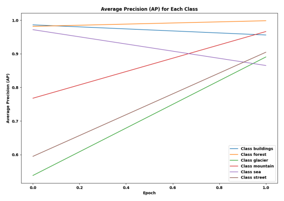
Report Mandetory assignment 1 for IN4310

Task1 a-e

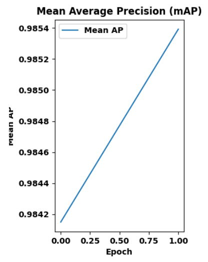
Task 1.a, In this task I use *os.getwd* which ensures that as long as the data-folder *mandatory_data.zip* is in the mandatory1 folder the code is able to run.I use the *train_test_split* from skitlearn with $seed = 42$ so that the word is reproducible.This ensures that the same test,train and validation sets get loaded every time the $Data_utils$ form *dataloader.py* is called.

In this assignment I choose to train a Googlenet model from Pytorch torchvisions models.The metrics of used to evaluate the model is AP,mAP and $Accuracy$. For AP I use skitlearns *average_precision_score*.For each class that class is set to the true label.Firs for each class we check how many of our predictions matches.This gives us a binary vector.Together with the binary vector we also store what the probability of confidence the model has in its prediction of the positve class.Finally we pass bought vectors into the *average_precision_score* true label. mAP get calculated by averaging over class specific $AP's$ and the accuracy is calculated manually.

Our simulations are done using batch-size of 10.With our own GPU card NVIDIA GeForce RTX 3050 this simulation runs smoothly.The code was not tested on ml9 because of login issues but the observation was that even exceeding 666 mb lead to a memory error. We tested training the model with the following learning rate hyper-parameters:$0.001, 0.01, 0.05$.The best results where attained with the learning-rate set to 0.001.The results from the figures below are attained from our best performing hyper-parameter.

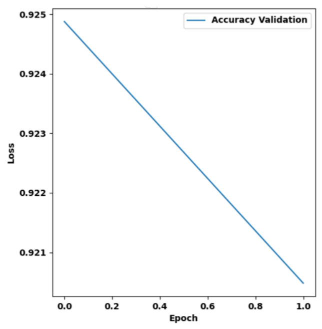


(a) Plot showing AP for each class of the best model on the validation set

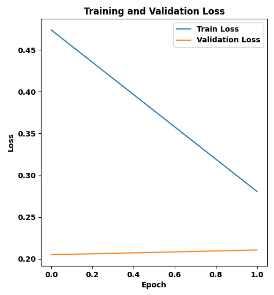


(b) Plot showing mAP of the best model on the validation set

Figure 1: These plots are generated with 2 epochs



(a) Plot shows Accuracy each class of the best model on the validation set



(b) Plot shows loss of training and validation loss for the best model

Figure 2: These plots are generated with 2 epochs

We choose to run the simulation on 2 epochs since we observed that already after the fire epoch the loss stopped changing when when looking at improvement between the consecutive losses for each batch within each epoch.The figure are generated by collecting the loss for each epoch and the does not illustrate this observation, but we saved the output of the loss progression in the $Task1\_1\_a\_e.out$ where this observation could be observed.

The best model ended up having the following performance:

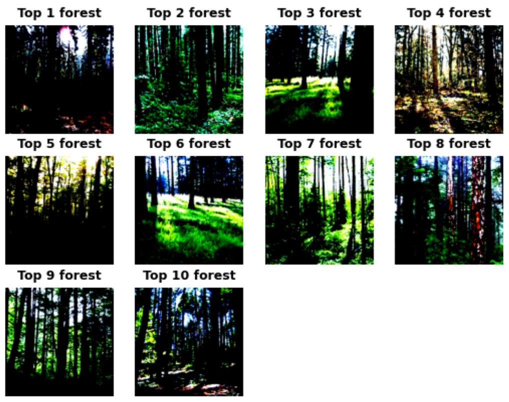
- mAP for test set: 0.9879048488309602
- accuracy for test set: 0.93249184
- Ap pr class for test set:
  - buildings: 0.9956474224195278
  - forest: 0.9996732004716232
  - glacier: 0.9814687679915896
  - mountain: 0.9637840778433583
  - sea: 0.9976937616296498
  - street: 0.989161862630013

The result is suspiciously good on the test set,but we did observe that in 1.f the lowest confidence probability softmax score was a little over 0.3 and the corresponding predicted values where correct.This could indicate that even when the model has low confidence the correct class still gets higher score then each of the others.

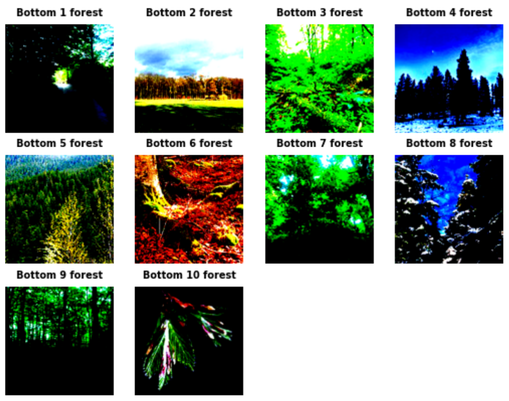
The plot for the validation stay constant for epoch 0 and 1,and the rest of the plot indicate that the model are doing good predictions just after 1 epoch,which could be expected since we are using a pre-trained google-net model.

Task1 f

For this section we use our stored model to evaluate predict on the test images before sorting the predictions in descending order relative to the confidence the model have about its prediction so we take the softmax of log probability predictions to attain the softmax scores.We save these scores before loading them back,before choosing three classes(forest,mountain and street) to show the top 10 image images s rated according to how sure the model is about its prediction.We also find the bottom 10 images for each of the three classes.

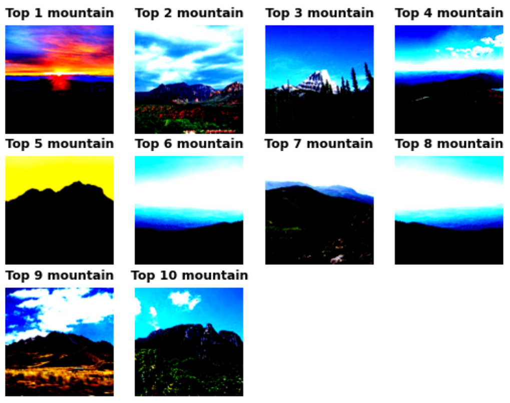


(a) Figure shows the top 10 images from the forest class

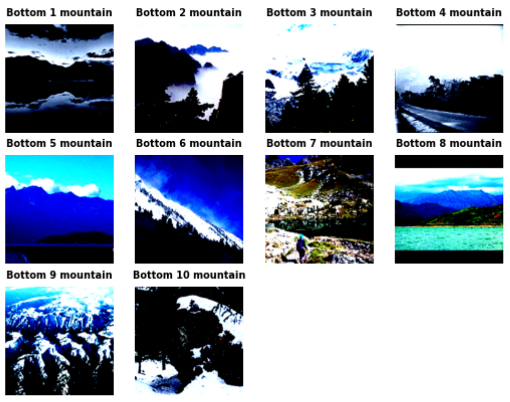


(b) Figure shows the bottom 10 images from the forest class

Figure 3

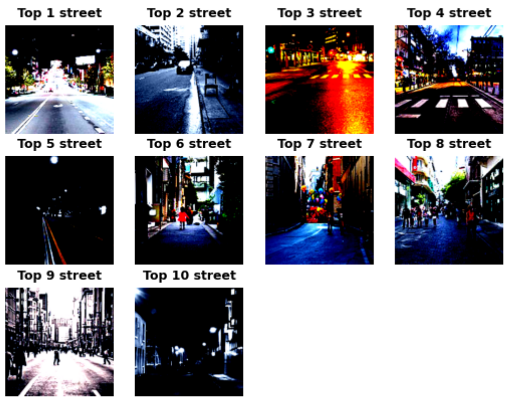


(a) Figure shows the top 10 images from the mountain class

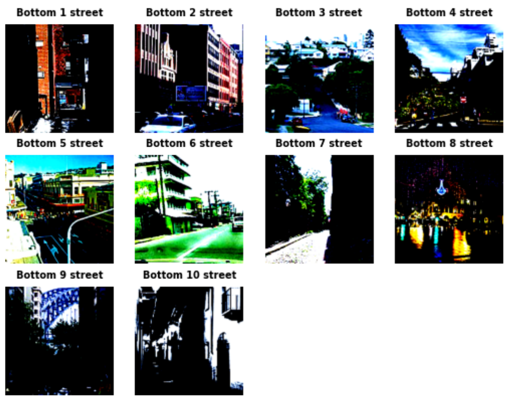


(b) Figure shows the bottom 10 images from the mountain class

Figure 4



(a) Figure shows the top 10 images from the street class



(b) Figure shows the bottom 10 images from the street class

Figure 5

2

We choose to track the map features:$maxpool1$,$conv3.conv$,$inception3a.branch1$, $inception4c.branch4.1.bn$ and $inception5b.branch4.1.conv$ to track what happens to there weights when we do a forward pass for 200 images and record the average percentage of none positive weights for each of the layers.We do this by using hook functionality from Pytorch: `https://pytorch.org/docs/stable/generated/torch.nn.modules.module.register_module_forward_hook.html`.

We get the following result:

Average percentage of non-positive values:

- maxpool1: 20.25075982541455
- conv3.conv: 61.326126202434104
- inception3a.branch1: 35.43831712372448
- inception4c.branch4.1.bn: 58.29340720663266
- inception5b.branch4.1.conv: 50.55070153061223

These results can also be found in Task2.out.