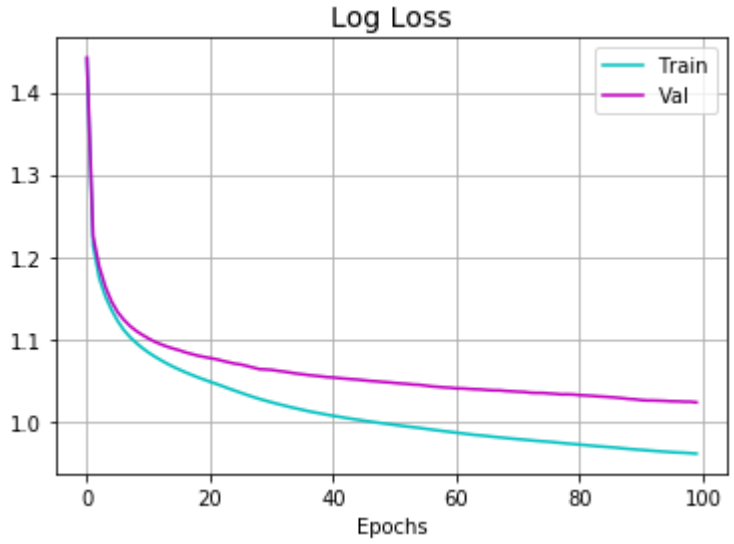
# Homework 3– Programming

## Multilayer Perceptron

1. Please train a MLP classifier with *hidden_layer_sizes=(16, 16, 16)* and *n_epochs=100*.
   a. (5%) Report the plot of log loss curves. How do the curves behave when the number of epochs increases? What is the difference between both curves and why?
   b. (5%) Report the plot of accuracy curves. How do the curves behave when the number of epochs increases? What is the difference between both curves and why?
   c. (5%) Report the scores of recalls, accuracy, and UAR. What do you observe and why? Please analyze it with the number of samples in each class. Hint: Imbalanced data.
   d. (5%) Report the confusion matrix. What do you observe and why?

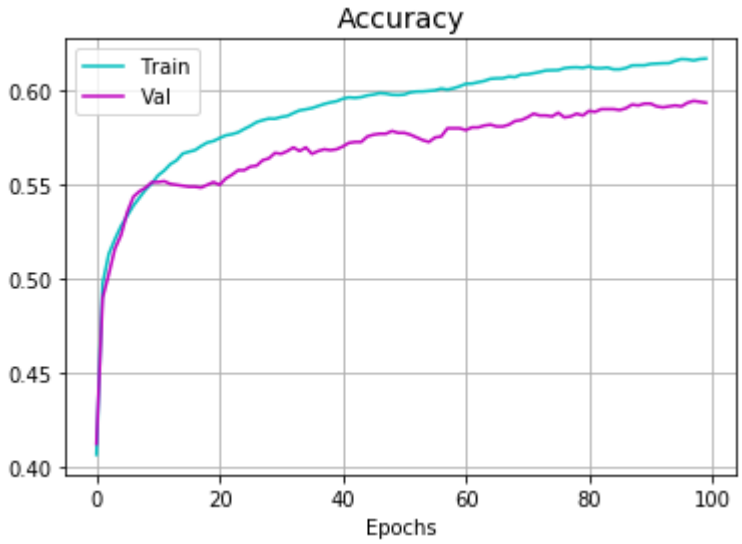
Ans:

a. Log loss decreases when the number of epochs increases. In the beginning, log loss of the training/validation sets decreased a lot and their curves are close. However, with an increase of epochs, log loss of the training set is (smaller than)/(separable to) the validation set since the classifier is to model the behavior of the training set substantially rather than the validation set (validation set has bigger loss).




b. Accuracy increases when the number of epochs increases. In the beginning, accuracy of the training/validation sets increase a lot and their curves are close. However, with an increase of epochs, accuracy of the training set is bigger than the validation set since the classifier is to model the data in training set substantially rather than the validation set

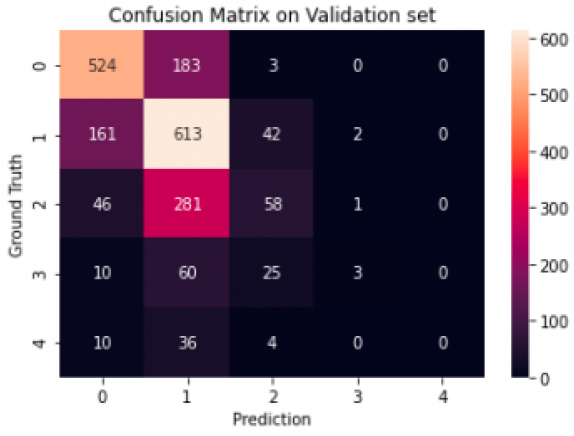
(validation set has smaller accuracy).



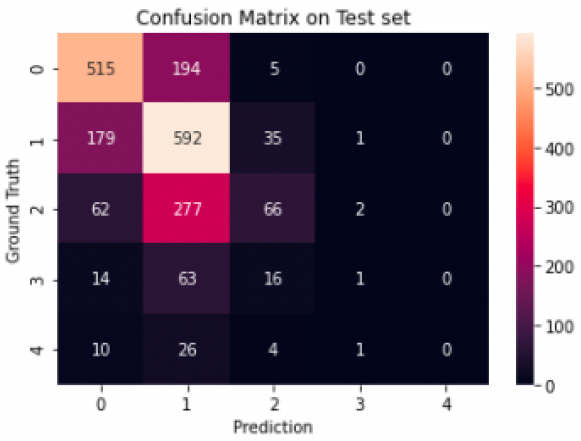

c. From all dataset samples (20621) we have, the numbers of samples of each class are [class1, class2, class3, class4, class5] = [7064, 8195, 3855, 1031, 476]. This data imbalance causes the large difference of recalls of each class. It implies that the majority could dominate the prediction of the model, thus resulting in low accuracy and UAR (or the large difference between accuracy and UAR). And the difference of recalls/accuracy/UAR of the two set is simply results from the randomness of data split.

```
======== Validation Set ========
Loss: 1.025
Recalls: [0.738 0.749 0.15  0.031 0.   ]
Accuracy 0.581
UAR: 0.334
======== Testing Set ========
Recalls: [0.721 0.734 0.162 0.011 0.   ]
Accuracy 0.569
UAR: 0.326
```

d. Confusion matrix on validation set:



Confusion matrix on testing set:




From plots of confusion matrix above, we can observe that the data imbalance ([class1, class2, class3, class4, class5] = [515+194+5, 179+592+35+1, 62+277+66+2, …, …] = [714, 807, 407, 94, 41], testing set for example). This allows easy identification of confusion between classes. We can also see that the class with larger samples is commonly mislabelled as the others.

2. Please train 9 MLP classifiers according to the combination of the following hyperparameters.
    - *hidden_layer_sizes*
        - *(16, 16, 16)*
        - *(16, 16, 16, 16)*
        - *(16, 16, 16, 16, 16)*
    - *n_epochs*
        - *100*
        - *200*
        - *400*

    a. (10%) Report the process times of training, as well as losses, accuracies, and UAR on the validation set, of these 9 classifiers, respectively. Please choose a best performing classifier and a worst one. On which criteria do you choose them? What and why result in these two best and worst classifiers?

    b. (10%) Do any classifiers overfit during the training? If yes, please list the classifiers and report their plots of log loss curves and plots of accuracy curves. On which criteria do you assume these models are overfitted?

Ans:

a.

| Epochs/hidden layers | (16, 16, 16) | (16, 16, 16, 16) | (16, 16, 16, 16, 16) |
|---|---|---|---|
| Epoch =100 | Time: 7 (s)<br>Loss: 1.025<br>Acc: 0.581<br>UAR: 0.334 | Time: 9 (s)<br>Loss: 1.009<br>Acc: 0.598<br>UAR: 0.35 | Time: 10 (s)<br>Loss: 1.035<br>Acc: 0.594<br>UAR: 0.347 |
| Epoch =200 | Time: 14 (s)<br>Loss: 1.027<br>Acc: 0.594<br>UAR: 0.342 | Time: 18 (s)<br>Loss: 1.012<br>Acc: 0.612<br>UAR: 0.37 | Time: 20 (s)<br>Loss: 1.049<br>Acc: 0.591<br>UAR: 0.353 |
| Epoch =400 | Time: 31 (s)<br>Loss: 1.029<br>Acc: 0.592<br>UAR: 0.345 | Time: 34 (s)<br>Loss: 0.993<br>Acc: 0.613<br>UAR: 0.378 | Time: 40 (s)<br>Loss: 1.101<br>Acc: 0.598<br>UAR: 0.362 |

According to the accuracy of the two classifiers, I think that the classifier with (hidden layers, epochs) = ((16, 16, 16, 16), 400) performs the best while the classifier with (hidden layers, epochs) = ((16, 16, 16), 100) performs the worst.

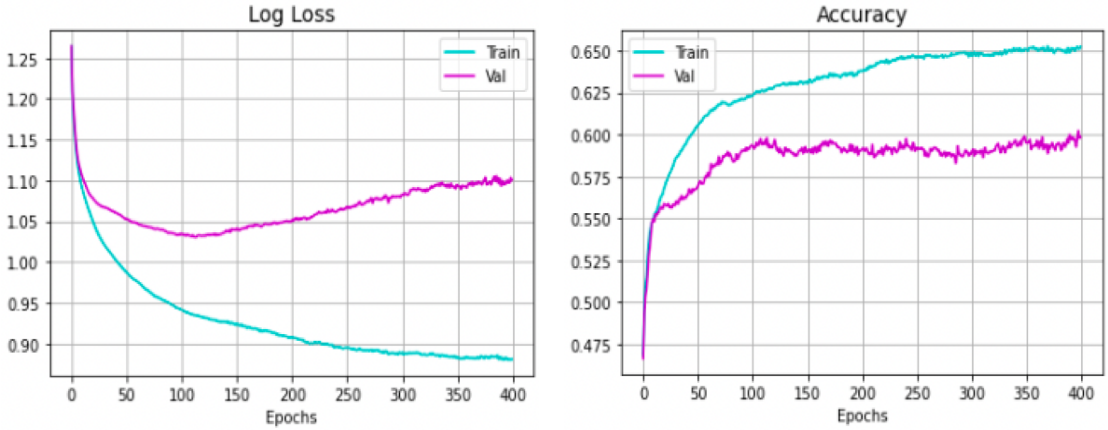
The classifier with (hidden layers, epochs) = ((16, 16, 16, 16), 400) performing the best is because it has adequate hidden layers and epochs, it wouldn't be overfitting. While the classifier with (hidden layers, epochs) = ((16,

16, 16), 100) has the least hidden layers and epochs, it is insufficient for this classifier to update its parameters well.

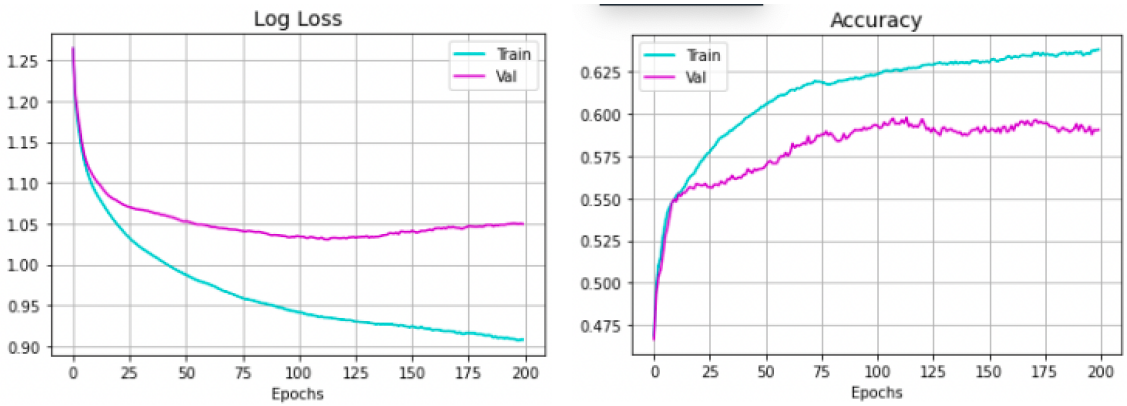
b.

Yes, I think the classifiers with (hidden layers, epochs) = ((16, 16, 16, 16, 16), 400) & ((16, 16, 16, 16, 16), 200) overfit during training process. If the log loss of the validation set stops decreasing and even starts increasing, we can speculate that the model is to "memorize the behavior of training set" substantially, resulting overfitting situation. The plots of the two classifiers are shown as below:

The classifier with (hidden layers, epochs) = ((16, 16, 16, 16, 16), 400):




The classifier with (hidden layers, epochs) = ((16, 16, 16, 16, 16), 200):



3. (10%) According to problem 2. (a), please evaluate the best performing classifier on the testing set. Report the corresponding recalls, accuracy, and UAR. Are these scores as competitive as those evaluated on the validation set? Why do we have to do such comparisons to find the best performing classifier?

| Epochs/hidden layers | (16, 16, 16) | (16, 16, 16, 16) | (16, 16, 16, 16, 16) |
|---|---|---|---|
| Epoch =100 | Recalls: [0.738, 0.749, 0.15, 0.031, 0]<br>Acc: 0.569<br>UAR: 0.326 | Recalls: [0.724, 0.709, 0.204, 0.011, 0]<br>Acc: 0.569<br>UAR: 0.329 | Recalls: [0.72, 0.727, 0.219, 0.011, 0]<br>Acc: 0.577<br>UAR: 0.335 |
| Epoch =200 | Recalls: [0.721, 0.726, 0.187, 0.021, 0]<br>Acc: 0.571<br>UAR: 0.331 | Recalls: [0.742, 0.703, 0.231, 0.011, 0.024]<br>Acc: 0.578<br>UAR: 0.342 | Recalls: [0.697, 0.719, 0.258, 0.043, 0]<br>Acc: 0.575<br>UAR: 0.343 |
| Epoch =400 | Recalls: [0.723, 0.724, 0.209, 0.011, 0]<br>Acc: 0.575<br>UAR: 0.333 | Recalls: [0.707, 0.745, 0.268, 0.064, 0.122]<br>Acc: 0.594<br>UAR: 0.381 | Recalls: [0.727, 0.714, 0.241, 0.021, 0.049]<br>Acc: 0.58<br>UAR: 0.35 |

The classifier with (hidden layers, epochs) = ((16, 16, 16, 16), 400) performs the best on the testing set.

The scores in training set are a little competitive as those evaluated on the validation set in this case. The reason why we have to do such comparisons to find the best classifier is that the data in validation set is the answer seen, while the data in testing set is unseen. If we don't make comparisons with each other, we couldn't know the actual performance in testing set, resulting in an useless classifier probably.