**Name(s) :Wan Yoo**

**Kaggle Team Name:Wan Yoo**

**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
2. Report test mAP for simple classifier:
   * 0.1330
3. Visualize loss and mAP plots:

Chart, line chart

Description automatically generated

1. Provide analysis of the plots (at least 3 sentences):
   * Although the model barely overfits and the loss as well as the mAP increases over time, the overall performance is poor as the resulting loss is still very high and mAP lower than 0.15. Perhaps with increased epochs, we might be able to see the model converge with better mAP, but we can assume that the bound is not that high.
2. AlexNet from Scratch
3. Report test mAP for alexnet:
   * 0.2660
4. Visualize loss and mAP plots:

Chart, line chart

Description automatically generated

1. Pretrained AlexNet
2. Report test mAP for pretrained alexnet:
   * 0.6772
3. Visualize loss and mAP plots

Graphical user interface

Description automatically generated

1. Provide analysis on differences to training from scratch (at least 3 sentences):
   * To begin with, the resulting mAP for pretrained AlexNet was much, much higher than of the AlexNet from scratch. As visible from plot, the beginning mAP from epoch 0 has a huge difference. The AlexNet begins with mAP of 0.1, which is guessing by chance whereas the pretrained Alexnet begins with mAP of 0.6 but converges early. Also, the pretrained AlexNet grossly overfit as we can see the training mAP reaches 1.0.

**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: 0.49273
2. Factors which helped improve my model
   1. Batch normalization
   2. Change in model architecture
   3. Addition of layers
   4. Dropout
   5. Learning rate decay
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 | 8 | 227 | 224 | 3 | 32 |
| 2 | batchnorm2d | - | 224 | 224 | 32 | 32 |
| 3 | relu | - | 224 | 224 | - |
| 4 | maxpool2d | 2 | 224 | 112 | - |
| 5 | conv2d | 5 | 112 | 112 | 32 | 64 |
| 6 | batchnorm2d | - | 112 | 112 | 64 | 64 |
| 7 | relu | - | 112 | 112 | - |
| 8 | conv2d | 3 | 112 | 112 | 32 | 64 |
| 9 | batchnorm2d | - | 112 | 112 | 64 | 64 |
| 10 | relu | - | 112 | 112 | - |
| 11 | maxpool2d | 2 | 112 | 56 | - |
| 12 | conv2d | 3 | 56 | 56 | 64 | 128 |
| 13 | batchnorm2d | - | 56 | 56 | 128 | 128 |
| 14 | relu | - | 56 | 56 | - |
| 15 | conv2d | 3 | 56 | 56 | 128 | 128 |
| 16 | batchnorm2d | - | 56 | 56 | 128 | 128 |
| 17 | relu | - | 56 | 56 | - |
| 18 | maxpool2d | 2 | 56 | 27 | - |
| 19 | conv2d | 5 | 27 | 27 | 128 | 256 |
| 20 | batchnorm2d | - | 27 | 27 | 256 | 256 |
| 21 | relu | - | 27 | 27 | - |
| 22 | conv2d | 3 | 27 | 27 | 256 | 256 |
| 23 | batchnorm2d | - | 27 | 27 | 256 | 256 |
| 24 | relu | - | 27 | 27 | - |
| 25 | conv2d | 3 | 27 | 27 | 256 | 256 |
| 26 | batchnorm2d | - | 27 | 27 | 256 | 256 |
| 27 | relu | - | 27 | 27 | - |
| 28 | conv2d | 3 | 27 | 27 | 256 | 256 |
| 29 | batchnorm2d | - | 27 | 27 | 256 | 256 |
| 30 | relu | - | 27 | 27 | - |
| 31 | conv2d | 3 | 27 | 27 | 256 | 256 |
| 32 | batchnorm2d | - | 27 | 27 | 256 | 256 |
| 33 | relu | - | 27 | 27 | - |
| 34 | maxpool2d | 3 | 27 | 14 | - |
| 35 | conv2d | 5 | 14 | 14 | 256 | 512 |
| 36 | batchnorm2d | - | 14 | 14 | 512 | 512 |
| 37 | relu | - | 14 | 14 | - |
| 38 | conv2d | 3 | 14 | 14 | 512 | 512 |
| 39 | batchnorm2d | - | 14 | 14 | 512 | 512 |
| 40 | relu | - | 14 | 14 | - |
| 41 | conv2d | 3 | 14 | 14 | 512 | 512 |
| 42 | batchnorm2d | - | 14 | 14 | 512 | 512 |
| 43 | relu | - | 14 | 14 | - |
| 44 | maxpool2d | 2 | 14 | 7 | - |
| 45 | linear | - | 25088 | 4096 | - |
| 46 | relu | - | 25088 | 4096 | - |
| 47 | dropout | - | - | - |
| 48 | linear | - | 4096 | 1000 | - |
| 49 | relu | - | 4096 | 1000 | - |
| 50 | dropout | - | 1000 | 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 | 0.17 |
| 2 | BaseNet + a | 0.23 |
| 3 | BaseNet + a + b + c + d | 0.38 |
| 4 | BaseNet + a + b + c + d + e | 0.49 |

**Part-2: Object Detection by YOLO**

1. My best mAP value on Kaggle :
2. Did you upload final CSV file on Kaggle:
3. My final loss value :
4. What did not work in my code(if anything):
5. Sample Images from my detector from PASCAL VOC:

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