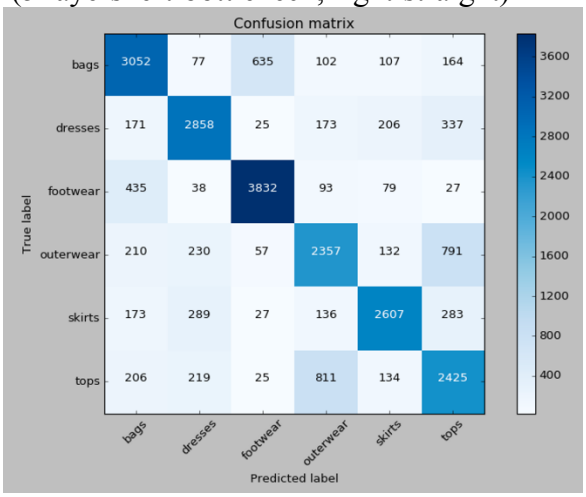
Changing kernel-size resulted in a lower accuracy of the model for our 7 layer model. The models that have the "straight" specification were run with identical layers with the exception of the max-pooling function which were only applied to the the four outermost layers to avoid feature map size issues. Changing the optimizer to SGD changes the result of the performance and increase the performance of our model. The model accuracy rate increases from 72% to 80%, which is a significant 8% increase.

The total time to run the model decreases with the change in the optimizer as well. The total time with the Adam optimizer was 795.74 sec and the total time with the SGD optimizer was 473.50 while keeping all the other parameters the same, which shows that SGD optimizer not only increases the performance of the model accuracy but also saves the time it took to run the model.

SGD, Stochastic gradient descent optimizer is called stochastic because samples are selected randomly instead of a single group. While it performs better than Adam, increasing the layer using the SGD doesn't improve the model performance, and the total time took longer.

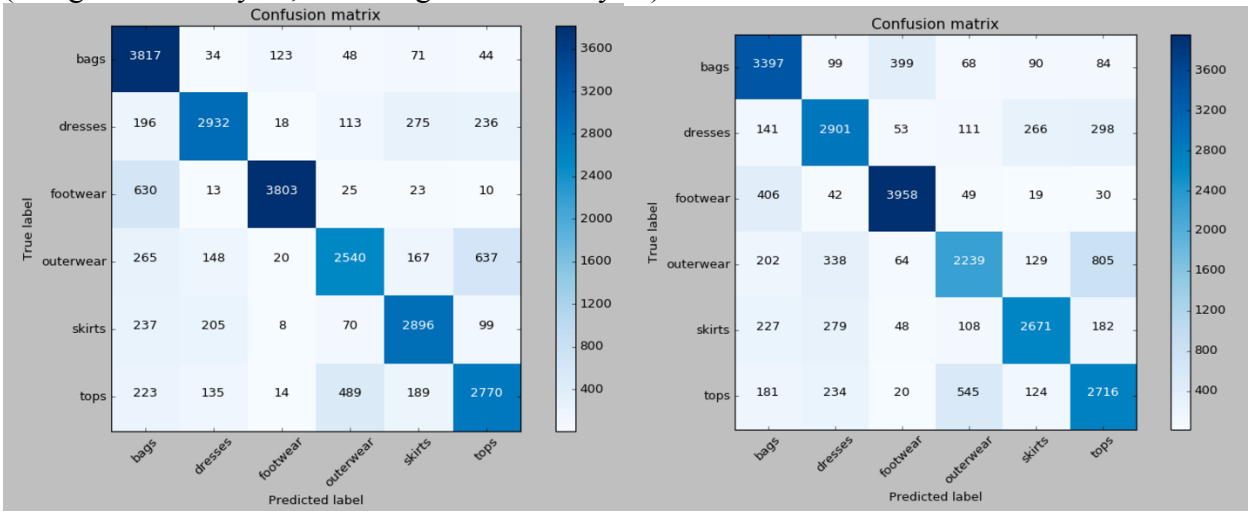
**Confusion Matrix**

The bottom two graphs are confusion matrix obtained comparing the bottleneck and straight approach with 5 layers convolutional neural network with Adam optimizer. From the diagonal line, we can observed that the straight approach with 5 layers has better accuracy.
(5 layers left-bottleneck, right-straight)



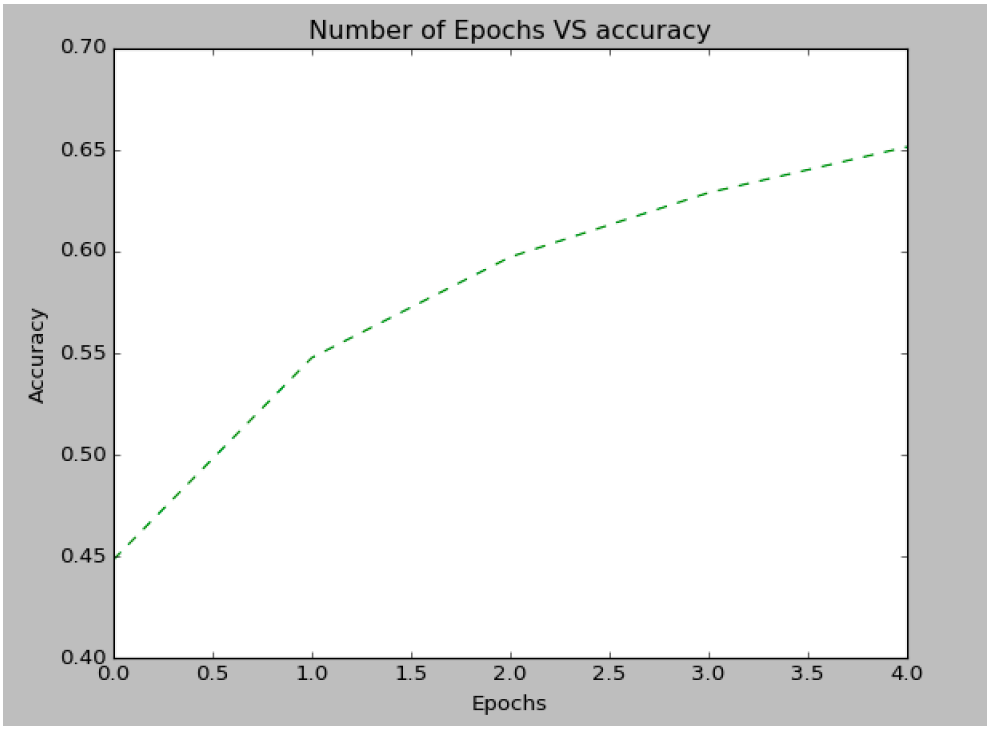

When comparing the confusion matrix models using of SGD optimizer and Adam optimizer with straight technique, we find that SGD optimizer has accuracy scores that are higher than the accuracy scores of Adam optimizer.

(straight SGD 5 layers, and straight Adam 7 layers)




**Accuray vs Epoch**

From the bottleneck approach with 5 layers, we also observed that the accuracy rates increase with the increase in the number of epoch from 0.45 to 0.65 in this case.

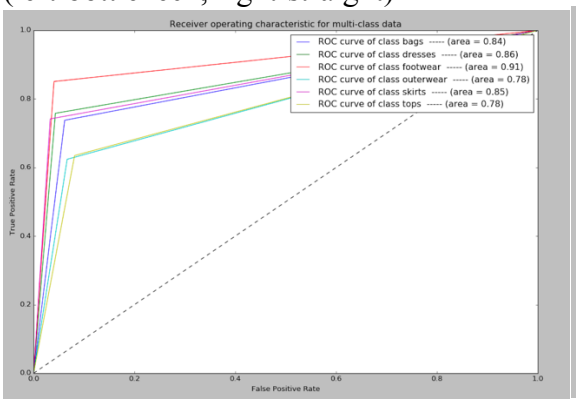


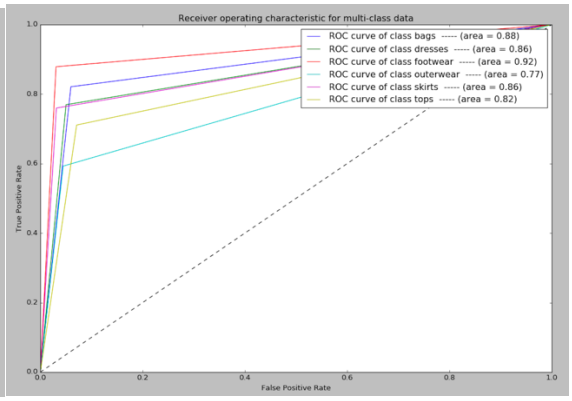

**The ROC curve**

Receiver Operating Characteristic (ROC) curve is defined as "the true positive rate (Sensitivity) is plotted function of the false positive rate (100-Specificity) for different cut-off

points". From the two graphs below, we can see that the right graph has the curve which is closer to the upper corner and we can interpret that it has more accuracy. On the other hand, we can also interpret using the area under the curve (AUC) to determine and compare the accuracy of this bottleneck and straight approach. The areas under the graphs are specified in the box located at the upper corner for each item, and we find that the footwears has the highest accuracy in our image classification when comparing the 5 layers with bottleneck technique and straight technique.

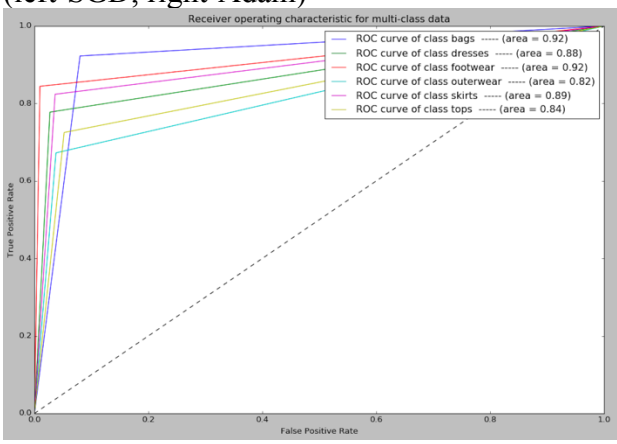
(left-bottleneck, right-straight)

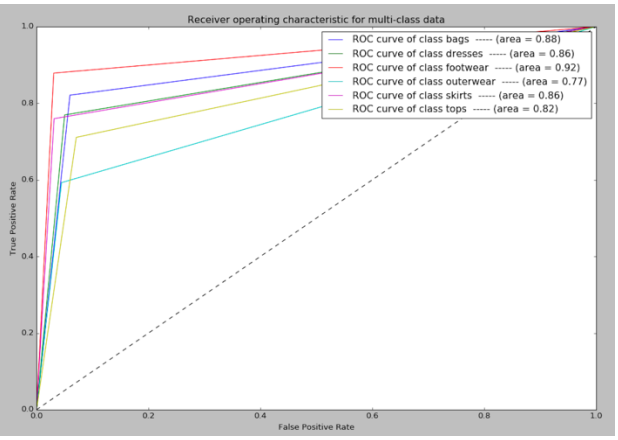



The two graphs below shows the SGD and Adam ROC graphs and the graphs show that SGD performs better than Adam optimizer.

(left-SGD, right-Adam)




**Accuray rates on each class (SGD best model)**

Test Accuracy of the model on the test images: 80 %

Accuracy of bags : 85 %
Accuracy of dresses : 83 %
Accuracy of footwear : 90 %
Accuracy of outerwear : 74 %
Accuracy of skirts : 88 %
Accuracy of tops : 62 %

**Classification Report (SGD best model)**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| bags | 0.84 | 0.85 | 0.85 | 4137 |
| dresses | 0.81 | 0.79 | 0.80 | 3770 |
| footwear | 0.94 | 0.89 | 0.92 | 4504 |
| outerwear | 0.69 | 0.77 | 0.73 | 3777 |
| skirts | 0.75 | 0.86 | 0.80 | 3515 |
| tops | 0.79 | 0.65 | 0.71 | 3820 |
| avg / total | 0.81 | 0.81 | 0.81 | 23523 |

{0: 0.9078810302987728, 1: 0.8789473337898108, 2: 0.9406002873447017, 3: 0.8529724442436417, 4: 0.9042918181233909, 5: 0.8077317271428355}

**Remaining Results:**

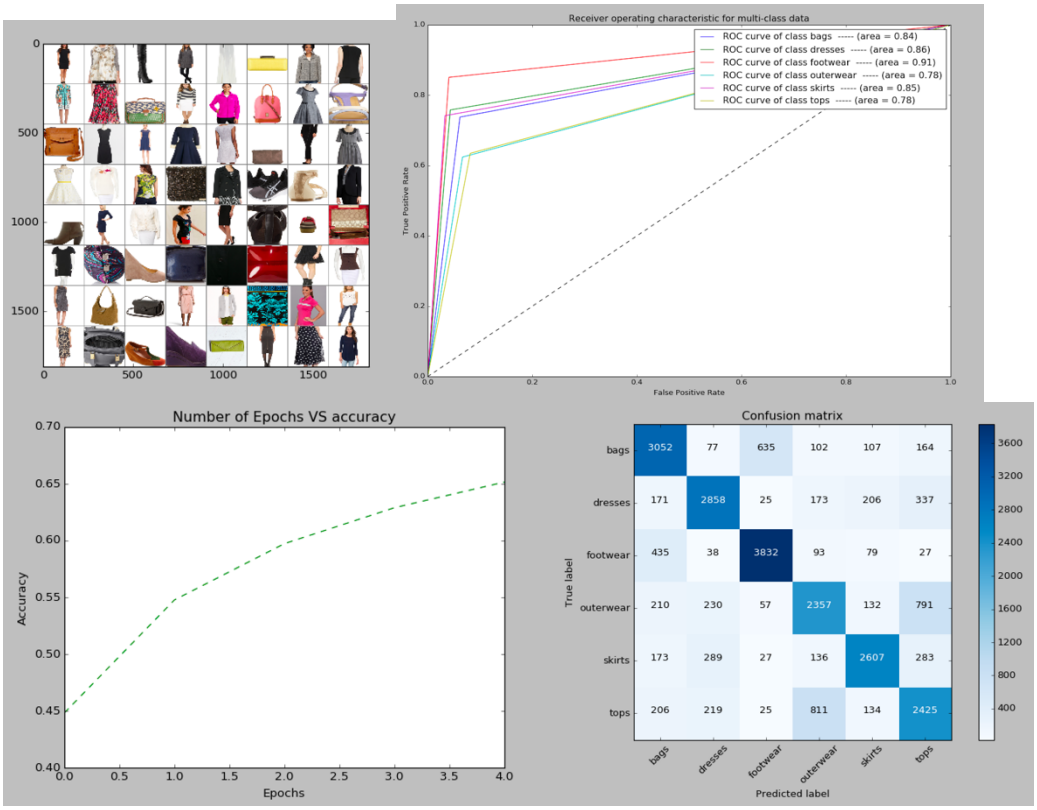
**5 Layers bottle neck 4 max pool, 5 epochs, 64 batchsize**

images shape on batch size = torch.Size([64, 3, 224, 224])
labels shape on batch size = torch.Size([64])
Epoch [1/5], Iter [100/734] Loss: 2.1190
Epoch [1/5], Iter [200/734] Loss: 1.6336
Epoch [1/5], Iter [300/734] Loss: 1.7083
Epoch [1/5], Iter [400/734] Loss: 1.4623
Epoch [1/5], Iter [500/734] Loss: 1.0955
Epoch [1/5], Iter [600/734] Loss: 1.1057
Epoch [1/5], Iter [700/734] Loss: 1.0495
Epoch [2/5], Iter [100/734] Loss: 1.0009
Epoch [2/5], Iter [200/734] Loss: 0.5802
Epoch [2/5], Iter [300/734] Loss: 0.8031
Epoch [2/5], Iter [400/734] Loss: 0.8572
Epoch [2/5], Iter [500/734] Loss: 0.8639
Epoch [2/5], Iter [600/734] Loss: 0.6727
Epoch [2/5], Iter [700/734] Loss: 1.1910
Epoch [3/5], Iter [100/734] Loss: 0.6867
Epoch [3/5], Iter [200/734] Loss: 0.8000
Epoch [3/5], Iter [300/734] Loss: 0.6526
Epoch [3/5], Iter [400/734] Loss: 0.9673
Epoch [3/5], Iter [500/734] Loss: 0.7886
Epoch [3/5], Iter [600/734] Loss: 0.7936
Epoch [3/5], Iter [700/734] Loss: 0.8199
Epoch [4/5], Iter [100/734] Loss: 0.9262
Epoch [4/5], Iter [200/734] Loss: 0.5819
Epoch [4/5], Iter [300/734] Loss: 0.5535
Epoch [4/5], Iter [400/734] Loss: 0.9992
Epoch [4/5], Iter [500/734] Loss: 0.7444
Epoch [4/5], Iter [600/734] Loss: 0.4865
Epoch [4/5], Iter [700/734] Loss: 0.5948
Epoch [5/5], Iter [100/734] Loss: 0.6216
Epoch [5/5], Iter [200/734] Loss: 0.7800

Epoch [5/5], Iter [300/734] Loss: 0.6399
Epoch [5/5], Iter [400/734] Loss: 0.4969
Epoch [5/5], Iter [500/734] Loss: 0.8024
Epoch [5/5], Iter [600/734] Loss: 0.5087
Epoch [5/5], Iter [700/734] Loss: 0.5625
Training set Accuracy of the model on the Train images: 65 %
717.424504 seconds
Test Accuracy of the model on the test images: 72 %
Accuracy of  bags : 74 %
Accuracy of dresses : 73 %
Accuracy of footwear : 86 %
Accuracy of outerwear : 65 %
Accuracy of skirts : 75 %
Accuracy of  tops : 66 %

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| bags | 0.72 | 0.74 | 0.73 | 4137 |
| dresses | 0.77 | 0.76 | 0.76 | 3770 |
| footwear | 0.83 | 0.85 | 0.84 | 4504 |
| outerwear | 0.64 | 0.62 | 0.63 | 3777 |
| skirts | 0.80 | 0.74 | 0.77 | 3515 |
| tops | 0.60 | 0.63 | 0.62 | 3820 |
| avg / total | 0.73 | 0.73 | 0.73 | 23523 |

{0: 0.8380451170738629, 1: 0.857453435874697, 2: 0.9051830192804298, 3: 0.7787222386742125, 4: 0.8543958376819968, 5: 0.776754669406126}



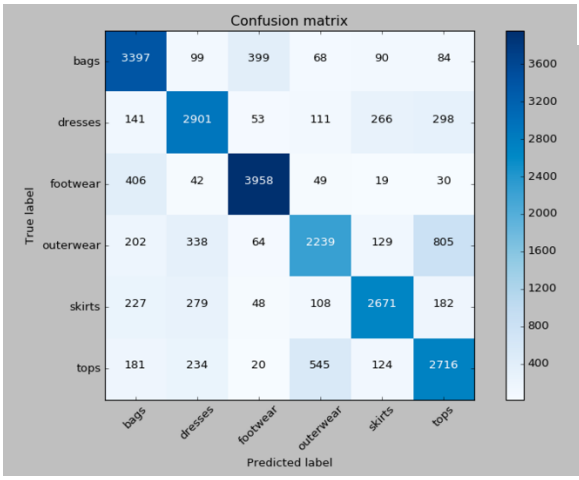

**5 Layers straight 4 max pool, 5 epochs, 64 batchsize**

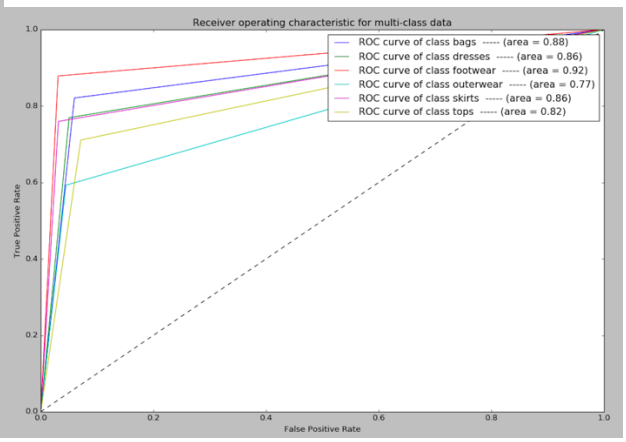
images shape on batch size = torch.Size([64, 3, 224, 224])
labels shape on batch size = torch.Size([64])
bags footwear outerwear  tops outerwear  bags
Epoch [1/5], Iter [100/734] Loss: 2.3676
Epoch [1/5], Iter [200/734] Loss: 1.6400
Epoch [1/5], Iter [300/734] Loss: 1.1064
Epoch [1/5], Iter [400/734] Loss: 1.2984
Epoch [1/5], Iter [500/734] Loss: 1.1671
Epoch [1/5], Iter [600/734] Loss: 1.0433
Epoch [1/5], Iter [700/734] Loss: 0.8376
Epoch [2/5], Iter [100/734] Loss: 1.0417
Epoch [2/5], Iter [200/734] Loss: 0.7722
Epoch [2/5], Iter [300/734] Loss: 0.8563
Epoch [2/5], Iter [400/734] Loss: 1.0028
Epoch [2/5], Iter [500/734] Loss: 0.6126
Epoch [2/5], Iter [600/734] Loss: 0.9350
Epoch [2/5], Iter [700/734] Loss: 0.6436
Epoch [3/5], Iter [100/734] Loss: 0.7296
Epoch [3/5], Iter [200/734] Loss: 0.4224
Epoch [3/5], Iter [300/734] Loss: 0.5868
Epoch [3/5], Iter [400/734] Loss: 0.5187
Epoch [3/5], Iter [500/734] Loss: 0.7172
Epoch [3/5], Iter [600/734] Loss: 0.8649

Epoch [3/5], Iter [700/734] Loss: 0.7659
Epoch [4/5], Iter [100/734] Loss: 0.6213
Epoch [4/5], Iter [200/734] Loss: 0.9780
Epoch [4/5], Iter [300/734] Loss: 0.8408
Epoch [4/5], Iter [400/734] Loss: 0.5910
Epoch [4/5], Iter [500/734] Loss: 0.5645
Epoch [4/5], Iter [600/734] Loss: 0.5254
Epoch [4/5], Iter [700/734] Loss: 0.6211
Epoch [5/5], Iter [100/734] Loss: 0.6844
Epoch [5/5], Iter [200/734] Loss: 0.7029
Epoch [5/5], Iter [300/734] Loss: 0.5809
Epoch [5/5], Iter [400/734] Loss: 0.8297
Epoch [5/5], Iter [500/734] Loss: 0.5689
Epoch [5/5], Iter [600/734] Loss: 0.3840
Epoch [5/5], Iter [700/734] Loss: 0.5465
Training set Accuracy of the model on the Train images: 67 %
795.74182 seconds
Test Accuracy of the model on the test images: 76 %
Accuracy of  bags : 85 %
Accuracy of dresses : 78 %
Accuracy of footwear : 88 %
Accuracy of outerwear : 59 %
Accuracy of skirts : 80 %
Accuracy of  tops : 75 %

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| bags | 0.75 | 0.82 | 0.78 | 4137 |
| dresses | 0.75 | 0.77 | 0.76 | 3770 |
| footwear | 0.87 | 0.88 | 0.88 | 4504 |
| outerwear | 0.72 | 0.59 | 0.65 | 3777 |
| skirts | 0.81 | 0.76 | 0.78 | 3515 |
| tops | 0.66 | 0.71 | 0.68 | 3820 |
| avg / total | 0.76 | 0.76 | 0.76 | 23523 |

{0: 0.8807220875960889, 1: 0.8596379007533489, 2: 0.9240341433830263, 3: 0.7740909430626783, 4: 0.8642493784847369, 5: 0.8199951744133365}
GroundTruth:   bags  bags  bags  bags
Predicted:   bags footwear  bags outerwear




## SGD

images shape on batch size = torch.Size([64, 3, 224, 224])
labels shape on batch size = torch.Size([64])
Epoch [1/5], Iter [100/734] Loss: 1.1525
Epoch [1/5], Iter [200/734] Loss: 0.8715
Epoch [1/5], Iter [300/734] Loss: 0.8618
Epoch [1/5], Iter [400/734] Loss: 0.6990
Epoch [1/5], Iter [500/734] Loss: 0.7519
Epoch [1/5], Iter [600/734] Loss: 0.8176
Epoch [1/5], Iter [700/734] Loss: 0.8236
Epoch [2/5], Iter [100/734] Loss: 0.5339
Epoch [2/5], Iter [200/734] Loss: 0.5881
Epoch [2/5], Iter [300/734] Loss: 0.7327
Epoch [2/5], Iter [400/734] Loss: 0.5550
Epoch [2/5], Iter [500/734] Loss: 0.6153
Epoch [2/5], Iter [600/734] Loss: 0.5799
Epoch [2/5], Iter [700/734] Loss: 0.5511
Epoch [3/5], Iter [100/734] Loss: 0.4423
Epoch [3/5], Iter [200/734] Loss: 0.6513
Epoch [3/5], Iter [300/734] Loss: 0.3372
Epoch [3/5], Iter [400/734] Loss: 0.5801
Epoch [3/5], Iter [500/734] Loss: 0.5338
Epoch [3/5], Iter [600/734] Loss: 0.7010
Epoch [3/5], Iter [700/734] Loss: 0.5461
Epoch [4/5], Iter [100/734] Loss: 0.4753
Epoch [4/5], Iter [200/734] Loss: 0.3725
Epoch [4/5], Iter [300/734] Loss: 0.3751
Epoch [4/5], Iter [400/734] Loss: 0.6334
Epoch [4/5], Iter [500/734] Loss: 0.4577
Epoch [4/5], Iter [600/734] Loss: 0.6039
Epoch [4/5], Iter [700/734] Loss: 0.3366
Epoch [5/5], Iter [100/734] Loss: 0.3060
Epoch [5/5], Iter [200/734] Loss: 0.4821
Epoch [5/5], Iter [300/734] Loss: 0.8020

Epoch [5/5], Iter [400/734] Loss: 0.4375
Epoch [5/5], Iter [500/734] Loss: 0.5330
Epoch [5/5], Iter [600/734] Loss: 0.5881
Epoch [5/5], Iter [700/734] Loss: 0.5916
473.502485 seconds
Test Accuracy of the model on the test images: 80 %

Accuracy of  bags : 85 %
Accuracy of dresses : 83 %
Accuracy of footwear : 90 %
Accuracy of outerwear : 74 %
Accuracy of skirts : 88 %
Accuracy of  tops : 62 %

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| bags | 0.84 | 0.85 | 0.85 | 4137 |
| dresses | 0.81 | 0.79 | 0.80 | 3770 |
| footwear | 0.94 | 0.89 | 0.92 | 4504 |
| outerwear | 0.69 | 0.77 | 0.73 | 3777 |
| skirts | 0.75 | 0.86 | 0.80 | 3515 |
| tops | 0.79 | 0.65 | 0.71 | 3820 |
| avg / total | 0.81 | 0.81 | 0.81 | 23523 |

{0: 0.9078810302987728, 1: 0.8789473337898108, 2: 0.9406002873447017, 3: 0.8529724442436417, 4: 0.9042918181233909, 5: 0.8077317271428355}

**7Layers_straight_4maxpool_10epochs_64batchsize**

cuda:0
<class 'torch.utils.data.dataloader._DataLoaderIter'>
images shape on batch size = torch.Size([64, 3, 224, 224])
labels shape on batch size = torch.Size([64])
Epoch [1/10], Iter [100/734] Loss: 1.7384
Epoch [1/10], Iter [200/734] Loss: 1.5978
Epoch [1/10], Iter [300/734] Loss: 1.6892
Epoch [1/10], Iter [400/734] Loss: 1.1238
Epoch [1/10], Iter [500/734] Loss: 1.1008
Epoch [1/10], Iter [600/734] Loss: 1.7555
Epoch [1/10], Iter [700/734] Loss: 1.2471
Epoch [2/10], Iter [100/734] Loss: 1.2666
Epoch [2/10], Iter [200/734] Loss: 1.0766
Epoch [2/10], Iter [300/734] Loss: 1.1622
Epoch [2/10], Iter [400/734] Loss: 0.8324
Epoch [2/10], Iter [500/734] Loss: 1.0236
Epoch [2/10], Iter [600/734] Loss: 1.4816
Epoch [2/10], Iter [700/734] Loss: 1.0749
Epoch [3/10], Iter [100/734] Loss: 1.0550
Epoch [3/10], Iter [200/734] Loss: 0.8160

Epoch [3/10], Iter [300/734] Loss: 0.9476
Epoch [3/10], Iter [400/734] Loss: 0.8308
Epoch [3/10], Iter [500/734] Loss: 0.8286
Epoch [3/10], Iter [600/734] Loss: 1.0995
Epoch [3/10], Iter [700/734] Loss: 0.8181
Epoch [4/10], Iter [100/734] Loss: 0.9433
Epoch [4/10], Iter [200/734] Loss: 0.9808
Epoch [4/10], Iter [300/734] Loss: 0.9997
Epoch [4/10], Iter [400/734] Loss: 0.8923
Epoch [4/10], Iter [500/734] Loss: 0.9113
Epoch [4/10], Iter [600/734] Loss: 0.9280
Epoch [4/10], Iter [700/734] Loss: 0.9028
Epoch [5/10], Iter [100/734] Loss: 0.7001
Epoch [5/10], Iter [200/734] Loss: 0.9468
Epoch [5/10], Iter [300/734] Loss: 0.9191
Epoch [5/10], Iter [400/734] Loss: 0.8411
Epoch [5/10], Iter [500/734] Loss: 0.9765
Epoch [5/10], Iter [600/734] Loss: 0.9019
Epoch [5/10], Iter [700/734] Loss: 0.8397
Epoch [6/10], Iter [100/734] Loss: 0.7925
Epoch [6/10], Iter [200/734] Loss: 1.1802
Epoch [6/10], Iter [300/734] Loss: 0.9014
Epoch [6/10], Iter [400/734] Loss: 0.6037
Epoch [6/10], Iter [500/734] Loss: 0.9972
Epoch [6/10], Iter [600/734] Loss: 0.9715
Epoch [6/10], Iter [700/734] Loss: 0.8063
Epoch [7/10], Iter [100/734] Loss: 0.6520
Epoch [7/10], Iter [200/734] Loss: 1.0989
Epoch [7/10], Iter [300/734] Loss: 0.8753
Epoch [7/10], Iter [400/734] Loss: 0.7345
Epoch [7/10], Iter [500/734] Loss: 0.6590
Epoch [7/10], Iter [600/734] Loss: 0.7322
Epoch [7/10], Iter [700/734] Loss: 0.7358
Epoch [8/10], Iter [100/734] Loss: 0.8404
Epoch [8/10], Iter [200/734] Loss: 0.5090
Epoch [8/10], Iter [300/734] Loss: 0.8169
Epoch [8/10], Iter [400/734] Loss: 0.6299
Epoch [8/10], Iter [500/734] Loss: 0.6758
Epoch [8/10], Iter [600/734] Loss: 0.7371
Epoch [8/10], Iter [700/734] Loss: 0.5271
Epoch [9/10], Iter [100/734] Loss: 0.5350
Epoch [9/10], Iter [200/734] Loss: 0.8278
Epoch [9/10], Iter [300/734] Loss: 1.0682
Epoch [9/10], Iter [400/734] Loss: 1.0672
Epoch [9/10], Iter [500/734] Loss: 0.7146
Epoch [9/10], Iter [600/734] Loss: 0.6417
Epoch [9/10], Iter [700/734] Loss: 0.8504
Epoch [10/10], Iter [100/734] Loss: 0.6552
Epoch [10/10], Iter [200/734] Loss: 0.6977
Epoch [10/10], Iter [300/734] Loss: 0.7299
Epoch [10/10], Iter [400/734] Loss: 0.7889

Epoch [10/10], Iter [500/734] Loss: 0.8208
Epoch [10/10], Iter [600/734] Loss: 0.5978
Epoch [10/10], Iter [700/734] Loss: 0.6368
Training set Accuracy of the model on the Train images: 64 %
1727.824153 seconds
Test Accuracy of the model on the test images: 71 %
Accuracy of  bags : 81 %
Accuracy of dresses : 74 %
Accuracy of footwear : 76 %
Accuracy of outerwear : 63 %
Accuracy of skirts : 81 %
Accuracy of  tops : 52 %

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| bags | 0.72 | 0.80 | 0.76 | 4137 |
| dresses | 0.72 | 0.78 | 0.75 | 3770 |
| footwear | 0.91 | 0.75 | 0.82 | 4504 |
| outerwear | 0.58 | 0.66 | 0.62 | 3777 |
| skirts | 0.71 | 0.78 | 0.74 | 3515 |
| tops | 0.69 | 0.55 | 0.61 | 3820 |
| avg / total | 0.73 | 0.72 | 0.72 | 23523 |

{0: 0.8663738370587628, 1: 0.8597762875491094, 2: 0.8648170155076297, 3: 0.7844414372297841, 4: 0.8623133747923306, 5: 0.7515264172968583}

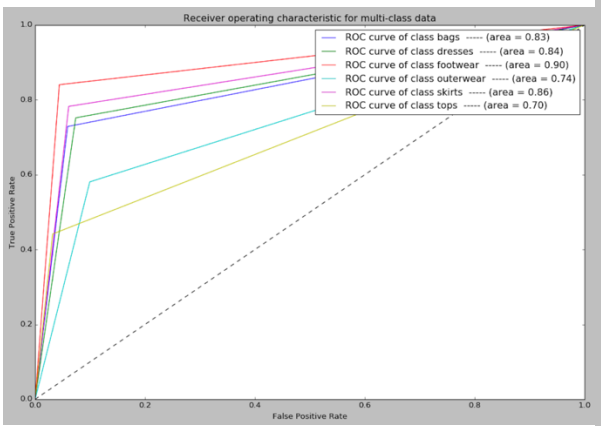
**7Layers_straight_4maxpool_10epochs_64batchsize_3kernelsize**
images shape on batch size = torch.Size([64, 3, 224, 224])
labels shape on batch size = torch.Size([64])
Epoch [1/10], Iter [100/734] Loss: 1.5868
Epoch [1/10], Iter [200/734] Loss: 1.5498
Epoch [1/10], Iter [300/734] Loss: 1.3855
Epoch [1/10], Iter [400/734] Loss: 1.4096
Epoch [1/10], Iter [500/734] Loss: 1.4473
Epoch [1/10], Iter [600/734] Loss: 1.3095
Epoch [1/10], Iter [700/734] Loss: 1.2992
Epoch [2/10], Iter [100/734] Loss: 1.1442
Epoch [2/10], Iter [200/734] Loss: 1.2236
Epoch [2/10], Iter [300/734] Loss: 0.9719
Epoch [2/10], Iter [400/734] Loss: 0.8542
Epoch [2/10], Iter [500/734] Loss: 1.0130
Epoch [2/10], Iter [600/734] Loss: 0.8867
Epoch [2/10], Iter [700/734] Loss: 0.9214
Epoch [3/10], Iter [100/734] Loss: 0.8348
Epoch [3/10], Iter [200/734] Loss: 0.7168
Epoch [3/10], Iter [300/734] Loss: 0.9214
Epoch [3/10], Iter [400/734] Loss: 0.6325
Epoch [3/10], Iter [500/734] Loss: 0.8184

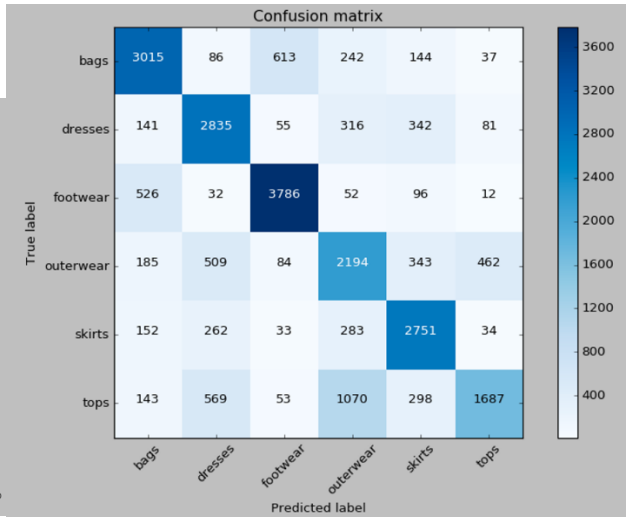
Epoch [3/10], Iter [600/734] Loss: 0.9593
Epoch [3/10], Iter [700/734] Loss: 0.7305
Epoch [4/10], Iter [100/734] Loss: 0.7207
Epoch [4/10], Iter [200/734] Loss: 0.8923
Epoch [4/10], Iter [300/734] Loss: 0.9438
Epoch [4/10], Iter [400/734] Loss: 0.6959
Epoch [4/10], Iter [500/734] Loss: 0.7747
Epoch [4/10], Iter [600/734] Loss: 0.7402
Epoch [4/10], Iter [700/734] Loss: 0.7599
Epoch [5/10], Iter [100/734] Loss: 0.7755
Epoch [5/10], Iter [200/734] Loss: 0.7499
Epoch [5/10], Iter [300/734] Loss: 0.7914
Epoch [5/10], Iter [400/734] Loss: 0.9518
Epoch [5/10], Iter [500/734] Loss: 0.6999
Epoch [5/10], Iter [600/734] Loss: 0.7043
Epoch [5/10], Iter [700/734] Loss: 0.7256
Epoch [6/10], Iter [100/734] Loss: 0.4863
Epoch [6/10], Iter [200/734] Loss: 0.8476
Epoch [6/10], Iter [300/734] Loss: 0.8546
Epoch [6/10], Iter [400/734] Loss: 0.6579
Epoch [6/10], Iter [500/734] Loss: 0.5809
Epoch [6/10], Iter [600/734] Loss: 0.5134
Epoch [6/10], Iter [700/734] Loss: 0.7616
Epoch [7/10], Iter [100/734] Loss: 0.5119
Epoch [7/10], Iter [200/734] Loss: 0.6025
Epoch [7/10], Iter [300/734] Loss: 0.6867
Epoch [7/10], Iter [400/734] Loss: 0.6755
Epoch [7/10], Iter [500/734] Loss: 0.4947
Epoch [7/10], Iter [600/734] Loss: 0.5777
Epoch [7/10], Iter [700/734] Loss: 0.7827
Epoch [8/10], Iter [100/734] Loss: 0.6306
Epoch [8/10], Iter [200/734] Loss: 0.4200
Epoch [8/10], Iter [300/734] Loss: 0.5762
Epoch [8/10], Iter [400/734] Loss: 0.5348
Epoch [8/10], Iter [500/734] Loss: 0.5426
Epoch [8/10], Iter [600/734] Loss: 0.6827
Epoch [8/10], Iter [700/734] Loss: 0.5600
Epoch [9/10], Iter [100/734] Loss: 0.7473
Epoch [9/10], Iter [200/734] Loss: 0.4525
Epoch [9/10], Iter [300/734] Loss: 0.5563
Epoch [9/10], Iter [400/734] Loss: 0.6238
Epoch [9/10], Iter [500/734] Loss: 0.6597
Epoch [9/10], Iter [600/734] Loss: 0.7225
Epoch [9/10], Iter [700/734] Loss: 0.4683
Epoch [10/10], Iter [100/734] Loss: 0.6528
Epoch [10/10], Iter [200/734] Loss: 0.8435
Epoch [10/10], Iter [300/734] Loss: 0.6604
Epoch [10/10], Iter [400/734] Loss: 0.4524
Epoch [10/10], Iter [500/734] Loss: 0.4923
Epoch [10/10], Iter [600/734] Loss: 0.6113
Epoch [10/10], Iter [700/734] Loss: 0.5534

Training set Accuracy of the model on the Train images: 67 %
1511.4266280000002 seconds
Test Accuracy of the model on the test images: 69 %
Accuracy of  bags : 76 %
Accuracy of dresses : 76 %
Accuracy of footwear : 85 %
Accuracy of outerwear : 58 %
Accuracy of skirts : 84 %
Accuracy of  tops : 40 %

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| bags | 0.72 | 0.73 | 0.73 | 4137 |
| dresses | 0.66 | 0.75 | 0.70 | 3770 |
| footwear | 0.82 | 0.84 | 0.83 | 4504 |
| outerwear | 0.53 | 0.58 | 0.55 | 3777 |
| skirts | 0.69 | 0.78 | 0.73 | 3515 |
| tops | 0.73 | 0.44 | 0.55 | 3820 |
| avg / total | 0.70 | 0.69 | 0.69 | 23523 |

{0: 0.8348112843856803, 1: 0.8390889084973965, 2: 0.8982624718461869, 3: 0.740735880230154, 4: 0.8607601269591736, 5: 0.7049256126249677}




**7Layers_bottleneck_4maxpool_10epochs_64batchsize**

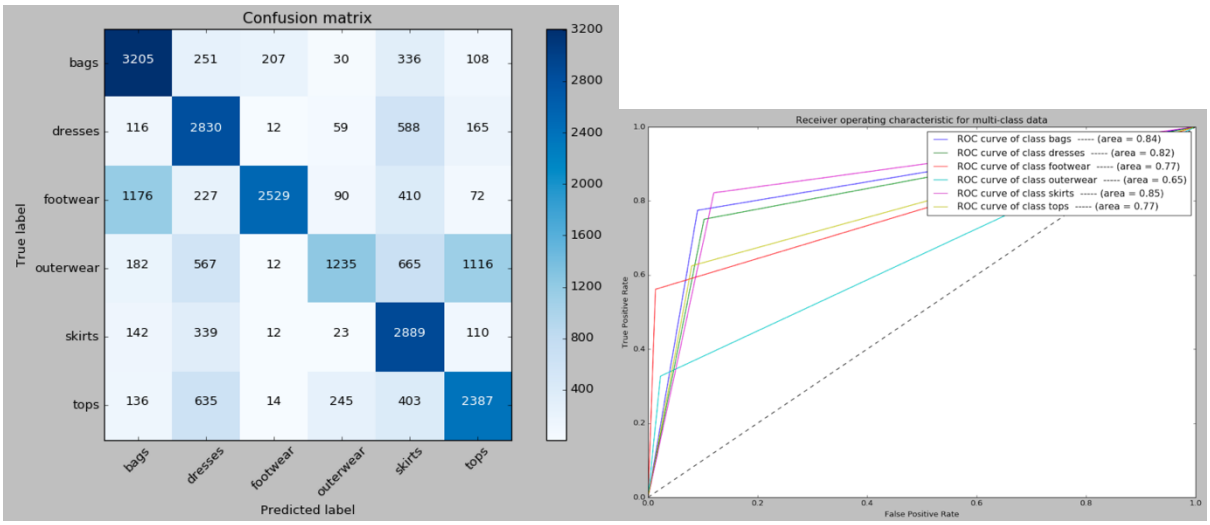
cuda:0
<class 'torch.utils.data.dataloader._DataLoaderIter'>
images shape on batch size = torch.Size([64, 3, 224, 224])
labels shape on batch size = torch.Size([64])
 tops  tops footwear skirts  bags  tops footwear outerwear  bags skirts footwear dresses footwear  bags outerwear dresses skirts outerwear skirts footwear outerwear  tops footwear footwear skirts outerwear dresses outerwear  tops skirts footwear skirts footwear  bags outerwear outerwear  tops skirts skirts outerwear  bags dresses footwear  tops skirts  bags

dresses  bags  bags  bags footwear outerwear skirts outerwear outerwear  tops skirts  bags
bags footwear outerwear  tops outerwear  bags
Epoch [1/10], Iter [100/734] Loss: 1.4676
Epoch [1/10], Iter [200/734] Loss: 1.3276
Epoch [1/10], Iter [300/734] Loss: 1.2785
Epoch [1/10], Iter [400/734] Loss: 1.1935
Epoch [1/10], Iter [500/734] Loss: 1.3077
Epoch [1/10], Iter [600/734] Loss: 0.9357
Epoch [1/10], Iter [700/734] Loss: 1.0591
Epoch [2/10], Iter [100/734] Loss: 0.8426
Epoch [2/10], Iter [200/734] Loss: 0.8500
Epoch [2/10], Iter [300/734] Loss: 0.9319
Epoch [2/10], Iter [400/734] Loss: 1.2197
Epoch [2/10], Iter [500/734] Loss: 0.7788
Epoch [2/10], Iter [600/734] Loss: 0.8853
Epoch [2/10], Iter [700/734] Loss: 0.8381
Epoch [3/10], Iter [100/734] Loss: 0.7493
Epoch [3/10], Iter [200/734] Loss: 0.6120
Epoch [3/10], Iter [300/734] Loss: 0.9029
Epoch [3/10], Iter [400/734] Loss: 0.5365
Epoch [3/10], Iter [500/734] Loss: 0.8527
Epoch [3/10], Iter [600/734] Loss: 0.6645
Epoch [3/10], Iter [700/734] Loss: 0.6812
Epoch [4/10], Iter [100/734] Loss: 0.8217
Epoch [4/10], Iter [200/734] Loss: 0.7340
Epoch [4/10], Iter [300/734] Loss: 0.7476
Epoch [4/10], Iter [400/734] Loss: 0.6601
Epoch [4/10], Iter [500/734] Loss: 0.7787
Epoch [4/10], Iter [600/734] Loss: 0.7153
Epoch [4/10], Iter [700/734] Loss: 0.4962
Epoch [5/10], Iter [100/734] Loss: 0.7229
Epoch [5/10], Iter [200/734] Loss: 0.7583
Epoch [5/10], Iter [300/734] Loss: 0.6273
Epoch [5/10], Iter [400/734] Loss: 0.7988
Epoch [5/10], Iter [500/734] Loss: 0.5132
Epoch [5/10], Iter [600/734] Loss: 0.6681
Epoch [5/10], Iter [700/734] Loss: 0.9181
Epoch [6/10], Iter [100/734] Loss: 0.7092
Epoch [6/10], Iter [200/734] Loss: 0.8273
Epoch [6/10], Iter [300/734] Loss: 0.6527
Epoch [6/10], Iter [400/734] Loss: 0.6090
Epoch [6/10], Iter [500/734] Loss: 0.6638
Epoch [6/10], Iter [600/734] Loss: 0.6300
Epoch [6/10], Iter [700/734] Loss: 0.7598
Epoch [7/10], Iter [100/734] Loss: 0.4893
Epoch [7/10], Iter [200/734] Loss: 0.5426
Epoch [7/10], Iter [300/734] Loss: 0.6837
Epoch [7/10], Iter [400/734] Loss: 0.7651
Epoch [7/10], Iter [500/734] Loss: 0.4926
Epoch [7/10], Iter [600/734] Loss: 0.5789
Epoch [7/10], Iter [700/734] Loss: 0.8050

Epoch [8/10], Iter [100/734] Loss: 0.7930
Epoch [8/10], Iter [200/734] Loss: 0.5370
Epoch [8/10], Iter [300/734] Loss: 0.6571
Epoch [8/10], Iter [400/734] Loss: 0.7404
Epoch [8/10], Iter [500/734] Loss: 0.6290
Epoch [8/10], Iter [600/734] Loss: 0.5522
Epoch [8/10], Iter [700/734] Loss: 0.6183
Epoch [9/10], Iter [100/734] Loss: 0.6639
Epoch [9/10], Iter [200/734] Loss: 0.6806
Epoch [9/10], Iter [300/734] Loss: 0.5304
Epoch [9/10], Iter [400/734] Loss: 0.8879
Epoch [9/10], Iter [500/734] Loss: 0.5506
Epoch [9/10], Iter [600/734] Loss: 0.8253
Epoch [9/10], Iter [700/734] Loss: 0.6097
Epoch [10/10], Iter [100/734] Loss: 0.4011
Epoch [10/10], Iter [200/734] Loss: 0.5939
Epoch [10/10], Iter [300/734] Loss: 0.6401
Epoch [10/10], Iter [400/734] Loss: 0.5383
Epoch [10/10], Iter [500/734] Loss: 0.6387
Epoch [10/10], Iter [600/734] Loss: 0.6728
Epoch [10/10], Iter [700/734] Loss: 0.9676
Training set Accuracy of the model on the Train images: 70 %
1654.217491 seconds
Test Accuracy of the model on the test images: 64 %
Accuracy of  bags : 80 %
Accuracy of dresses : 77 %
Accuracy of footwear : 60 %
Accuracy of outerwear : 27 %
Accuracy of skirts : 85 %
Accuracy of  tops : 60 %

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| bags | 0.65 | 0.77 | 0.70 | 4137 |
| dresses | 0.58 | 0.75 | 0.66 | 3770 |
| footwear | 0.91 | 0.56 | 0.69 | 4504 |
| outerwear | 0.73 | 0.33 | 0.45 | 3777 |
| skirts | 0.55 | 0.82 | 0.66 | 3515 |
| tops | 0.60 | 0.62 | 0.61 | 3820 |
| avg / total | 0.68 | 0.64 | 0.63 | 23523 |

{0: 0.8421707403509646, 1: 0.8242254038972827, 2: 0.7739940425564901, 3: 0.652170793863641, 4: 0.850927068717321, 5: 0.7725675296477296}
('GroundTruth: ', ' bags  bags  bags  bags')
('Predicted: ', 'skirts skirts  bags skirts')




## 5Layers_straight_4maxpool_10epochs_64batchsize_SGD

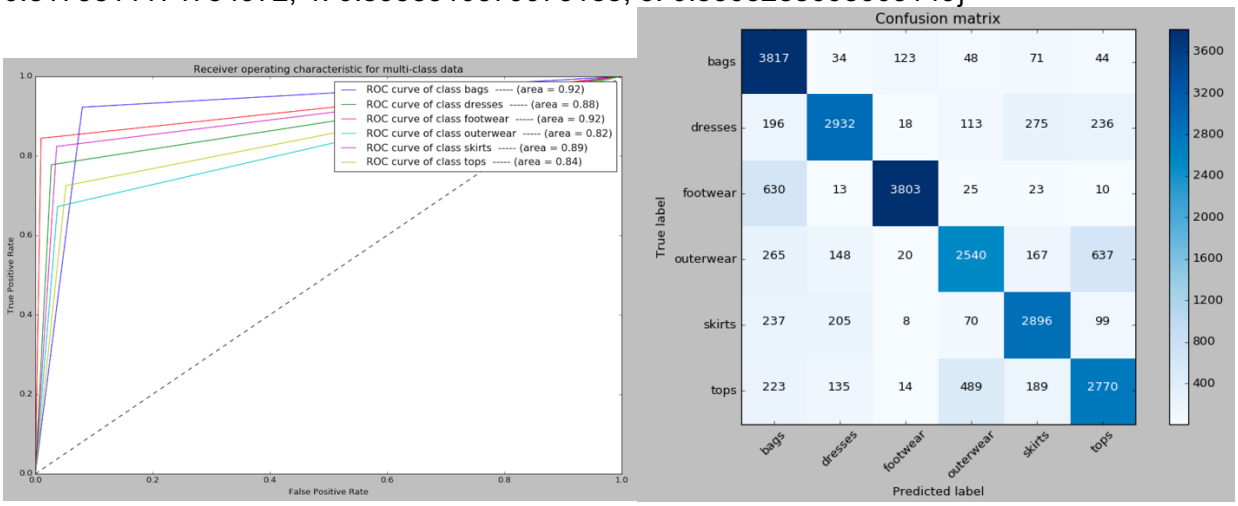
images shape on batch size = torch.Size([64, 3, 224, 224])
labels shape on batch size = torch.Size([64])
tops tops footwear skirts bags tops footwear outerwear bags skirts footwear dresses footwear bags outerwear dresses skirts outerwear skirts footwear outerwear tops footwear footwear skirts outerwear dresses outerwear tops skirts footwear skirts footwear bags outerwear outerwear tops skirts skirts outerwear bags dresses footwear tops skirts bags dresses bags bags bags footwear outerwear skirts outerwear outerwear tops skirts bags bags footwear outerwear tops outerwear bags
Epoch [1/10], Iter [100/734] Loss: 1.2654
Epoch [1/10], Iter [200/734] Loss: 0.9062
Epoch [1/10], Iter [300/734] Loss: 0.9264
Epoch [1/10], Iter [400/734] Loss: 1.0646
Epoch [1/10], Iter [500/734] Loss: 0.7315
Epoch [1/10], Iter [600/734] Loss: 0.7811
Epoch [1/10], Iter [700/734] Loss: 0.6110
Epoch [2/10], Iter [100/734] Loss: 0.6881
Epoch [2/10], Iter [200/734] Loss: 0.5268
Epoch [2/10], Iter [300/734] Loss: 0.6371
Epoch [2/10], Iter [400/734] Loss: 0.8786
Epoch [2/10], Iter [500/734] Loss: 0.4430
Epoch [2/10], Iter [600/734] Loss: 0.7251
Epoch [2/10], Iter [700/734] Loss: 0.4489
Epoch [3/10], Iter [100/734] Loss: 0.5221
Epoch [3/10], Iter [200/734] Loss: 0.4119
Epoch [3/10], Iter [300/734] Loss: 0.4001
Epoch [3/10], Iter [400/734] Loss: 0.5887
Epoch [3/10], Iter [500/734] Loss: 0.4385
Epoch [3/10], Iter [600/734] Loss: 0.7185
Epoch [3/10], Iter [700/734] Loss: 0.6209
Epoch [4/10], Iter [100/734] Loss: 0.4697

Epoch [4/10], Iter [200/734] Loss: 0.6585
Epoch [4/10], Iter [300/734] Loss: 0.6378
Epoch [4/10], Iter [400/734] Loss: 0.4570
Epoch [4/10], Iter [500/734] Loss: 0.5559
Epoch [4/10], Iter [600/734] Loss: 0.3208
Epoch [4/10], Iter [700/734] Loss: 0.4059
Epoch [5/10], Iter [100/734] Loss: 0.7248
Epoch [5/10], Iter [200/734] Loss: 0.5890
Epoch [5/10], Iter [300/734] Loss: 0.6253
Epoch [5/10], Iter [400/734] Loss: 0.6055
Epoch [5/10], Iter [500/734] Loss: 0.6216
Epoch [5/10], Iter [600/734] Loss: 0.2936
Epoch [5/10], Iter [700/734] Loss: 0.4689
Epoch [6/10], Iter [100/734] Loss: 0.3743
Epoch [6/10], Iter [200/734] Loss: 0.4792
Epoch [6/10], Iter [300/734] Loss: 0.3437
Epoch [6/10], Iter [400/734] Loss: 0.4199
Epoch [6/10], Iter [500/734] Loss: 0.5919
Epoch [6/10], Iter [600/734] Loss: 0.6372
Epoch [6/10], Iter [700/734] Loss: 0.4612
Epoch [7/10], Iter [100/734] Loss: 0.5745
Epoch [7/10], Iter [200/734] Loss: 0.3906
Epoch [7/10], Iter [300/734] Loss: 0.4917
Epoch [7/10], Iter [400/734] Loss: 0.4401
Epoch [7/10], Iter [500/734] Loss: 0.2848
Epoch [7/10], Iter [600/734] Loss: 0.4011
Epoch [7/10], Iter [700/734] Loss: 0.5174
Epoch [8/10], Iter [100/734] Loss: 0.5209
Epoch [8/10], Iter [200/734] Loss: 0.3897
Epoch [8/10], Iter [300/734] Loss: 0.6272
Epoch [8/10], Iter [400/734] Loss: 0.3681
Epoch [8/10], Iter [500/734] Loss: 0.4750
Epoch [8/10], Iter [600/734] Loss: 0.3578
Epoch [8/10], Iter [700/734] Loss: 0.3371
Epoch [9/10], Iter [100/734] Loss: 0.4898
Epoch [9/10], Iter [200/734] Loss: 0.6001
Epoch [9/10], Iter [300/734] Loss: 0.4876
Epoch [9/10], Iter [400/734] Loss: 0.4811
Epoch [9/10], Iter [500/734] Loss: 0.3513
Epoch [9/10], Iter [600/734] Loss: 0.3209
Epoch [9/10], Iter [700/734] Loss: 0.3917
Epoch [10/10], Iter [100/734] Loss: 0.4042
Epoch [10/10], Iter [200/734] Loss: 0.4195
Epoch [10/10], Iter [300/734] Loss: 0.3467
Epoch [10/10], Iter [400/734] Loss: 0.5148
Epoch [10/10], Iter [500/734] Loss: 0.3263
Epoch [10/10], Iter [600/734] Loss: 0.3237
Epoch [10/10], Iter [700/734] Loss: 0.5263
Training set Accuracy of the model on the Train images: 80 %
1542.561696 seconds
Test Accuracy of the model on the test images: 79 %

Accuracy of  bags : 94 %
Accuracy of dresses : 78 %
Accuracy of footwear : 86 %
Accuracy of outerwear : 69 %
Accuracy of skirts : 86 %
Accuracy of  tops : 73 %
Confusion matrix

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| bags | 0.71 | 0.92 | 0.80 | 4137 |
| dresses | 0.85 | 0.78 | 0.81 | 3770 |
| footwear | 0.95 | 0.84 | 0.90 | 4504 |
| outerwear | 0.77 | 0.67 | 0.72 | 3777 |
| skirts | 0.80 | 0.82 | 0.81 | 3515 |
| tops | 0.73 | 0.73 | 0.73 | 3820 |
| avg / total | 0.81 | 0.80 | 0.80 | 23523 |

{0: 0.9213215363583702, 1: 0.8753171696982939, 2: 0.9173693056966404, 3: 0.8173811174754972, 4: 0.8938310379973188, 5: 0.8365288008603149}




images shape on batch size = torch.Size([64, 3, 224, 224])
labels shape on batch size = torch.Size([64])
Epoch [1/5], Iter [100/734] Loss: 1.6147
Epoch [1/5], Iter [200/734] Loss: 1.3458
Epoch [1/5], Iter [300/734] Loss: 1.6286
Epoch [1/5], Iter [400/734] Loss: 1.4943
Epoch [1/5], Iter [500/734] Loss: 1.6882
Epoch [1/5], Iter [600/734] Loss: 1.3587
Epoch [1/5], Iter [700/734] Loss: 1.8820
Epoch [2/5], Iter [100/734] Loss: 1.2920
Epoch [2/5], Iter [200/734] Loss: 1.2436
Epoch [2/5], Iter [300/734] Loss: 1.2560
Epoch [2/5], Iter [400/734] Loss: 1.1185
Epoch [2/5], Iter [500/734] Loss: 1.1285

Epoch [2/5], Iter [600/734] Loss: 1.5031
Epoch [2/5], Iter [700/734] Loss: 1.1962
Epoch [3/5], Iter [100/734] Loss: 1.2916
Epoch [3/5], Iter [200/734] Loss: 0.9956
Epoch [3/5], Iter [300/734] Loss: 1.3433
Epoch [3/5], Iter [400/734] Loss: 1.0013
Epoch [3/5], Iter [500/734] Loss: 1.1175
Epoch [3/5], Iter [600/734] Loss: 0.9673
Epoch [3/5], Iter [700/734] Loss: 1.3639
Epoch [4/5], Iter [100/734] Loss: 1.6038
Epoch [4/5], Iter [200/734] Loss: 1.1203
Epoch [4/5], Iter [300/734] Loss: 1.0693
Epoch [4/5], Iter [400/734] Loss: 1.1472
Epoch [4/5], Iter [500/734] Loss: 1.5915
Epoch [4/5], Iter [600/734] Loss: 0.9166
Epoch [4/5], Iter [700/734] Loss: 1.0970
Epoch [5/5], Iter [100/734] Loss: 1.3605
Epoch [5/5], Iter [200/734] Loss: 1.0445
Epoch [5/5], Iter [300/734] Loss: 1.2392
Epoch [5/5], Iter [400/734] Loss: 1.1143
Epoch [5/5], Iter [500/734] Loss: 1.4075
Epoch [5/5], Iter [600/734] Loss: 1.1985
Epoch [5/5], Iter [700/734] Loss: 1.4106
1025.029345 seconds
Test Accuracy of the model on the test images: 57 %
Accuracy of  bags : 47 %
Accuracy of dresses : 59 %
Accuracy of footwear : 77 %
Accuracy of outerwear : 49 %
Accuracy of skirts : 67 %
Accuracy of  tops : 35 %

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| bags | 0.64 | 0.47 | 0.54 | 4137 |
| dresses | 0.55 | 0.61 | 0.58 | 3770 |
| footwear | 0.66 | 0.75 | 0.70 | 4504 |
| outerwear | 0.45 | 0.51 | 0.48 | 3777 |
| skirts | 0.62 | 0.68 | 0.65 | 3515 |
| tops | 0.52 | 0.41 | 0.46 | 3820 |
| avg / total | 0.58 | 0.57 | 0.57 | 23523 |

{0: 0.705917890253255, 1: 0.7575016439768542, 2: 0.8302203662468223, 3:
0.6959040859423011, 4: 0.8032823997001485, 5: 0.6680424991755846}

images shape on batch size = torch.Size([64, 3, 224, 224])
labels shape on batch size = torch.Size([64])
Epoch [1/10], Iter [100/734] Loss: 1.9546
Epoch [1/10], Iter [200/734] Loss: 1.2948

Epoch [1/10], Iter [300/734] Loss: 1.3810
Epoch [1/10], Iter [400/734] Loss: 1.0642
Epoch [1/10], Iter [500/734] Loss: 1.0350
Epoch [1/10], Iter [600/734] Loss: 0.8895
Epoch [1/10], Iter [700/734] Loss: 1.0499
Epoch [2/10], Iter [100/734] Loss: 0.6745
Epoch [2/10], Iter [200/734] Loss: 0.8023
Epoch [2/10], Iter [300/734] Loss: 0.9847
Epoch [2/10], Iter [400/734] Loss: 0.7109
Epoch [2/10], Iter [500/734] Loss: 0.7118
Epoch [2/10], Iter [600/734] Loss: 0.6310
Epoch [2/10], Iter [700/734] Loss: 0.7367
Epoch [3/10], Iter [100/734] Loss: 0.6599
Epoch [3/10], Iter [200/734] Loss: 0.7837
Epoch [3/10], Iter [300/734] Loss: 0.3868
Epoch [3/10], Iter [400/734] Loss: 0.5837
Epoch [3/10], Iter [500/734] Loss: 0.7627
Epoch [3/10], Iter [600/734] Loss: 0.7908
Epoch [3/10], Iter [700/734] Loss: 0.7715
Epoch [4/10], Iter [100/734] Loss: 0.6846
Epoch [4/10], Iter [200/734] Loss: 0.3889
Epoch [4/10], Iter [300/734] Loss: 0.3653
Epoch [4/10], Iter [400/734] Loss: 0.7146
Epoch [4/10], Iter [500/734] Loss: 0.5413
Epoch [4/10], Iter [600/734] Loss: 0.7096
Epoch [4/10], Iter [700/734] Loss: 0.3656
Epoch [5/10], Iter [100/734] Loss: 0.4213
Epoch [5/10], Iter [200/734] Loss: 0.5637
Epoch [5/10], Iter [300/734] Loss: 0.8980
Epoch [5/10], Iter [400/734] Loss: 0.5166
Epoch [5/10], Iter [500/734] Loss: 0.5732
Epoch [5/10], Iter [600/734] Loss: 0.7303
Epoch [5/10], Iter [700/734] Loss: 0.6087
Epoch [6/10], Iter [100/734] Loss: 0.7945
Epoch [6/10], Iter [200/734] Loss: 0.6837
Epoch [6/10], Iter [300/734] Loss: 0.5301
Epoch [6/10], Iter [400/734] Loss: 0.5391
Epoch [6/10], Iter [500/734] Loss: 0.4117
Epoch [6/10], Iter [600/734] Loss: 0.5727
Epoch [6/10], Iter [700/734] Loss: 0.7116
Epoch [7/10], Iter [100/734] Loss: 0.4469
Epoch [7/10], Iter [200/734] Loss: 0.4901
Epoch [7/10], Iter [300/734] Loss: 0.5236
Epoch [7/10], Iter [400/734] Loss: 0.5761
Epoch [7/10], Iter [500/734] Loss: 0.2883
Epoch [7/10], Iter [600/734] Loss: 0.4194
Epoch [7/10], Iter [700/734] Loss: 0.6461
Epoch [8/10], Iter [100/734] Loss: 0.4778
Epoch [8/10], Iter [200/734] Loss: 0.8473
Epoch [8/10], Iter [300/734] Loss: 0.5763
Epoch [8/10], Iter [400/734] Loss: 0.5120

Epoch [8/10], Iter [500/734] Loss: 0.4834
Epoch [8/10], Iter [600/734] Loss: 0.5395
Epoch [8/10], Iter [700/734] Loss: 0.4784
Epoch [9/10], Iter [100/734] Loss: 0.4852
Epoch [9/10], Iter [200/734] Loss: 0.4749
Epoch [9/10], Iter [300/734] Loss: 0.4621
Epoch [9/10], Iter [400/734] Loss: 0.3594
Epoch [9/10], Iter [500/734] Loss: 0.4487
Epoch [9/10], Iter [600/734] Loss: 0.6004
Epoch [9/10], Iter [700/734] Loss: 0.4709
Epoch [10/10], Iter [100/734] Loss: 0.5118
Epoch [10/10], Iter [200/734] Loss: 0.5617
Epoch [10/10], Iter [300/734] Loss: 0.3650
Epoch [10/10], Iter [400/734] Loss: 0.3906
Epoch [10/10], Iter [500/734] Loss: 0.4994
Epoch [10/10], Iter [600/734] Loss: 0.6473
Epoch [10/10], Iter [700/734] Loss: 0.5782
972.239551 seconds
Test Accuracy of the model on the test images: 81 %
Accuracy of  bags : 86 %
Accuracy of dresses : 76 %
Accuracy of footwear : 95 %
Accuracy of outerwear : 74 %
Accuracy of skirts : 89 %
Accuracy of  tops : 69 %

**Summary and conclusions.**

When observing the results of our experiments, we found that footwear, bags and skirts consistently performed the best in all of the experiments we conducted out of the categories we chose. In conclusion, the models that we ran performed better than random guess; however, there are ways this model could be improved going forward. One way to do this would be to simply gather more data and possessing more processing power. We could also try tuning the hyper-parameter differently.

**Percentage of the code that you found or copied from the internet.**

All the codes combined

(274 - 46)/ ( 274 + 180) * 100 = 39 %

**References.**


https://stackoverflow.com/questions/47850280/fastest-way-to-compute-image-dataset-channel-wise-mean-and-standard-deviation-in
http://acberg.com/papers/wheretobuyit2015iccv.pdf
https://github.com/flipkart-incubator/fk-visual-search
https://github.com/Airconaaron/blog_post_visualizing_pytorch_cnn/blob/master/Visualizing%20Learned%20Filters%20in%20PyTorch.ipynb
https://arxiv.org/pdf/1311.2901.pdf
https://stackoverflow.com/questions/47850280/fastest-way-to-compute-image-dataset-channel-wise-mean-and-standard-deviation-in
https://github.com/amir-jafari/Deep-Learning/blob/master/Pytorch_/6-Conv_Mnist/Conv_Mnist.py
https://discuss.pytorch.org/t/data-augmentation-in-pytorch/7925
https://stackoverflow.com/questions/2301789/read-a-file-in-reverse-order-using-python
https://pytorch.org/docs/stable/torchvision/transforms.html
https://stackoverflow.com/questions/12984426/python-pil-ioerror-image-file-truncated-with-big-images
http://www.apsipa.org/proceedings/2017/CONTENTS/papers2017/14DecThursday/Poster%204/TP-P4.14.pdf
https://www.quora.com/How-can-I-calculate-the-size-of-output-of-convolutional-layer
https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148