Name(s) : Sharvi Tomar , Samuel Folorunsho
Kaggle Team Name: Team SF

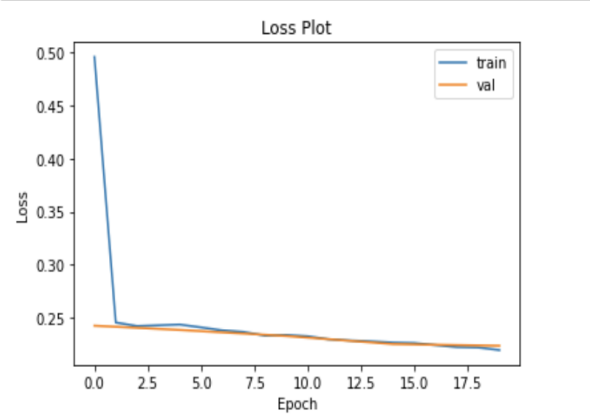
Part-1A: Pre-designed network for multi-label classification

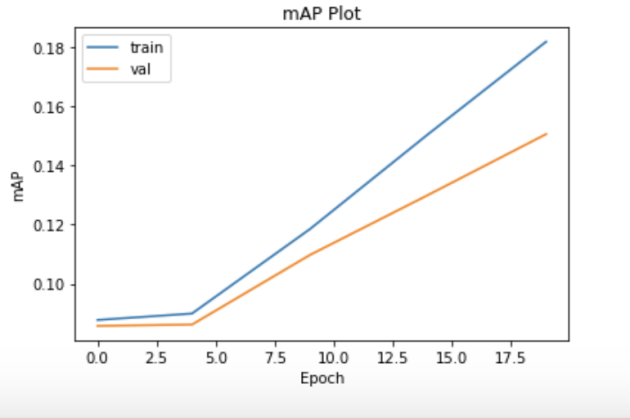
In this part, you will practice to train a neural network both by training from scratch or fine-tuning.

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: 0.1503
    b. Visualize loss and mAP plots:

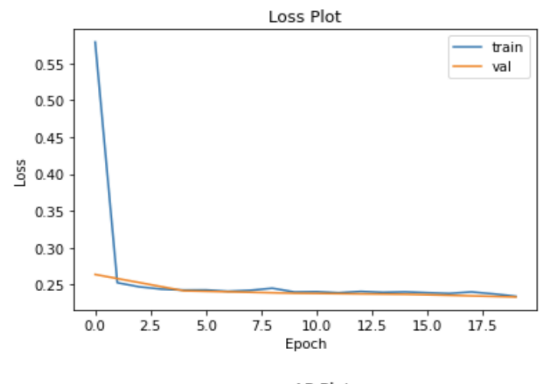


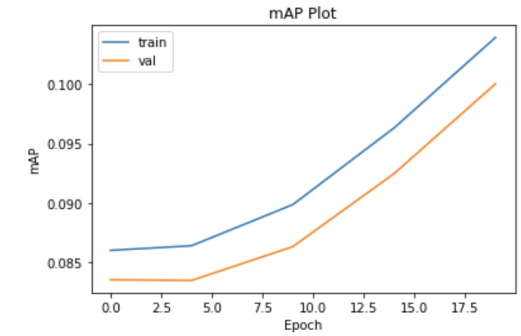


    c. Provide analysis of the plots (at least 3 sentences):

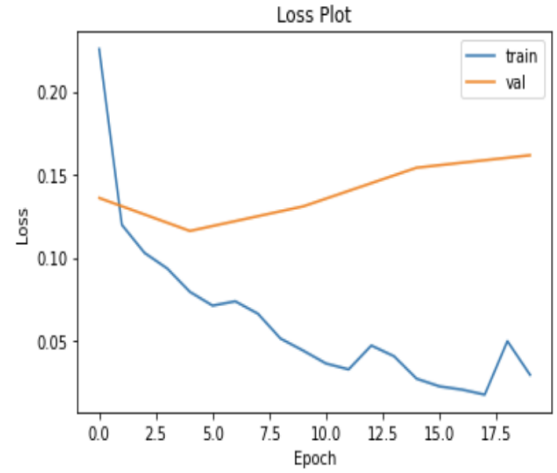
- The model starts with a very high training loss as it is being trained from scratch which initially drops drastically with epochs but then the decrease slows down.
- The best mAP value achieved with the given epochs on the training set is about 0.18.
- The mAP and loss value of training and validation data do not have a large gap indicating that the model has not overfit to the training data for those epochs.

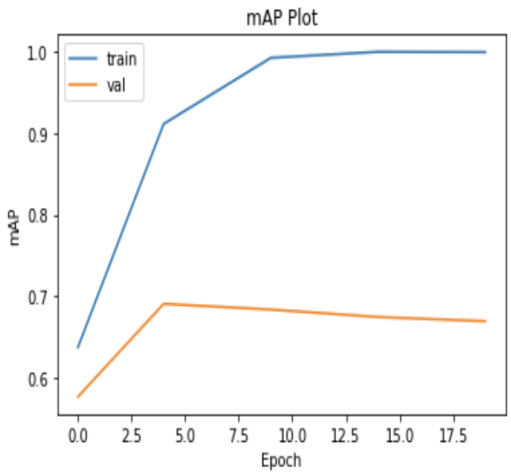
2. AlexNet from Scratch
    a. Report test mAP for alexnet: 0.0950
    b. Visualize loss and mAP plots:





3. Pretrained AlexNet
    a. Report test mAP for pretrained alexnet: 0.6850
    b. Visualize loss and mAP plots





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

- When training AlexNet from scratch, the training loss starts from a very high value of over 0.55 while with the pretrained network the loss starts from a significantly lower value of about 0.20.
- With the same number of epochs in training, the training loss with the pretrained network is around the range of 0.05 which is much lower than the training loss of the network from scratch which has loss in the range of 0.25.
- While there is a gradual increase in mAP value per epoch for the network trained from scratch, however for the pretrained network we see a dramatic increase in mAP with epoch initially however it then flattens out.
- For the pretrained network, there seems to be a large gap in training loss and validation loss (also, mAP value for training and validation set), suggesting a case of model overfitting to the training data. This is not the case with the network trained from scratch for the same number of epochs.

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: 0.62905
2. Factors which helped improve my model
    a. Included batch normalization after every convolutional layer
    b. Added more convolutional layers
    c. Added dropout after the fully connected layer
    d. Used Adam optimizer with learning rate=1e-4 in place of SGD optimizer with momentum(0.9) and lr=0.01

3. Table for final architecture:

Layer No.	Layer Type	Kernel size (for conv layers)	Input | Output dimension	Input | Output Channels (for conv layers)
1	conv2d	3	Stride = 1, padding =0	3 | 64
	BatchNorm2d	-	Num_features = 64	-
	relu	-	-	-
	maxpool2d	-	Kernel = 2	-
2	conv2d	3	Stride = 1, padding =0	64 | 64
	BatchNorm2d	-	Num_features = 64	-
	relu	-	-	-
3	conv2d	3	Stride = 1, padding =1	64 | 128
	BatchNorm2d	-	Num_features = 128	-
	relu	-	-	-
	maxpool2d	-	Kernel = 2	-
4	conv2d	3	Stride = 1, padding =0	128 | 128
	BatchNorm2d	-	128	-
	relu	-	-	-
5	conv2d	3	Stride = 1, padding =1	128 | 256
	BatchNorm2d	-	Num_features = 256	-
	relu	-	-	-
	maxpool2d	-	Kernel = 2	-
6	conv2d	3	Stride = 1, padding =0	256 | 256
	BatchNorm2d	-	Num_features = 256	-
	relu	-	-	-
7	conv2d	3	Stride = 1, padding =1	256 | 512
	BatchNorm2d	-	Num_features = 512	-
	relu	-	-	-
	maxpool2d	-	Kernel = 2	-

8	conv2d	3	Stride = 1, padding =1	512 | 1024
	BatchNorm2d	-	Num_features = 1024	-
	relu	-	-	-
	maxpool2d	-	Kernel = 2	-
	dropout	-	0.1	-
9	conv2d	3	Stride = 1, padding =1	1024 | 2048
	BatchNorm2d	-	Num_features =2048	-
	relu	-	-	-
10	conv2d	3	Stride = 1, padding =1	2048 | 2048
	BatchNorm2d	-	Num_features = 2048	-
	relu	-	-	-
	maxpool2d	-	Kernel = 2	-
	Linear	-	1179648 | 2048	-
	Relu	-	2048 | 2048	-
	Dropout	-	0.505	-
	Linear	-	2048 | 1024	-
	Relu	-	1024 | 1024	-
	Dropout	-	0.505	-
	Linear	-	1024 | 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 :

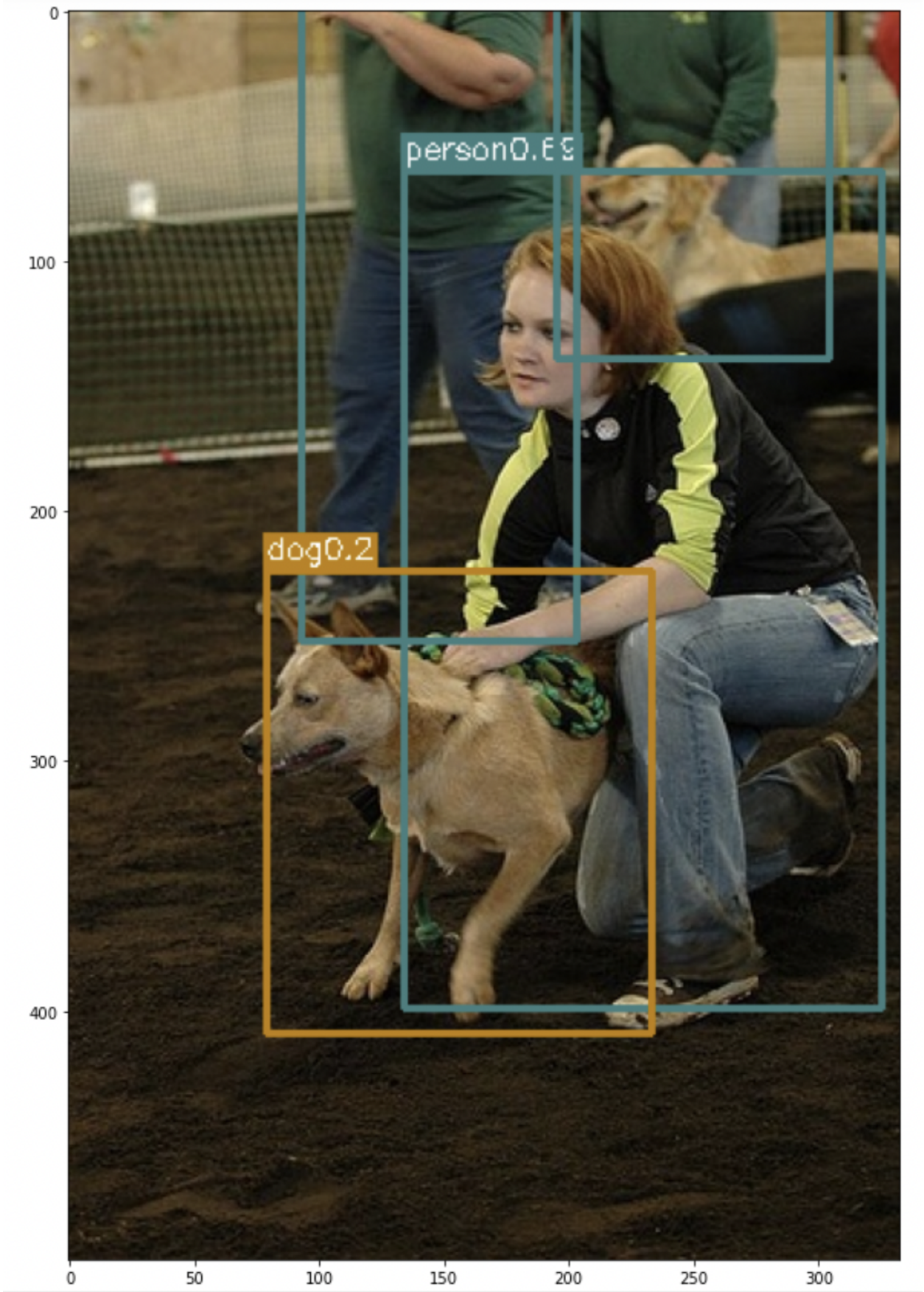
Serial #	**Model architecture**	**Best mAP on test set**
1	VGGNet inspired (dropout of 0.505 after FCs)	0.3011
2	Above + batchnorm	0.3241
3	Above + dropout after 8th conv layer of 0.1	0.3448

Part-2: Object Detection by YOLO

1. My best mAP value on Kaggle : 0.46840
2. Did you upload final CSV file on Kaggle: YES

3. My final loss value : 2.87515
4. What did not work in my code(if anything):
5. Sample Images from my detector from PASCAL VOC:


person0.54
0
50
100
150
200
250
300
350
0
100
200
300
400


person0.69
dog0.2
0
100
200
300
400
0
50
100
150
200
250
300


nonitor0.75
person0.72
person0.69
0
50
100
150
200
250
300
350
0
100
200
300
400

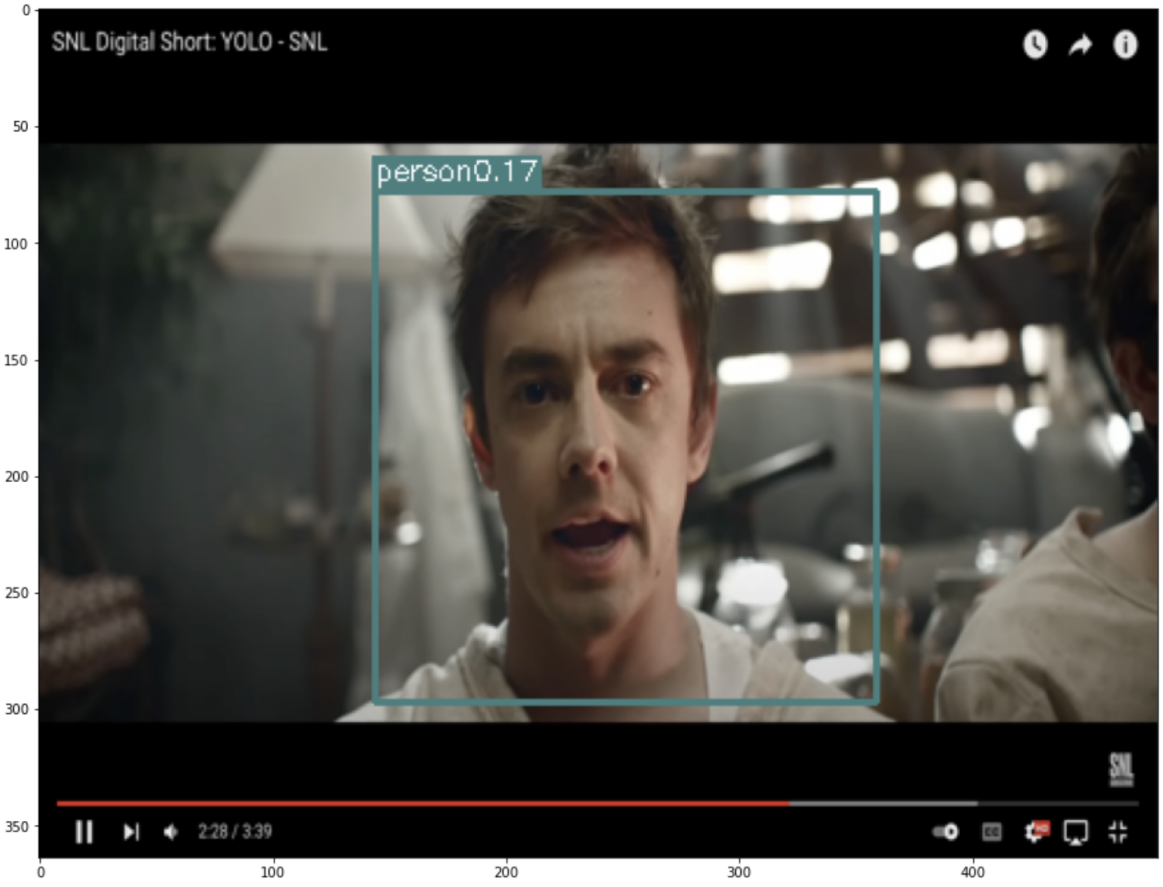

aeroplane0.37
0
50
100
150
200
250
300
0
100
200
300
400


dog0.31
0
50
100
150
200
250
300
0
100
200
300
400

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


person0.66
person0.42
person0.87
50
100
150
200
250
300
350
0
100
200
300
400


person0.53
pottedplant0.28
0
50
100
150
200
250
300
350
0
100
200
300
400


SNL Digital Short: YOLO - SNL
person0.17
2:28 / 3:39
0
50
100
150
200
250
300
350
0
100
200
300
400