IN5400 Mandatory 1

Espen Lønes (espenlon)

March 16, 2022

The source code is located in the 4 python files,
('train_pytorch_in5400_studentversion.py', 'YourNetwork.py', 'RainforestDataset.py', 'read_trained.py').
The train_pytorch... file contains the code for running the training of the networks for the three tasks. The read_trained file contains the code for loading the pre trained models, comparing results, calculating tail accuracy for task 2 and making plots.
The code assumes it is located at an uio ml node. This can be done with the flowing command:
$ scp path/to/file/on/your/computer your_username@ml6.hpc.uio.no:.
The programs are then run using:
$ CUDA_VISIBLE_DEVICES=X python yourscript.py
where X is changed to a given gpu number.

The trained models are located in the Task1.pt, Task3.pt, Task4.pt files.
The data/results from the best epoch as well as losses and test AP scores from all epochs are in Task1.npz, Task3.npz, Task4.npz files.
The root for the dataset is set to be '/itf-fi-ml/shared/IN5400/2022_mandatory1/'. With 'train_v2.csv' as addition for the label/image-name file and 'train-tif-v2/' for the image directory.
All these files are expected to bee in the same directory as the .py files.
All netts where trained for 12 epochs, with a batch size of 16 (test batch size of 64), original learning rate of 0.005.
For lr scheduling i used the torch.optim.lr_scheduler.StepLR() function with step size of 5 and gamma of 0.3.
For random seed values 'torch.manual_seed(0)' and seed=0 in the splitting of test and train data was used.
As optimizer the torch.optim.SGD() Stochastic gradient decent was used. (with momentum=0.9)
For loss and activation function i used Binary-cross-entropy loss and sigmoid activation at the last layer.
(Both from torch.nn, nn.BCELoss() and nn.Sigmoid())

$$L_{BCE} = \frac{1}{n}\sum_{i=1}^{n} y_n * log(x_n) + (1 - y_n) * log(1 - x_n)$$

This works for multi label classification by taking the Binary cross-entropy loss for each label/class and at the end summing up all the individual losses. Thereby making it able to train on all 17 labels/classes separately.

Could also have used Multi label soft margin loss.

For Task 1 the best epoch was 9 with a mAP of 0.47 and class/label wise AP of:

```
[0.94286683 0.59885883 0.53989198 0.73346883 0.82825735 0.58773021
 0.17440899 0.03481306 0.0024718  0.02080455 0.39071507 0.52500477
 0.99600525 0.76342075 0.05781284 0.0420444  0.74655047]
```

For Task 3 the best epoch was 11 with a mAP of 0.52 and class/label wise AP of:

```
[0.94844684 0.76703967 0.56916545 0.75781304 0.8497898  0.71609065
 0.20580291 0.04444695 0.00302273 0.26046536 0.40594809 0.55637063
 0.99690697 0.77866851 0.04756682 0.07555648 0.79608889]
```

For Task 4 the best epoch was 8 with a mAP of 0.46 and class/label wise AP of:

```
[0.94452742 0.69378743 0.53963354 0.7066414  0.81554246 0.52666297
 0.18466993 0.03461248 0.00292161 0.00815927 0.37435567 0.5006309
 0.99619225 0.74806291 0.03177645 0.02725292 0.75401366]
```

Comparing these results we see that Task 1 and Task 4 provided quite similar results leading us to a theory of adding the near-ir channel gives little effect for the model.
While The 'TwoNetworks' model in Task 3 gave a substantially better mAP score.

Now comparing the test AP scores and test prediction scores from the best epoch with those from the loaded model we get:

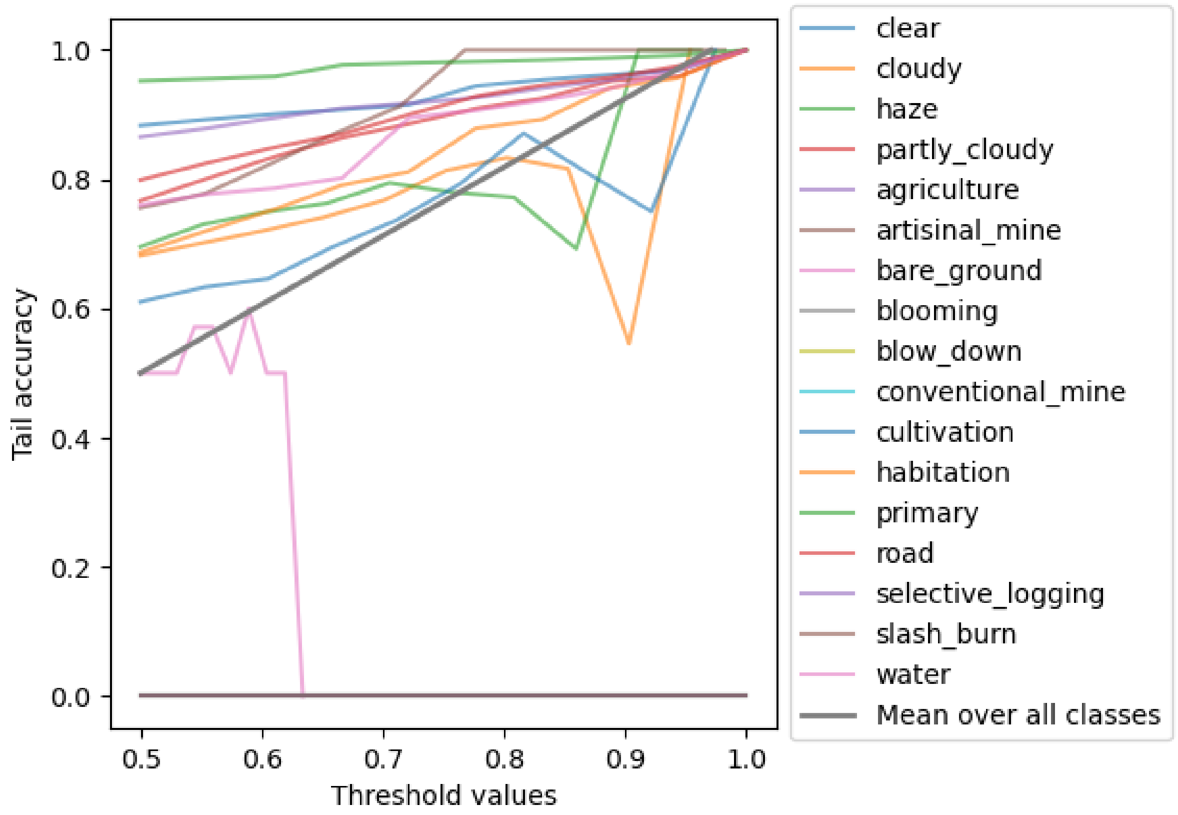
```
Task1  - Differace in AP:
[0.00028086 0.01323783 0.0257037  0.0025158  0.00399913 0.04876219
 0.00369989 0.00434034 0.00034137 0.00585285 0.00174137 0.00024558
 0.00018056 0.00065025 0.01663423 0.00117686 0.00133497]
Task1  - Differace in mAP:  0.005350826969142131
Task1  - Mean differace in predictions:  0.013781451226606399
Task1  - Max differace in predictions:  0.7657131180167198

Task3  - Differace in AP:
[0.0021921  0.00335372 0.00415189 0.00031694 0.00071281 0.02275714
 0.00214009 0.00257057 0.00032272 0.02152926 0.00277341 0.00207177
 0.00018134 0.00419892 0.00442369 0.00093866 0.000563  ]
Task3  - Differace in mAP:  0.002371634248843124
Task3  - Mean differace in predictions:  0.013076516196150242
Task3  - Max differace in predictions:  0.6927015781402588

Task4  - Differace in AP:
[0.00184556 0.00475762 0.00520604 0.00299627 0.00197636 0.00940589
 0.00817361 0.01038314 0.00057623 0.00109958 0.00702966 0.00550988
 0.00018982 0.00699453 0.01498971 0.00378108 0.00018992]
Task4  - Differace in mAP:  0.0002896788788057658
Task4  - Mean differace in predictions:  0.014903311702692512
Task4  - Max differace in predictions:  0.765098012983799
```

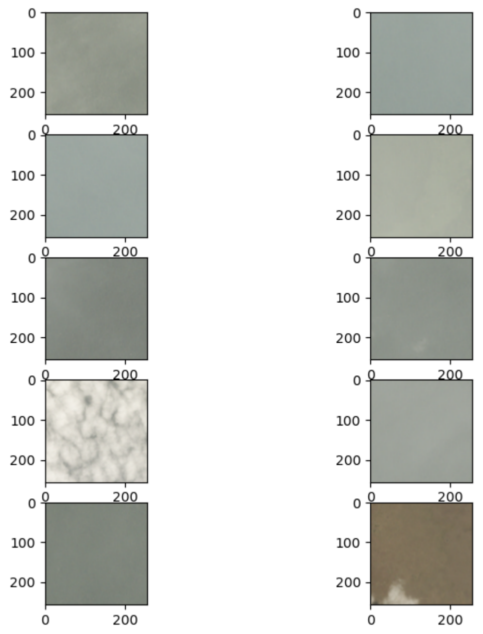
We see here that for the classwise AP and mAP scores the difference is quite small.
The mean difference for prediction scores is also low at 0.013 - 0.015.
But the maximum difference ranges from 0.69 - 0.76, this is allot when the prediction scores only range from 0 to 1.
This shows that most predictions are quite similar or the same but at least one (probably more than one) is very different. It is expected that there is some difference in the reproduction but not as much as the maximum. It is likely because i forgot to remove the random crop and random horizontal flip transforms for the test datasets.

Task 2:
The figure under shows the tail accuracies for 10 values of t from $t = 0.5$ to $t = max_x f(x)$
With tail accuracy on the y-axis. And threshold on the x-axis.



The classes with few representations in the dataset had very bad prediction scores on their images. This resulted in some classes having no predictions over 0.5. For these i just set their accuracy to 0 for all t.
But class many classes had normal behaviour. Namely as t becomes larger we look at images the model is more and more confident about. If the model is confident we would expect high accuracy, which is exactly what we see. Especial for the line depicting the mean tail-acc over all classes.

As for the top 10 and bottom 10 images for the class with highest AP (Seen in Figure 1 on the next page). The worst images were well bad images for the given class. All of them had low contrast and almost completely gray, very different to the 'best' images. One of the 'bad' images is also probably miss labelled as it looks like clouds. The 'best' images on the other hand where much better, having clearly defined features making them much easier for the model to work with.


(a) Worst images

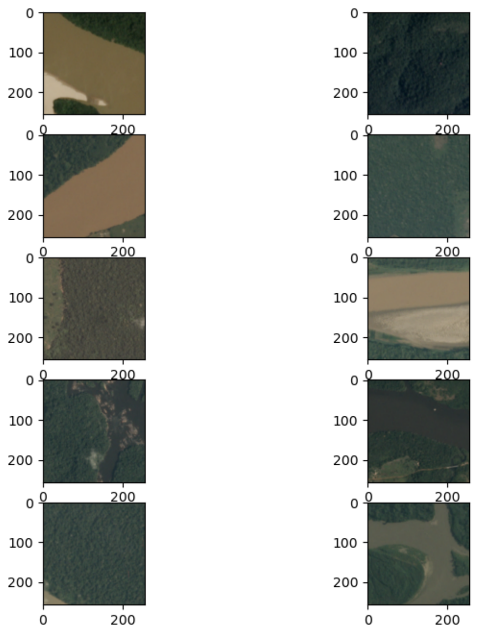

(b) Best images

Figure 1: Best and Worst images

Furthermore i show plots of the classwise AP score per epoch, mAP per epoch and test/train losses per epoch for all the tasks. These all look as one would expect.

Task 1:

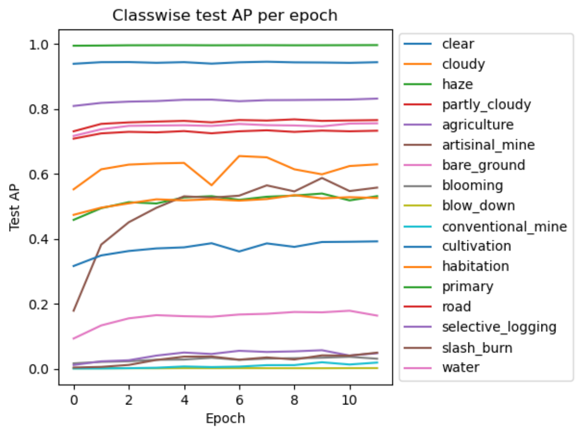


Figure 2: Classwise AP

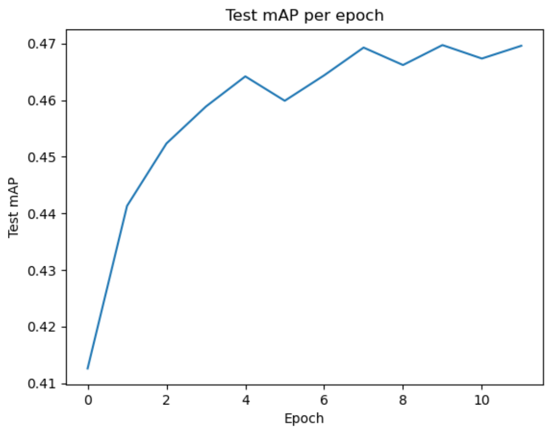


Figure 3: mAP

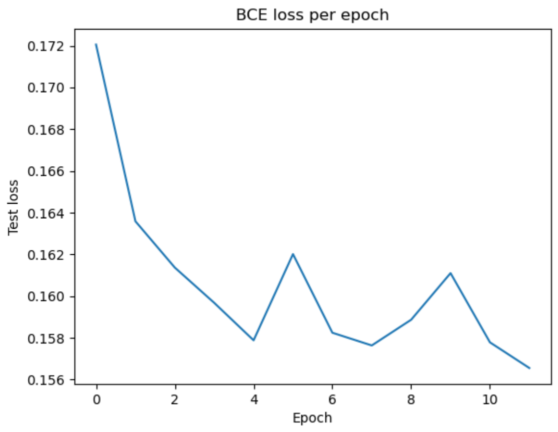


Figure 4: Test loss

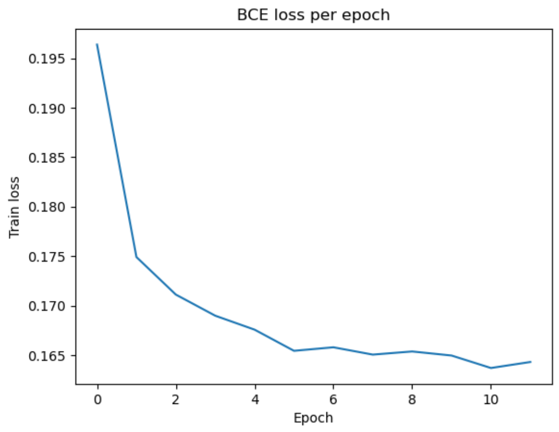


Figure 5: Train loss

Task 3:

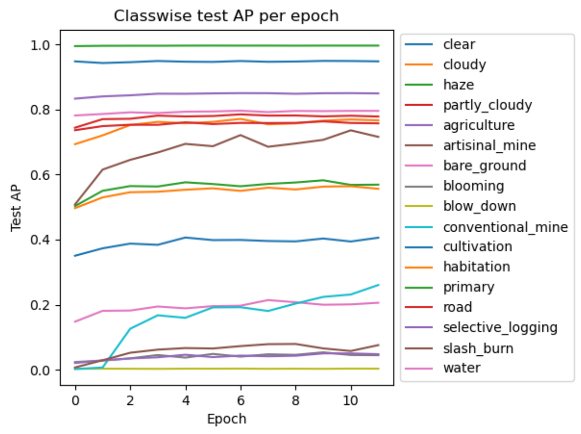


Figure 6: Classwise AP

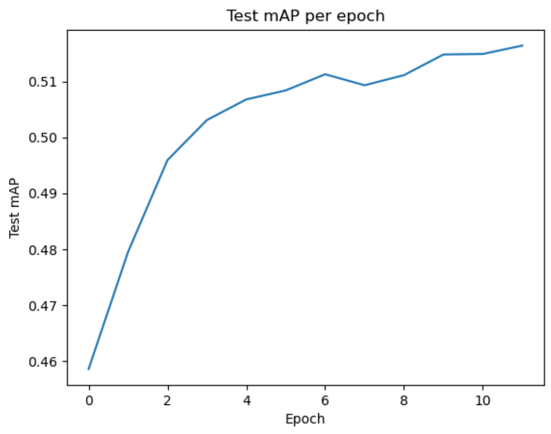


Figure 7: mAP

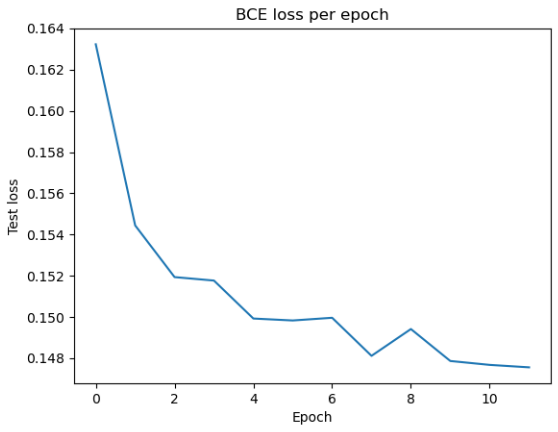


Figure 8: Test loss

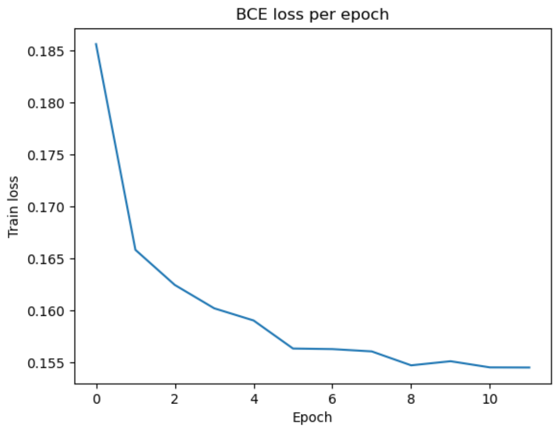


Figure 9: Train loss

Task 4:

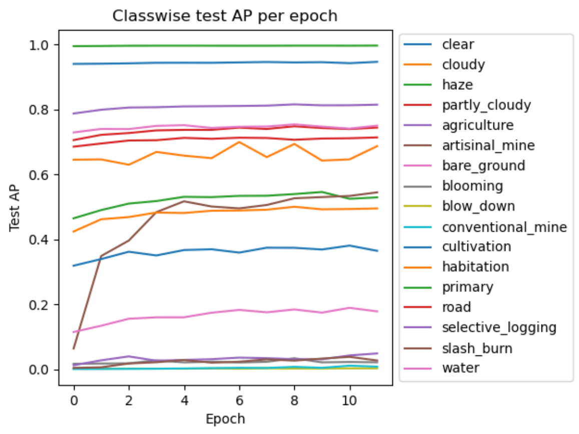


Figure 10: Classwise AP

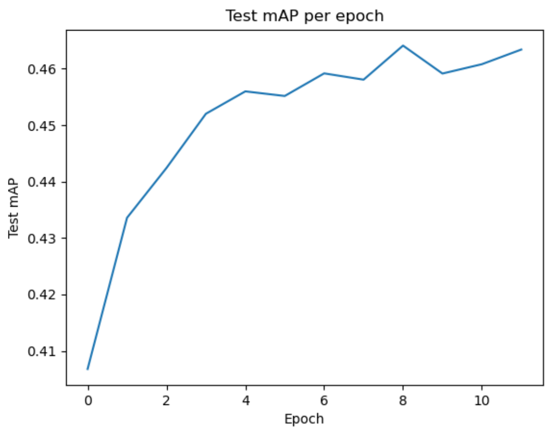


Figure 11: mAP

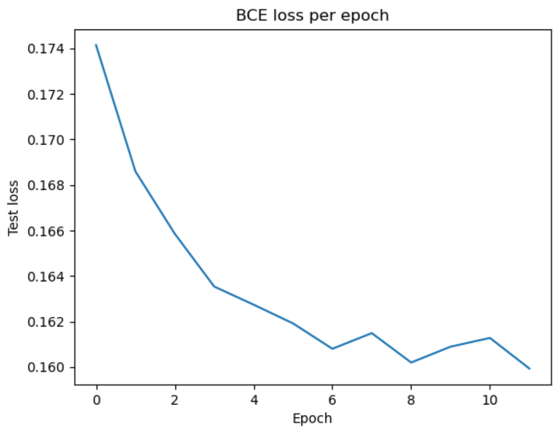


Figure 12: Test loss

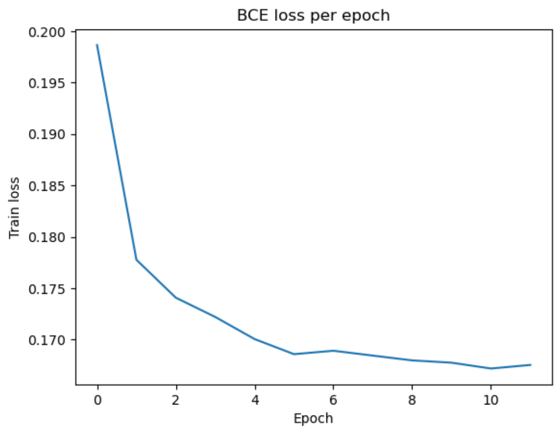


Figure 13: Train loss