Mandatory Assignment 1 - IN5400

Karl Henrik Fredly
(Dated: March 21, 2021)

PART 1: MULTI-LABEL PREDICTION

A. Setup

We loaded a resnet18 network pretrained on imagenet pictures. The aim was to use it for multi-label prediction on a different dataset. We trained the parameters in the entire network, and not just the final layer, as this gave the best results. This is not always the best practice for transfer learning, but it worked. We did however overwrite the final fully connected layer with random weights.

For the last layer in the network, we chose a fully connected linear layer with 20 sigmoid outputs, since we want to classify 20 separate things non-exclusively. The problem can be tackled like 20 binary logistic regression problems with the same inputs.

Which is why the binary cross entropy loss was chosen, as it lets each class be optimized seperately (like they would be in logistic regression). We used the standard binary cross entropy loss given in pytorch. The loss for each class is given by

$$l_n = -w_n[y_n * logx_n + (1 - y_n) * log(1 - x_n)]$$

and we take the mean over all classes.

We tried to make our results reproducible by following the steps at https://pytorch.org/docs/stable/notes/randomness.html, but it did not work.

The parameters for the training were the following:

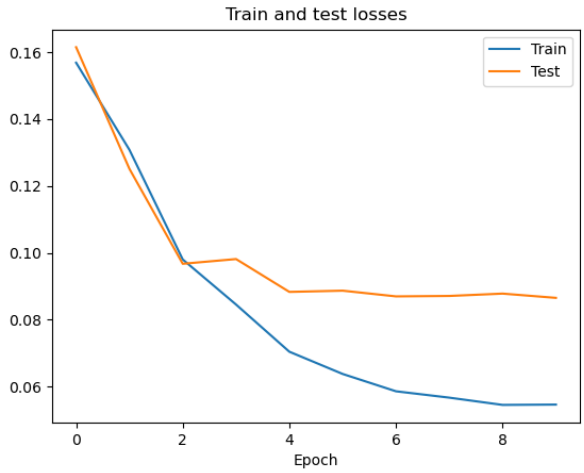
```
config["use_gpu"] = True
config["lr"] = 0.0005
config["batchsize_train"] = 16
config["batchsize_val"] = 64
config["maxnumepochs"] = 10

config["scheduler_stepsize"] = 2
config["scheduler_factor"] = 0.3
```

We used the optimizer Adam, and the StepLR learning rate scheduler.

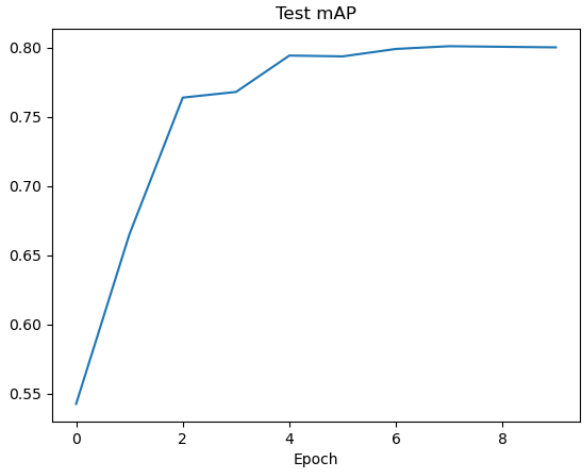
For the GUI we used PySimpleGUI, which is listed in requirements.txt and can be installed with pip.

B. Results

The losses during training are as follows:

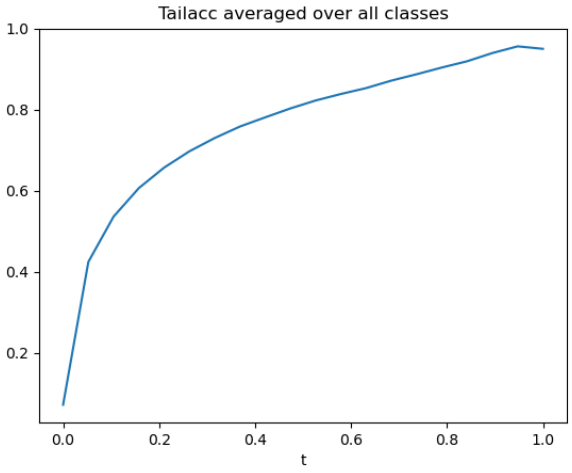


And the mAP:



This shows that the model improved quickly, and slowly got an overconfident training error, but without overfitting.

The tailaccuracy shown below demonstrates how the accuracy of the model was dependant on the threshhold t. The tailacc is the number of correct true predictions divided by the number of true predictions, where a true prediction is any prediction over the threshold t. With a high threshold, the model makes very few mistakes, but even with a small threshold almost half of the true predictions are correct. Which threshold is best depends on what you want from your model.



PART 2: CHANGING THE NETWORK STRUCTURE

In our network we have many convolutions followed by batchnorms. We wish to normalize the weights in the convolutions to possibly improve training. In order to not change how the network predicts we need to change the batchnorm in a way which nullifies the normalization in the forward pass.

A convolution can be written as

$$z = W * x$$

And the normalized convolution is given by

$$\hat{z} = \hat{W} * x = \frac{W * x}{n_c} = \frac{W * x}{\sqrt{std^2(W[c,:]) + \epsilon}}$$

where n_c is calculated for each output channel. The batchnorm can be written as

$$\begin{aligned} y(W, \alpha, \beta, \mu, \sigma) &= \alpha \frac{W * x - \mu}{\sqrt{\sigma^2 + \epsilon_2}} + \beta \\ &= \alpha n_c \frac{\hat{W} * x - \mu}{\sqrt{\sigma^2 + \epsilon_2}} + (\alpha n_c - \alpha)\frac{\mu}{\sqrt{\sigma^2 + \epsilon_2}} + \beta \\ &= \hat{\alpha} \frac{\hat{W} * x - \mu}{\sqrt{\sigma^2 + \epsilon_2}} + \hat{\beta} \end{aligned}$$

where we have substituted in $\hat{W}$ and "corrected" the expression by finding

$$\begin{aligned} \hat{\alpha} &= \alpha n_c \\ \hat{\beta} &= (\alpha n_c - \alpha)\frac{\mu}{\sqrt{\sigma^2 + \epsilon_2}} + \beta \end{aligned}$$

This method, changing α and β will be called case(A).

We can also write the batchnorm as

$$\begin{aligned} y(W, \alpha, \beta, \mu, \sigma) &= \alpha \frac{W * x - \mu}{\sqrt{\sigma^2 + \epsilon_2}} + \beta \\ &= \alpha \frac{n_c \hat{W} * x - \mu}{\sigma \sqrt{1 + \frac{\epsilon_2}{\sigma^2}}} + \beta \\ &= \alpha \frac{n_c}{\sigma} \frac{\hat{W} * x - \frac{\mu}{n_c}}{\sqrt{1 + \frac{\epsilon_2}{\sigma^2}}} + \beta \\ &= \alpha \frac{1}{\hat{\sigma}} \frac{\hat{W} * x - \hat{\mu}}{\sqrt{1 + \frac{\epsilon_2}{\sigma^2}}} + \beta \\ &= \alpha \frac{\hat{W} * x - \hat{\mu}}{\sqrt{\hat{\sigma}^2 + \frac{\hat{\sigma}^2}{\sigma^2}\epsilon_2}} + \beta \\ &\approx \alpha \frac{\hat{W} * x - \hat{\mu}}{\sqrt{\hat{\sigma}^2 + \epsilon_2}} + \beta \end{aligned}$$

where we have substituted in $\hat{W}$ and "corrected" the expression by finding

$$\begin{aligned} \hat{\mu} &= \frac{\mu}{n_c} \\ \hat{\sigma} &= \frac{\sigma}{n_c} \end{aligned}$$

The equality is not excact, as the stability term ϵ_2 gets in the way. This method, changing μ and σ will be called case(B).

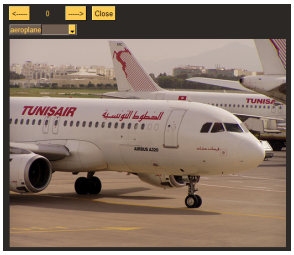
The average difference over predictions for a model with and without the normalization was $8.712e-05$ for case A and $3.395e-05$ for case B. This shows that the predictions of the network were mostly unaltered.

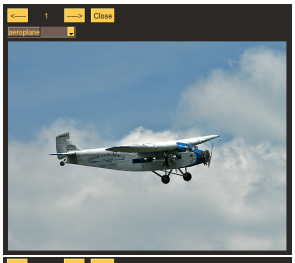
When training the network, the training was very slow and gave very bad results for both cases. This might be because I messed up the gradient flow in the network in some way. We were therefore unable to determine which method implemented properly would still allow for training. We'd guess that case B would work best, since the parameters we changed are not trainable parameters.

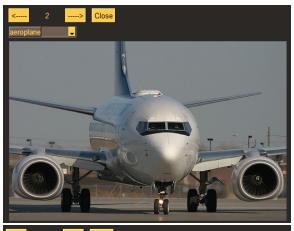
I. PART 3: GUI DEMONSTRATOR

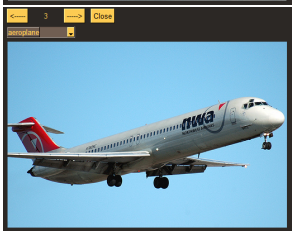
Results as shown by the GUI are shown below. The top predictions for all classes are all very good, even though we only have a mean average precision of 0.8. This is because the network is "better than 0.8" on data it is very sure of, as is also shown in the tailacc plot.

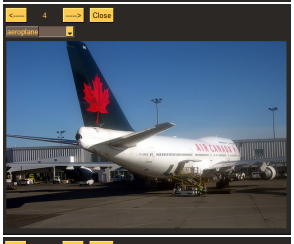
A. Aeroplane Top/Bot 10

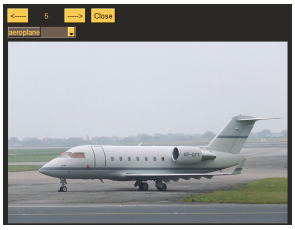


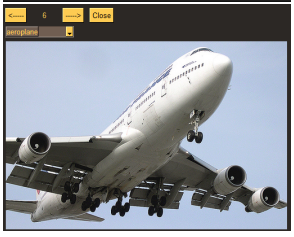


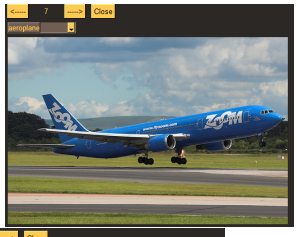


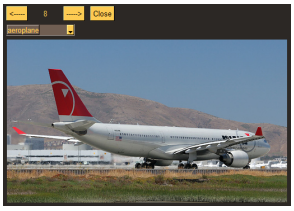


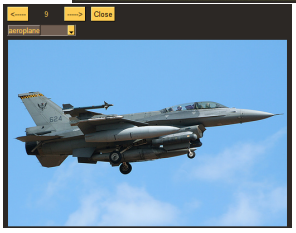


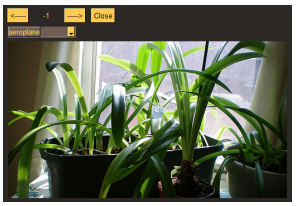


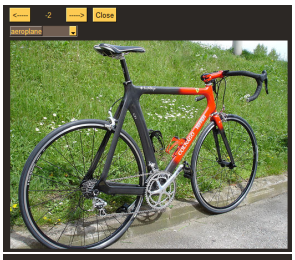


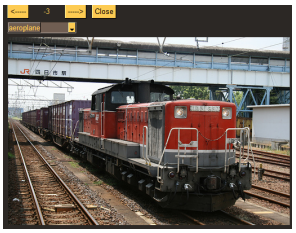


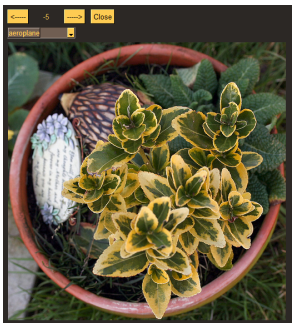


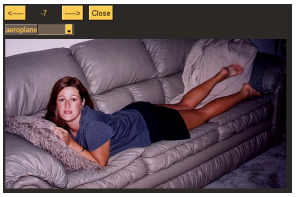


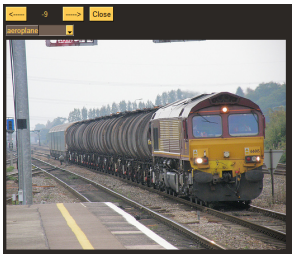


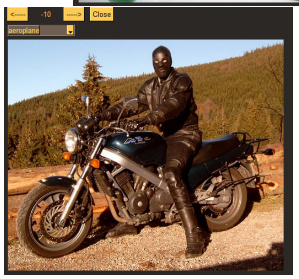




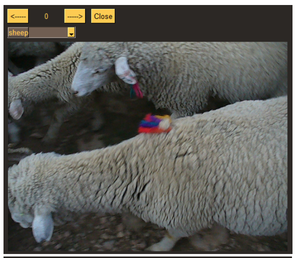


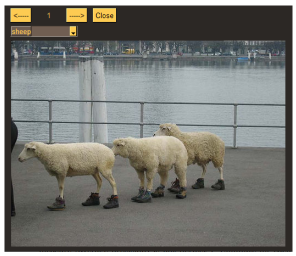


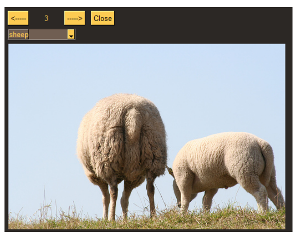


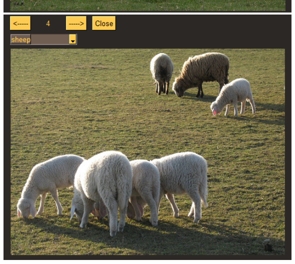


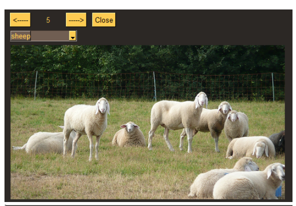
B. Sheep Top/Bot 10

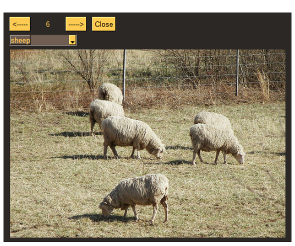


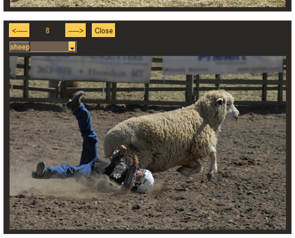


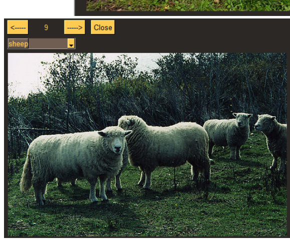


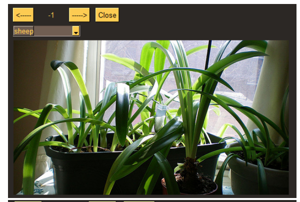


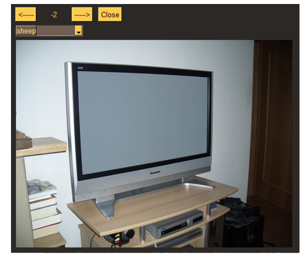


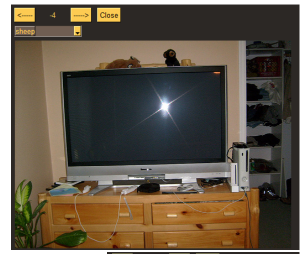


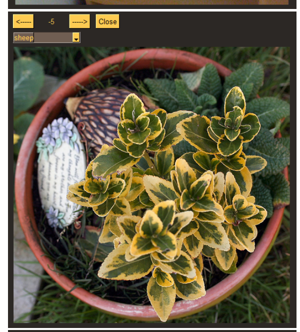


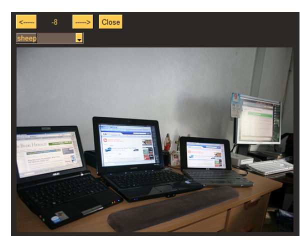




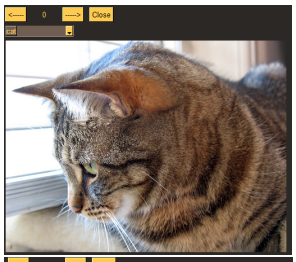
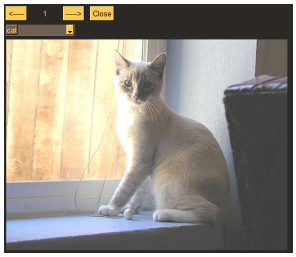
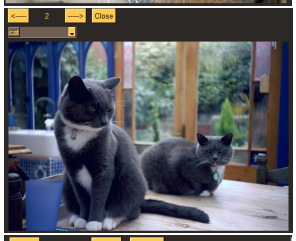
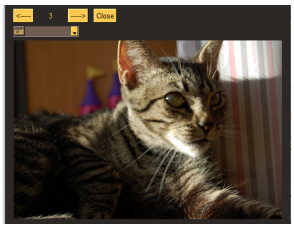
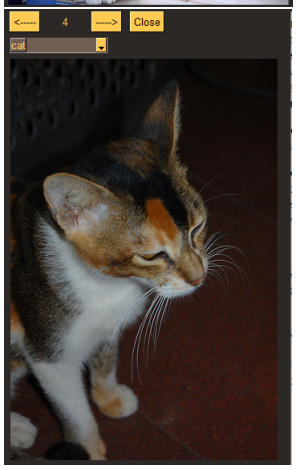
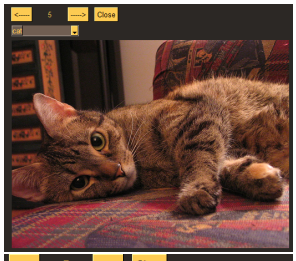


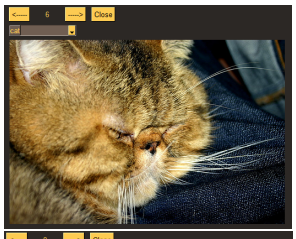
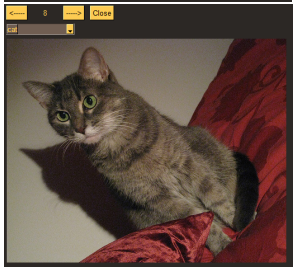
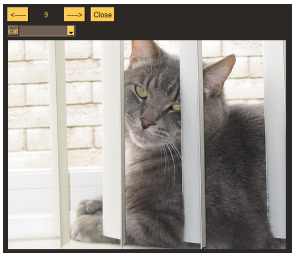


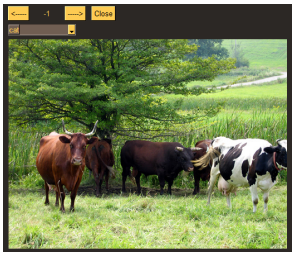


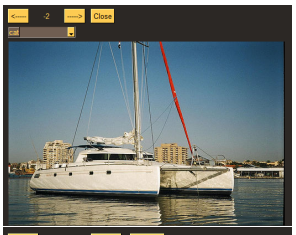


C. Cat Top/Bot 10 n

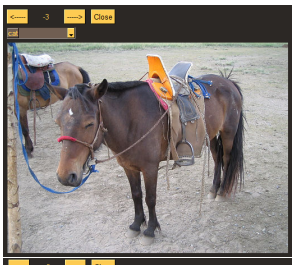







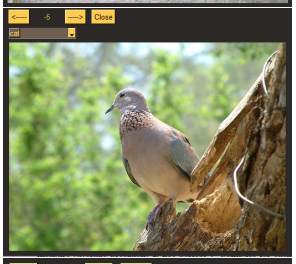




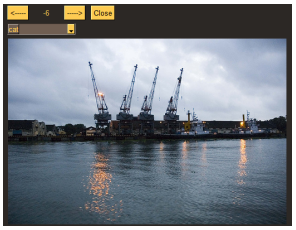

<---- -1 ----> Close
cat


<---- -2 ----> Close
cat

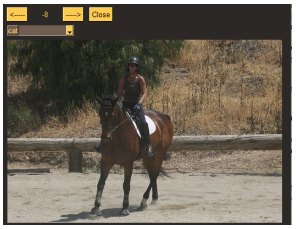

<---- -4 ----> Close
cat

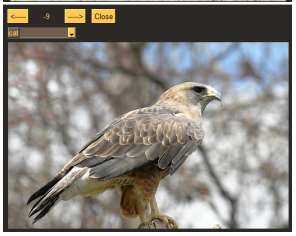

<---- -3 ----> Close
cat

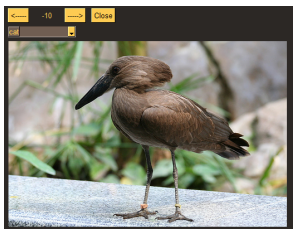

<---- -5 ----> Close
cat


<---- -6 ----> Close
cat


<---- -7 ----> Close
cat


<---- -8 ----> Close
cat


<---- -9 ----> Close
cat


<---- -10 ----> Close
cat