IN3310 2025 UiO - Exercise Week 5

1 Two exemplary ways of fine-tuning, setup with a separate validation set

- Take the 102 class flowers dataset and write a dataset class that can work with a train/val/test split for it. The difference to the 102 flowers from Oxford is that the train/val/test split is provided for your convenience. You can find the flowers dataset on the course page under exercises.
- Take any deep network you like that has pre-trained weights (a small ResNet or DenseNet)
- Train a deep neural network in three different modes
  - (A) without loading weights and training all layers.
  - (B) with loading model weights before training and training all layers
  - (C) with loading model weights before training and training only the last trainable layer (note: for quite some problems, the second approach is better than the third)

For each of these three modes, select the best epoch by the performance of the model on the validation set. Typically, less than 20 epochs should suffice for training when using finetuning. You can also run optionally a selection over a few learning rates if you use a GPU.

Typical steps to follow

- Write a new dataset class. You can either create three different instances of this class for train/val/test, each with its own path, or create one instance that takes the dataset type as an argument to the constructor and splits the data internally
- Use the following transforms (inside `__getitem__`) as part of preprocessing the images:

```
data_transforms ={
    'train': transforms.Compose([
```

```
        transforms.Resize(256),
        transforms.RandomCrop(224),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize([0.485,0.456,0.406],[0.229,0.224,0.225])
        ]),
    'val': transforms.Compose([
        transforms.Resize(224),
        transforms.CenterCrop(224),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize([0.485,0.456,0.406],[0.229,0.224,0.225])
        ]),
    }
```

Be aware that we are using the ImageNet mean and standard deviation.

- Create the data loaders for all datasets.
- Use some deep-learning models from the model zoo. Load its weights before training for settings B and C
- Adjust the number of classes in the last linear/dense layer
- In the optimizer, provide the proper parameters for training depending on the mode you are using (A, B, or C).
- Write a training regime and run it.
- Collect and plot curves of the training loss, the validation set loss, and the validation set accuracy. Also, print the final test accuracy of the selected model.

2 Spatial Output Size

- You are given a 2-dimensional convolution with feature map input size (78, 84). When using a kernel of size (5, 5) and stride 3 with padding of 2, what will be the spatial size of the feature map, which is the output of the convolution? Note that the spatial size does not depend on the number of input or output channels.
- You are given a 1-dimensional convolution. When using a kernel of size 9 and stride 3 with padding 1, which spatial input size do you need to have so that you have a spatial output size of 16?
- You are given a 2-dimensional convolution. When using a kernel of size (3, 5) and stride 2 with padding of 0, what will be the spatial size of the

feature map, which is the output of the convolution, which spatial input size do you need to have, so that you have a spatial output size of (128, 96)?

3 Number of Trainable Parameters

How many trainable parameters are in

- a 2-D convolutional layer with input (32, 19, 19), kernel size (7, 7), stride 3, 64 output channels?
- a 2-D convolutional layer with input (512, 25, 25), kernel size (1, 1), stride 1, 128 output channels?
- How many multiplications and how many additions are performed in the first case above?