# Assignment - 2

Elkhan Ismayilzada

**Problem 1**

Following results have been obtained after running the “normal_train.py” script.

```
extend model layers as a incremental classes in each task
Epoch [1/3]: 100%|
Epoch [1/3] training - Avg Loss: 0.087532, Accuracy: 97.212054%
Epoch [2/3]: 100%|
Epoch [2/3] training - Avg Loss: 0.026805, Accuracy: 99.173095%
Epoch [3/3]: 100%|
Epoch [3/3] training - Avg Loss: 0.021154, Accuracy: 99.307099%

 ----------------------evaluation start----------------------
100%|
\Test Classes: [0, 1, 2, 3, 4]   Test Accuracy: 99.863787%
seen classes : [0, 1, 2, 3, 4]   seen classes acc : 99.863787
--------------------------------------------------

extend model layers as a incremental classes in each task
Epoch [1/3]: 100%|
Epoch [1/3] training - Avg Loss: 0.137635, Accuracy: 95.827098%
Epoch [2/3]: 100%|
Epoch [2/3] training - Avg Loss: 0.040096, Accuracy: 98.809686%
Epoch [3/3]: 100%|
Epoch [3/3] training - Avg Loss: 0.030236, Accuracy: 99.051150%

 ----------------------evaluation start----------------------
100%|
\Test Classes: [0, 1, 2, 3, 4]   Test Accuracy: 0.000000%
100%|
\Test Classes: [5, 6, 7, 8, 9]   Test Accuracy: 99.423987%
seen classes : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]    seen classes acc : 49.711993
--------------------------------------------------
```

In the first task we train to predict only the first 5 digits of 10 digits (0, 1, 2, 3, 4) and we get an accuracy of 99.05%. In the second task we change the last layer of the model to predict 10 classes rather than 5 but we train to predict only the last 5 digits of 10 digits (5, 6, 7, 8, 9). As we train on only the new task, the model loses its information on predicting the first 5 digits. Therefore, we get zero accuracy on the old task while getting 99.42% accuracy on the new task.

**Problem 2**

Following results have been obtained after running the “Prob2.py” script.

```
extend model layers as a incremental classes in each task
Epoch [1/3]: 100%|
Epoch [1/3] training - Avg Loss: 0.087532, Accuracy: 97.212054%
Epoch [2/3]: 100%|
Epoch [2/3] training - Avg Loss: 0.026805, Accuracy: 99.173095%
Epoch [3/3]: 100%|
Epoch [3/3] training - Avg Loss: 0.021154, Accuracy: 99.307099%

 ----------------------evaluation start----------------------
100%|
\Test Classes: [0, 1, 2, 3, 4]   Test Accuracy: 99.863787%
seen classes : [0, 1, 2, 3, 4]   seen classes acc : 99.863787
-------------------------------------------------

extend model layers as a incremental classes in each task
Epoch [1/3]: 100%|
Epoch [1/3] training - Avg Loss: 6.521263, Accuracy: 0.561148%
Epoch [2/3]: 100%|
Epoch [2/3] training - Avg Loss: 6.560232, Accuracy: 0.591756%
Epoch [3/3]: 100%|
Epoch [3/3] training - Avg Loss: 6.572091, Accuracy: 0.510135%

 ----------------------evaluation start----------------------
100%|
\Test Classes: [0, 1, 2, 3, 4]   Test Accuracy: 99.805410%
100%|
\Test Classes: [5, 6, 7, 8, 9]   Test Accuracy: 0.000000%
seen classes : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]    seen classes acc : 49.902705
-------------------------------------------------
```

As can be seen from the results, unlike Problem 1, our model retains the information on the old task but fails to train on the new task. This is because we freeze all the model parameters except the last layer and since the model parameters are trained on the old task, it does not generate useful features for the new task to the last layer for predictions. Therefore, the last layer cannot learn anything on the new task.

**How to run**

To train from scratch:

“python Prob2.py”

To load checkpoint and test:

“python Prob2.py –checkpoint Prob2.pth

**Problem 3**

Following results have been obtained after running the “Prob3.py” script:

```
extend model layers as a incremental classes in each task
Epoch [1/3]: 100%|
Epoch [1/3] training - Avg Loss: 0.087532, Accuracy: 97.212054%
Epoch [2/3]: 100%|
Epoch [2/3] training - Avg Loss: 0.026805, Accuracy: 99.173095%
Epoch [3/3]: 100%|
Epoch [3/3] training - Avg Loss: 0.021154, Accuracy: 99.307099%

 ----------------------evaluation start----------------------
100%|
\Test Classes: [0, 1, 2, 3, 4]   Test Accuracy: 99.863787%
seen classes : [0, 1, 2, 3, 4]   seen classes acc : 99.863787
-------------------------------------------------

extend model layers as a incremental classes in each task
Epoch [1/3]: 100%|
Epoch [1/3] training - Avg Loss: 0.119296, Accuracy: 96.385000%
Epoch [2/3]: 100%|
Epoch [2/3] training - Avg Loss: 0.056372, Accuracy: 98.346667%
Epoch [3/3]: 100%|
Epoch [3/3] training - Avg Loss: 0.044468, Accuracy: 98.705000%

 ----------------------evaluation start----------------------
100%|
\Test Classes: [0, 1, 2, 3, 4]   Test Accuracy: 99.532983%
100%|
\Test Classes: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]    Test Accuracy: 99.280000%
seen classes : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]    seen classes acc : 99.406492
```

This time, after training to predict the first 5 digits of 10 digits we train to predict all 10 digits on the second task. As can be seen from the results, our accuracy on the first task as well as the second task is around 99%. This is because since the second task contains the data of the first task, the model does not forget the old task and adapts easily to the second task.

**How to run**

To train from scratch:

“python Prob3.py”

To load checkpoint and test:

“python Prob3.py –checkpoint Prob3.pth

**Problem 4**

Following results have been obtained after running the “Prob4.py” script:

```
extend model layers as a incremental classes in each task
Epoch [1/3]: 100%|
Epoch [1/3] training - Avg Loss: 0.087532, Accuracy: 97.212054%
Epoch [2/3]: 100%|
Epoch [2/3] training - Avg Loss: 0.026805, Accuracy: 99.173095%
Epoch [3/3]: 100%|
Epoch [3/3] training - Avg Loss: 0.021154, Accuracy: 99.307099%

 ----------------------evaluation start----------------------
100%|
\Test Classes: [0, 1, 2, 3, 4]   Test Accuracy: 99.863787%
seen classes : [0, 1, 2, 3, 4]   seen classes acc : 99.863787
-------------------------------------------------

extend model layers as a incremental classes in each task
Epoch [1/3]: 100%|
Epoch [1/3] training - Avg Loss: 0.166975, Accuracy: 95.211343%
Epoch [2/3]: 100%|
Epoch [2/3] training - Avg Loss: 0.056774, Accuracy: 98.297887%
Epoch [3/3]: 100%|
Epoch [3/3] training - Avg Loss: 0.041700, Accuracy: 98.752675%

 ----------------------evaluation start----------------------
100%|
\Test Classes: [0, 1, 2, 3, 4]   Test Accuracy: 86.884608%
100%|
\Test Classes: [5, 6, 7, 8, 9]   Test Accuracy: 99.485703%
seen classes : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]    seen classes acc : 93.185155
-------------------------------------------------
```

This time, unlike Problem 3, instead of using all data of the old task, we sample 500 images from the old task data and combine with the data of the new task. As a result, the model retains information about the old task but does lose some information from around 99% accuracy to around 87%. This is because although our new task data contains some data from the old task, the distribution of the data from the old task is very minor. Therefore, the model adapts to the new task quickly while losing some information on the old task.

**How to run**

To train from scratch:

“python Prob4.py”

To load checkpoint and test:

“python Prob4.py –checkpoint Prob4.pth

**Problem 5**

Following results have been obtained after running the “Prob5.py” script:

```
extend model layers as a incremental classes in each task
Epoch [1/3]: 100%|
Epoch [1/3] training - Avg Loss: 0.087532, Accuracy: 97.212054%
Epoch [2/3]: 100%|
Epoch [2/3] training - Avg Loss: 0.026805, Accuracy: 99.173095%
Epoch [3/3]: 100%|
Epoch [3/3] training - Avg Loss: 0.021154, Accuracy: 99.307099%

 ----------------------evaluation start----------------------
100%|
\Test Classes: [0, 1, 2, 3, 4]   Test Accuracy: 99.863787%
seen classes : [0, 1, 2, 3, 4]   seen classes acc : 99.863787
-------------------------------------------------

100%|
extend model layers as a incremental classes in each task
Epoch [1/3]: 100%|
Epoch [1/3] training - Avg Loss: 1.542880, Accuracy: 65.321045%
Epoch [2/3]: 100%|
Epoch [2/3] training - Avg Loss: 1.533244, Accuracy: 85.638008%
Epoch [3/3]: 100%|
Epoch [3/3] training - Avg Loss: 1.531157, Accuracy: 89.447014%

 ----------------------evaluation start----------------------
100%|
\Test Classes: [0, 1, 2, 3, 4]   Test Accuracy: 84.354933%
100%|
\Test Classes: [5, 6, 7, 8, 9]   Test Accuracy: 97.119934%
seen classes : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]    seen classes acc : 90.737434
-------------------------------------------------
```

This time, I use the LwF (learning without forgetting) approach. First, we train on the old task as usual. Second, before changing the last layer of the model, we predict the model on the new task data and collect the outputs. Third, while training the model on the new task, we consider following two loss functions:

$$\mathcal{L}_{new}(\mathbf{y}_n, \hat{\mathbf{y}}_n) = -\mathbf{y}_n \cdot \log \hat{\mathbf{y}}_n,$$

Where $\hat{\mathbf{y}}_n$ is the softmax output of the model and $\mathbf{y}_n$ is the ground truth. This is the loss of model on the new task.

$$\mathcal{L}_{old}(\mathbf{y}_o, \hat{\mathbf{y}}_o) = -H(\mathbf{y}'_o, \hat{\mathbf{y}}'_o)$$

$$= -\sum_{i=1}^{l} y_o'^{(i)} \log \hat{y}_o'^{(i)},$$

Where $l$ is the number of labels and $y_o'^{(i)}, \hat{y}_o'^{(i)}$ are the modified versions of recorded and current probabilities as follows:

$$y_o'^{(i)} = \frac{\left(y_o^{(i)}\right)^{1/T}}{\sum_j \left(y_o^{(j)}\right)^{1/T}}, \qquad \hat{y}_o'^{(i)} = \frac{\left(\hat{y}_o^{(i)}\right)^{1/T}}{\sum_j \left(\hat{y}_o^{(j)}\right)^{1/T}}.$$

Where T is the temperature and is set to 2 in our experiment. Our total loss is the sum of these two losses with their corresponding lambda parameters for balancing the information loss between the old task and the new task. For the picture above, I used 0.01 as a lambda parameter for new task loss and 0.99 for old task loss. Here is the effect of various lambda parameters on the model:

| Lmbd_new / Lmbd_old | Old task (%) | New task (%) |
|---|---|---|
| 0.005 / 0.995 | 91.79 | 94.75 |
| 0.01 / 0.99 | 84.35 | 97.12 |
| 0.05 / 0.95 | 61.27 | 99.05 |
| 0.1 / 0.9 | 46.94 | 99.24 |
| 0.5 / 0.5 | 3.78 | 99.47 |

As can be seen from the table, as we increase the lambda parameter for the new task, the accuracy on the old task degrades significantly. On the other hand, increasing the lambda parameter for the old task does not significantly affect the performance of the model on the new task. This implies that having a much larger lambda parameter for the old task is preferable.

**How to run**

To train from scratch:

“python Prob5.py”
To load checkpoint and test:
“python Prob5.py –checkpoint Prob5.pth