Theory and Practice of Deep Learning – Mini Project

Caleb Lee Liang Heng 1002673

Chan Chen Chang Joseph 1002948

Files:

parser.py – Configurations and hyperparameters

data.py – Dataset class for PASCAL VOC dataset, to pre-process data from data directory and return image with a binary encoding of target classes.

utils.py – Helper functions to get class-wise and mean average precision score, loss, tail accuracy and top 50 images

main.py – Main method, to run training or to reproduce results.

To train, run:

python main.py --run train --data_dir <path to PASCAL VOC data directory>

To reproduce results, run:

python main.py --run results --data_dir <path to PASCAL VOC data directory>

Results: Class-wise average precision, mean average precision, tail accuracy graph, top 5 and bottom 5 images for 5 random classes.

Implementation

Model: Transfer learning using resnet-18 with pre-trained weights, override last FC layer with nn.Linear(512, 20) for predictions of the 20 classes respectively, and train all layers.

Loss: nn.BCEWithLogitsLoss() – The target is a binary list of length 20, where each element could be "1" representing that the image lies in that respective class, or "0" otherwise. The output of the FC layer of the network (without any non-linearity) is the prediction scores of the 20 classes respectively. The loss function first applies a sigmoid function to "compress" these scores (raw logits) into a value from 0 to 1, so each element can be interpreted as a "probability" of the image falling under that class. Since each entry in the target and output from sigmoid function represents a class, it can be seen as 20 different binary classifiers for the 20 classes respectively, and the mean of each binary cross entropy loss of these 20 different binary classifiers is taken by default to give the classification loss.

Performance measure: sklearn.metrics.average_precision_score – class-wise average precision. The mean of the class-wise average precision scores is used as the performance measure.

Optimiser: torch.optim.SGD – Stochastic Gradient Descent with different learning rates

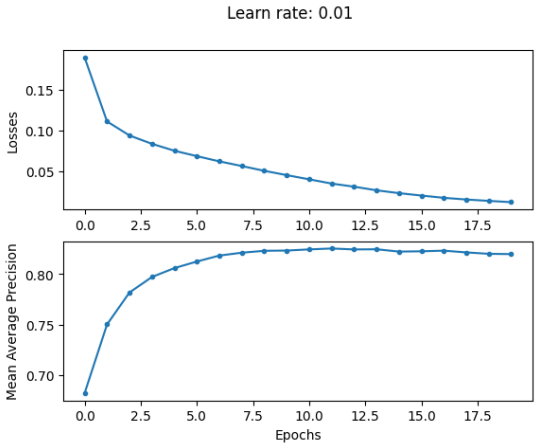
Hyperparameter: Learning rate – trained 3 different models with using learning rate = 0.01, 0.005, 0.001

Training Procedure:

Train 3 different models with using learning rate = 0.01, 0.005, 0.001 respectively, and then picking the best epoch based on the mean average precision score over the 20 classes. Comparing the best epochs for each of the learning rate used, the one with the best mean average precision score will be selected as our model to produce the results shown later.

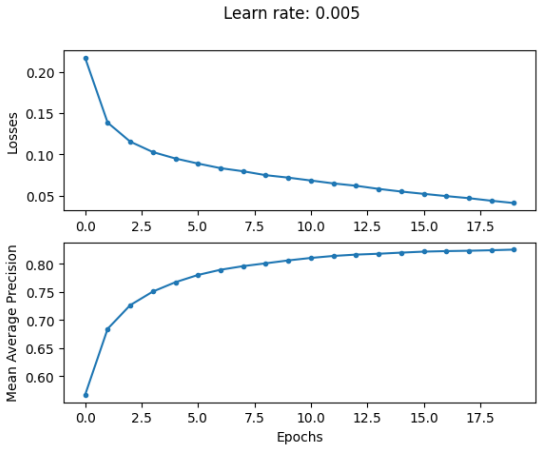
Model Selection:

Learning Rate = 0.01



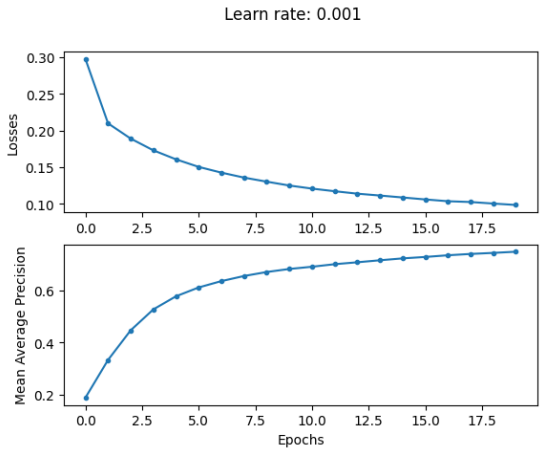
	Train Loss	Average Precision Measure
0	0.18916303035933213	0.6818427166986142
1	0.11070068875671099	0.7501655965805862
2	0.09353843774625709	0.7818541314305623
3	0.08333724435398032	0.7972302027398264
4	0.07486333121550816	0.8060646009482125
5	0.06826283256185121	0.8125859269625213
6	0.06184980083860499	0.8184935339649764
7	0.05615101494519404	0.8213274389914069
8	0.050293455451893405	0.8232207639774911
9	0.04503621337883299	0.8234787014095322
10	0.039873964907653506	0.8246267299813178
11	0.03467881047900496	0.8255220010438566
12	0.03086388163143696	0.8245887760346597
13	0.026456782807642854	0.8247589087242396
14	0.02296007732605801	0.8224383937437236
15	0.020015178055778227	0.8227395208116188
16	0.017212666586749047	0.8233646736310984
17	0.015154195600839634	0.8216159701587122
18	0.013488358791979998	0.8202339498937073
19	0.011923836029067053	0.8199348650941458

Learning Rate = 0.005



	Train Loss	Average Precision Measure
0	0.21713134030390052	0.5664738834102412
1	0.13834170447714503	0.6842517250678357
2	0.1152982619804377	0.7263723188219144
3	0.10266299312674133	0.7504928057809909
4	0.09486741426936741	0.7671800303469759
5	0.08873061789980148	0.7800826917902443
6	0.08311502408631687	0.7893354222485933
7	0.07934973628244586	0.7957760286413785
8	0.07463602320025753	0.8008089906072584
9	0.07165771876799994	0.8059801449225452
10	0.06817345004840936	0.8103898437288928
11	0.06462708714954014	0.8139322484536182
12	0.0617907566314969	0.8163257827960351
13	0.05797244298308255	0.8178190897707704
14	0.0547716049217312	0.8197984632308911
15	0.05197148010467684	0.8215891170076841
16	0.04924274803082037	0.8225340715730312
17	0.04673009904819494	0.8231531010051942
18	0.04362473012515287	0.8240525944080928
19	0.04080127644763645	0.8252145351577591

Learning Rate = 0.001



	Train Loss	Average Precision Measure
0	0.29785719788607273	0.18767859674425172
1	0.20980243875993695	0.33270355913631033
2	0.18895161135236643	0.4463561091089557
3	0.1729186985579283	0.5264837710988333
4	0.16061621005308693	0.5767363321795747
5	0.15031075473437763	0.6105602260902813
6	0.14248635390950315	0.6348430436582231
7	0.13567375912013666	0.654457443964781
8	0.13014473920094902	0.669721894960292
9	0.12485461111841255	0.6814227994637028
10	0.1206456732899783	0.689955528738856
11	0.11707368422320436	0.6997042392038095
12	0.11372783068338586	0.7070557828460982
13	0.11116654113684286	0.7147748355261958
14	0.10851773844417913	0.7219872317054208
15	0.105834020665904	0.7275061776953377
16	0.10339778291280043	0.7339572279240495
17	0.1023212130592522	0.7390453104422241
18	0.10021084005939228	0.743056243208433
19	0.09828076303504699	0.7475866650418049

Learning Rates: 0.01, 0.005, 0.001

Best Average Precision Measures: 0.82552 (epoch 11) , 0.82521 (epoch 19) , 0.74758 (epoch 19)

Best Learning Rate: **0.01**

Results:

Mean Average Precision: 0.8255220010438566

Class-wise average precision

aeroplane	0.9701190878422842
bicycle	0.8417051108449738
bird	0.949234395840914
boat	0.8567520622415427
bottle	0.6268438039932406
bus	0.9238711875777725
car	0.7812191766769104
cat	0.9589509311347214
chair	0.706506996607985
cow	0.7663862104184249
diningtable	0.6462856800192354
dog	0.9333103753387456
horse	0.888297027890729
motorbike	0.88170689539861
person	0.9549598087814033
pottedplant	0.5393086017277482
sheep	0.8922349923403414
sofa	0.6264113785886816
train	0.9524905315611604
tvmonitor	0.8138457660517104

train: top and bottom 5

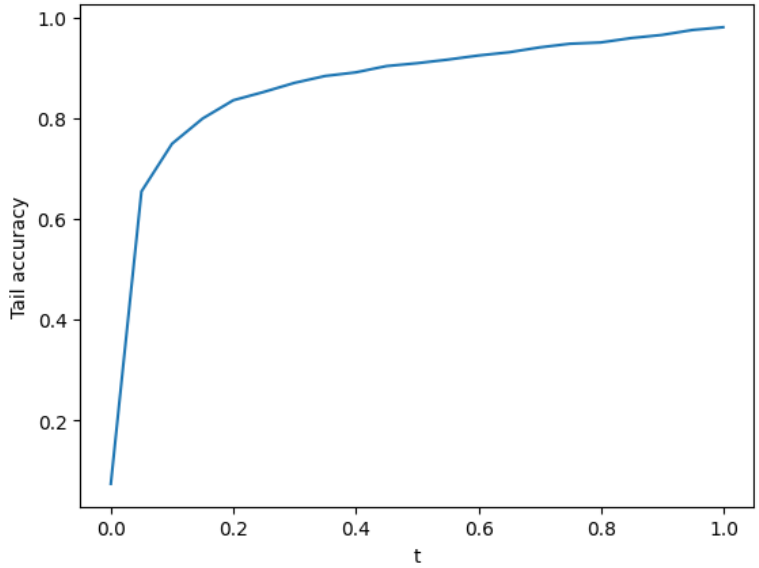
bird: top and bottom 5

pottedplant: top and bottom 5

sheep: top and bottom 5

sofa: top and bottom 5

Tail Accuracies vs threshold (from 0 to max(f(x))):



We first feed the output of the FC layer of the network (without any non-linearity) through a sigmoid function to give f(x), which is a value between 0 and 1.

Tail accuracy works by first classifying this value to 1 if it is above the threshold and 0 if the value is below. The precision is then measured, which is given by the ratio of the true positive vs (true positive + false positive). Thus, as t starts out from 0, every output from the sigmoid function will be above that threshold, resulting in mostly false positives and hence a low tail accuracy. As t reaches 1, every output from the sigmoid function will be below that threshold, and there will thus be no false positives, although there will be mostly false negatives which are not accounted by how precision is measured.

With this, the graph will be strictly increasing as t increases, and choosing to vary the threshold from 0 to max(f(x)) which is upper-bounded by 1, we can have an understanding of the distribution of the outputs from out network based on the slope.

Given a steep increase at the start before t reaches 0.5, we can infer that most of the output from the network has a negative value which results in a value <0.5 after passing through the sigmoid function.