Name(s): Won Woo Lyu
NetID(s): wlyu2
Team name on Kaggle leaderboard: Mercy_Plz

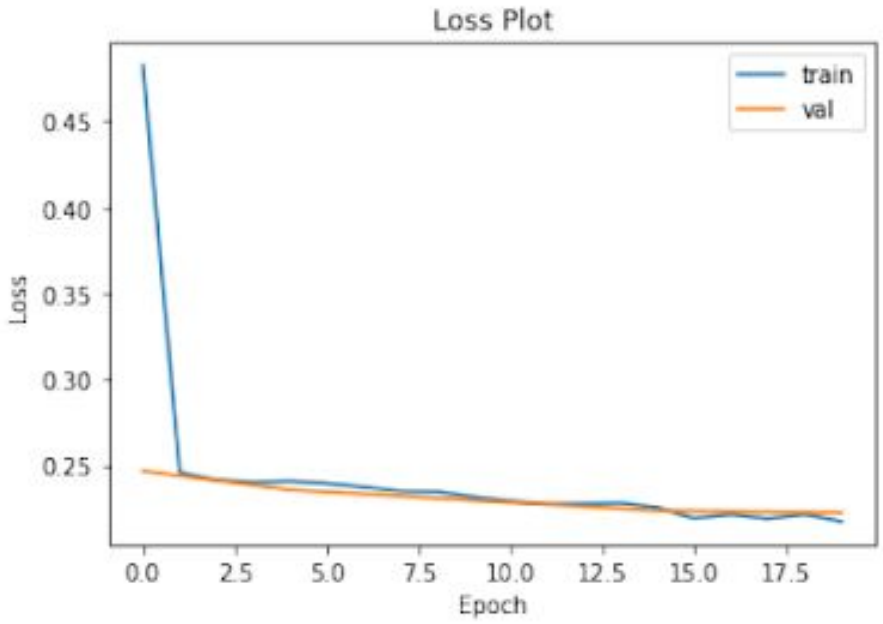
Part-1A: Pre-designed network for multi-label classification

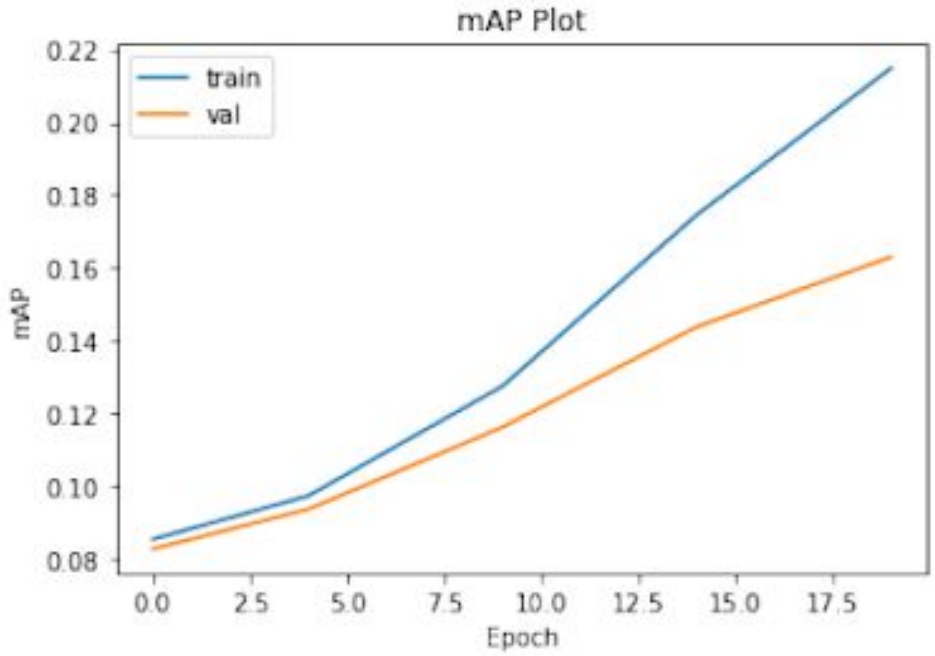
In this part, you will practice to train a neural network both by training from scratch or fine-tuning.

MP3_P1_Introduction.ipynb in your assignment3_p1_starterkit should provide you with enough instruction to start with.

We are asking you to provide the following results.

1. Simple Classifier
    a. Report test mAP for simple classifier: **0.1612**
    b. Visualize loss and mAP plots:

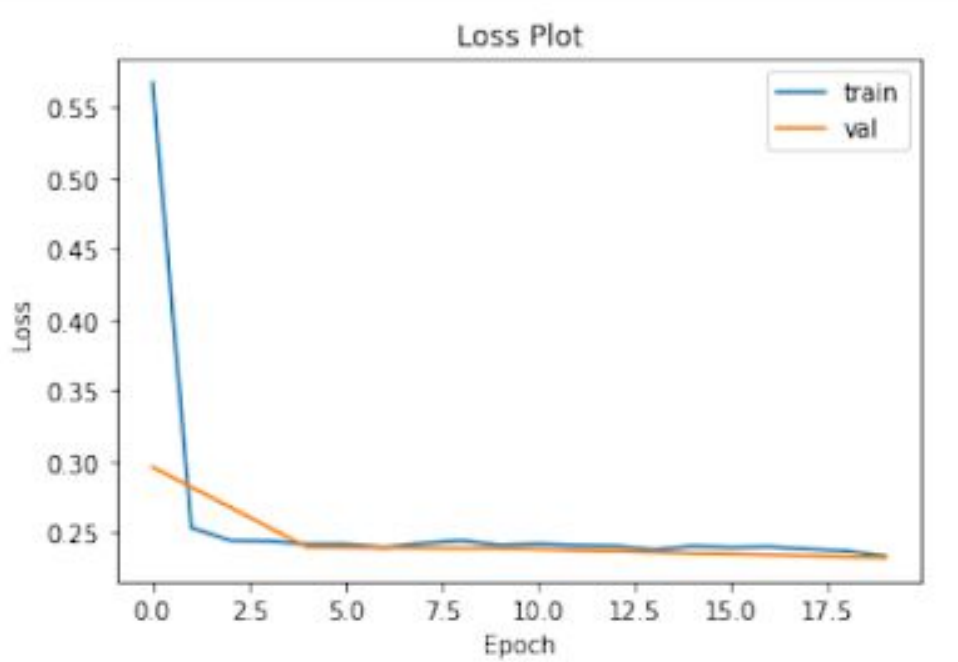


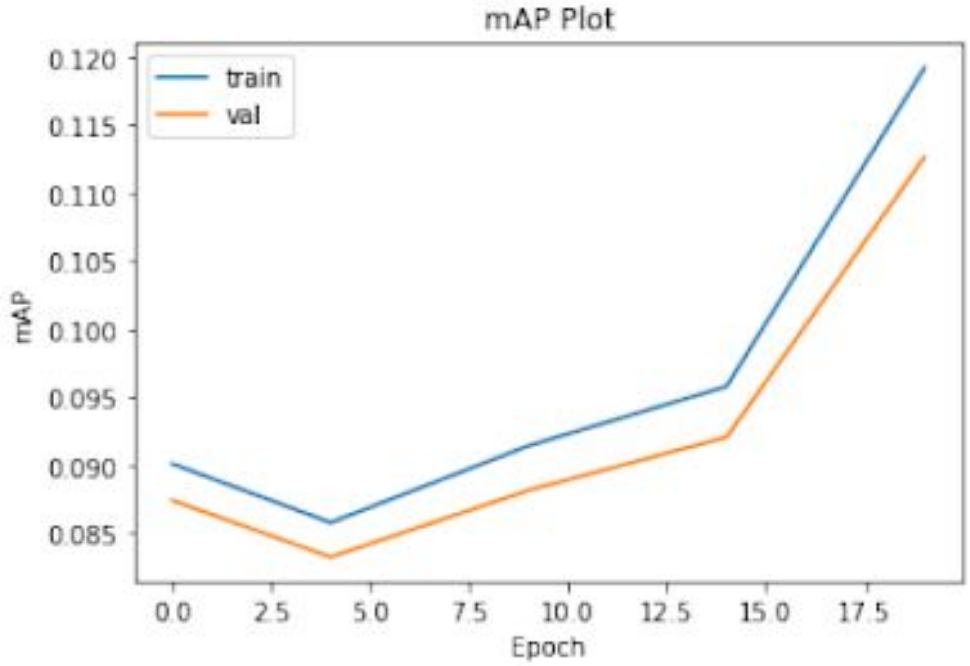


c. Provide analysis (at least 3 sentences):

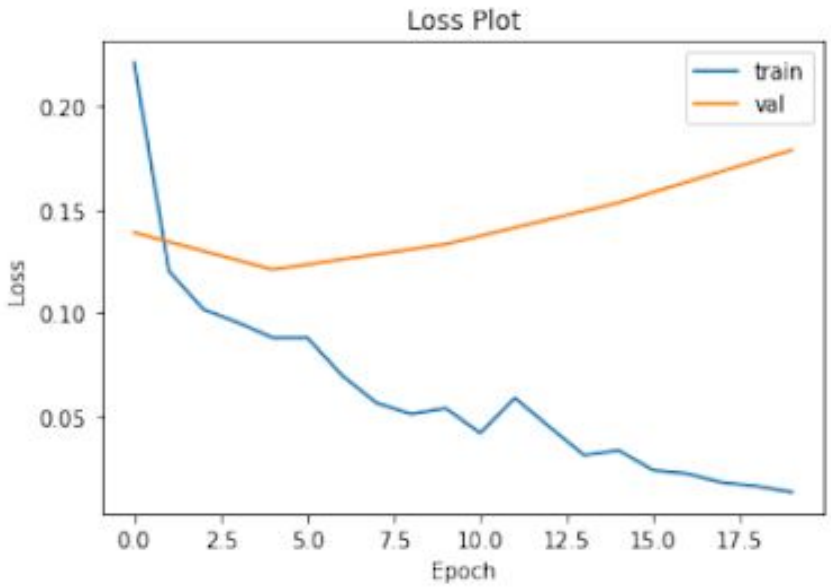
Looking at the training, it is certain that there is an increase of mAP if average loss decreases. It seems that the accuracy can go higher than 0.16 if there was more epoch to train the training data since mAP was kept increasing, not stable yet. Class person has an AP of 0.5366 in the test and it is the highest AP in the class, so this tells that simple classifier is good at detecting class person compared with other classes. Next high AP ones are class of aeroplane and car with APs of 0.3399 and 0.3238. These two classes are also good at detecting compared to other low APs classes such as bicycle or bottle, which have 0.0696 and 0.0792

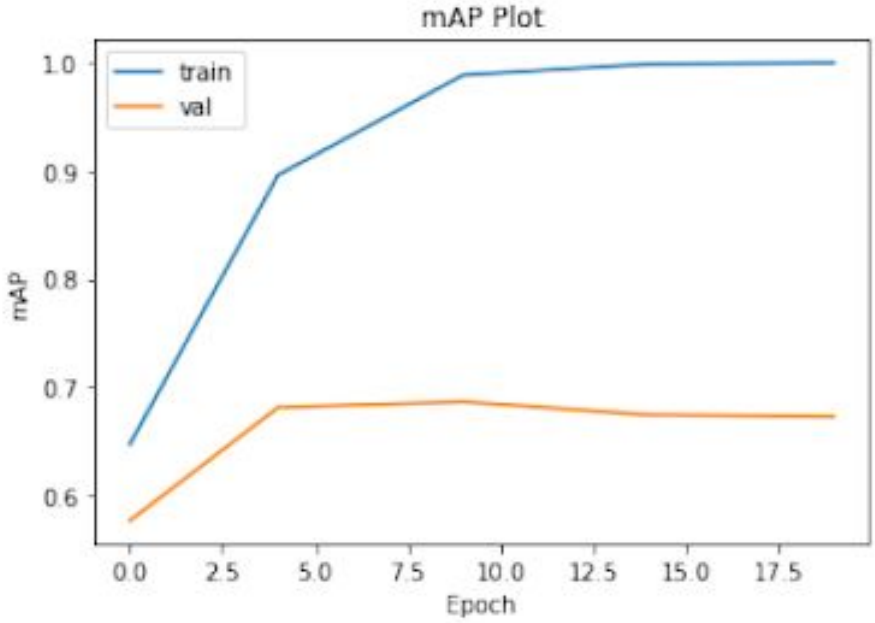
2. AlexNet from Scratch
    a. Report test mAP for alexnet: **0.1074**
    b. Visualize loss and mAP plots:





3. Pretrained AlexNet
    a. Report test mAP for pretrained alexnet: **0.6887**
    b. Visualize loss and mAP plots





c. Provide analysis on differences to training from scratch (at least 3 sentences):

Pretrained AlexNet definitely has the highest accuracy between three models. Graphs for pretrained AlexNet also show that the decrease of loss leads to the increase of mAP. Furthermore, it tells that mAP will look like a constant in the graph if the loss changes to the increase while it was decreasing. It seems there is an overfitting in the graph since train value reaches 1 while test value reaches approximately later in 0.6, which means that there is a big gap between train value and test value on mAP. Even though it has an overfit, it still produced

the highest mAP comparing with mAP and AlexNet from scratch, so it tells that it is the best pre-designed network between three of them.

Part-1B: Self designed network for multi-label classification

MP3_P1_Develop_Classifier in your assignment3_p1_starterkit should provide you with enough instruction to start with. You upload your output of your self-designed network to kaggle.

Did you upload final CSV file on Kaggle: **Yes**

1. My best mAP on Kaggle: 0.64050
2. Factors which helped improve my model
    a. Changed filters of conv2d to start from smaller ones. (Original was 64 to 16 | New is 16 to 64)
    b. Changed optimizer from SGD to Adam
    c. Changed to Alexnet
        i. First filter- in:3 out: 63 kernel:11 stride:4 pad:2
        ii. Second filter- in: 64 out: 192 kernel:5 pad:1
        iii. Third filter- in: 192 out: 384 kernel:3 pad:1
        iv. Fourth filter- in: 384 out: 256 kernel:3 pad:1
        v. Fifth filter- in: 256 out: 256 kernel: 3 pad:1
        vi. Max pool- kernel:3 stride:2
        vii. Average pool- output: 6x6
    d. Add dropout of p = 0.5 for fc3
    e. Add batchnorm for every convolutional layers
    f. Added more layers
    g. Data Augmentation by Random Rotation of 25 for train and random horizontal flip after center crop
    h. Vectorize tensor with -1, flatten
    i. Changed dropout p to 0.1 to drop less and applied drop for first and last linear

Table for final architecture (replace below with your best architecture design):

Layer No.	Layer Type	Kernel size (for conv layers)	Input | Output dimension	Input | Output Channels (for conv layers)
1	conv2d	11	227 | 56	3 | 64
2	BatchNorm2d	-	56 | 56	-
3	relu	-	56 | 56	-
4	maxpool2d	3	56 | 27	-

5	conv2d	5	27 | 27	64 | 128
6	BatchNorm2d	-	27 | 27	-
7	relu	-	27 | 27	-
8	maxpool2d	3	27 | 13	-
9	conv2d	3	13 | 13	128 | 256
10	BatchNorm2d	-	13 | 13	-
11	relu	-	13 | 13	-
12	conv2d	3	13 | 13	256 | 512
13	BatchNorm2d	-	13 | 13	-
14	relu	-	13 | 13	-
15	conv2d	3	13 | 13	512 | 1024
16	BatchNorm2d	-	13 | 13	-
17	relu	-	13 | 13	-
18	conv2d	3	13 | 13	1024 | 2048
19	BatchNorm2d	-	13 | 13	-
20	relu	-	13 | 13	-
21	conv2d	3	13 | 13	2048 | 512
22	BatchNorm2d	-	13 | 13	-
23	relu	-	13 | 13	-
24	conv2d	3	13 | 13	512 | 384
25	BatchNorm2d	-	13 | 13	-
26	relu	-	13 | 13	-
27	conv2d	3	13 | 13	384 | 256
28	BatchNorm2d	-	13 | 13	-
29	relu	-	13 | 13	-
30	conv2d	3	13 | 13	256 | 256
31	BatchNorm2d	-	13 | 13	-
32	relu	-	13 | 13	-
33	maxpool2d	3	13 | 6	-
34	avgpool2d	-	6 | 6	-
35	Linear1	-	9216 | 120	-
36	relu	-	120 | 120	-
37	Linear2	-	120 | 84	-
38	relu	-	84 | 84	-
39	Linear3	-	84 | 21	-

The initial network provided to you can be considered as the BaseNet. A very important part of deep learning is understanding the ablation studies of various networks. So we would like you to do a few experiments. Note, this **doesn't need to be very exhaustive** and can be in a cumulative manner in an order you might prefer. Fill in the following table :

Serial #	**Model architecture**	**Best mAP on test set**

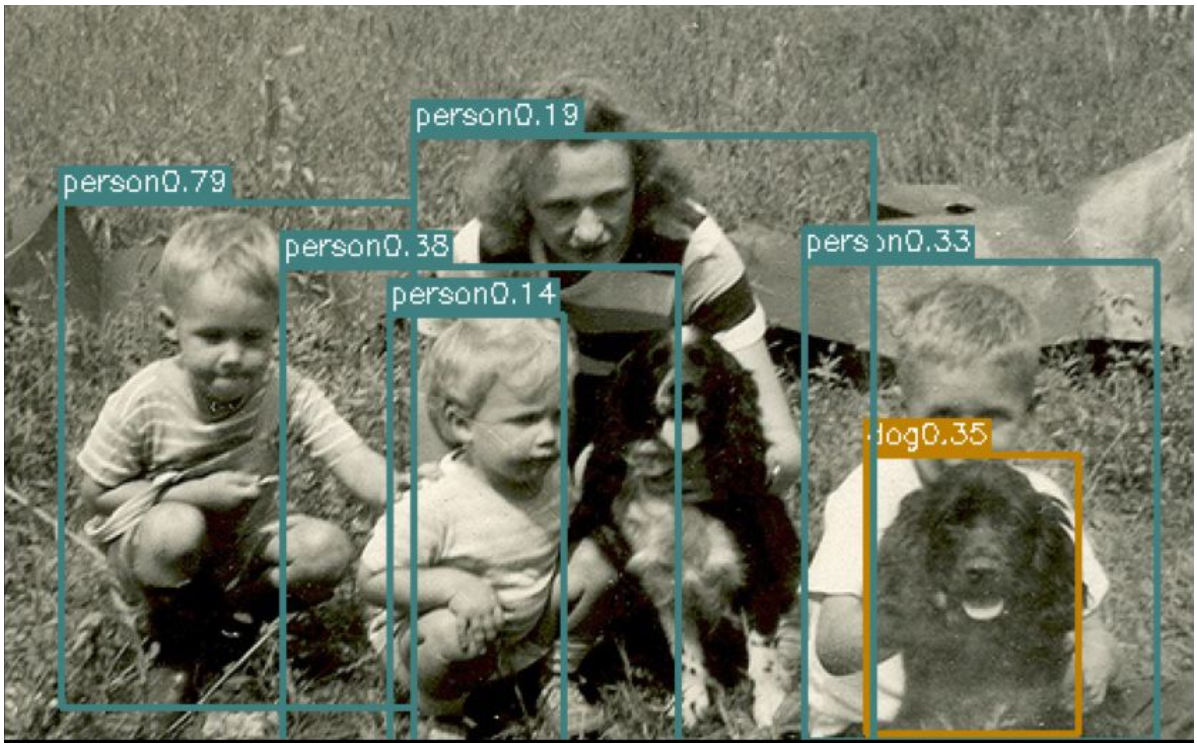
1	BaseNet	0.13827
2	BaseNet + a	0.17030 0.16329 0.16456
3	BaseNet + a + b	0.20579 0.22796 0.19523
4	BaseNet + a + b + c	0.2558
5	BaseNet + a + b + c + d	0.2650
6	BaseNet + a + b + c + d + e	0.3132 0.3344
7	BaseNet + a + b + c + d + e + f	0.3124
8	BaseNet + a + b + c + d + e + f + g	0.3329
9	BaseNet + a + b + c + d + e + f + g + h	0.3474 0.3452
10	BaseNet + a + b + c + d + e + f + g + h + i	0.3595

Make some analysis on why and how you think certain changes helped or didn't help:

Starting classifier was a modification of the Simple Classifier that was provided. The original one was starting from big to small. Since small to big layers are better, we modified the numbers for layers and it certainly affected the increase of mAP. Next approach was to change the optimizer to adam. This simple change also improved the increase of mAP. We have tried a few approaches such as increasing layers by decreasing start number of layer and increasing end number of layer; however, this did not work. We felt that the original Alex net will produce better than Simple Classifier, so we changed the layers with Alex net. This change led to improvement, and we added dropout. We thought that adding dropout to all linear would produce better, but it did not. We think it got lower since we were dropping too many elements since p was 0.5. So we only dropped the last linear, which improved the accuracy very little. With the help from piazza and lectures, we applied batch normalization for layers and this definitely improved the mAP. To make it better, we extended layers to output of 4096, but this was too much for gpu to handle. So we made it until 2048.This improved the mAP very little. After this, we implemented most of the improvements that are stated in the lecture such as data augmentation on train and flatten, which also helped. We were almost closed to the goal of mAP greater than 0.35, and we thought that droppin 0.5, which is half, would be too much, so we changed the number to 0.1 and applied to first and third linear, and this made to pass the goal of mAP to be greater than 0.35.

Part-2: Objection Detection by YOLO

1. My best mAP value on Kaggle : **0.47891**
2. Did you upload final CSV file on Kaggle: **Yes**
3. My final loss value : Last Updating best test loss happened when epoch = 40, with the loss of 2.57834
4. What did not work in my code(if anything):
   Even though passing the debug tool, I had a lot of problems to pass the part 2 yolo loss. First was the gradient problem, I had a lot of confusion how to make sure that gradient is used, and the way was to use Variable(). In the piazza, there were a lot of statements whether to use require_grads=True or detach(), but Variable() seems to solve them. To find this, I changed my code a lot, which made it hard to go back when I got lost. The other problem was that the data was not placed in cuda. Debug tool was fine with data having cpu; however, data needed to be inside of gpu for part 2, and modifying this part was tough since I first made all other of them to put in cuda. But finding best iou data only needed to be inside of gpu, not all other data. AFter fixing these, I was able to pass the mAP greater than 0.20 when epoch is at 10, which means that it works.
5. Sample Images from my detector from PASCAL VOC:



Sample Images from YOLO on YOLO to get **Extra Credit** for YOLO :