Deep Learning with MatConvNet Practical Guide II

Joost van de Weijer, Marc Masana, German Ros

In this second practical sessions we will train a CNN from scratch (without any pretrained network). Since we have only little training data we will need to design a smaller network. There is a large danger of overfitting with little training data. There are several techniques to prevent overfitting:

- Design a network with less variables,
- Increase the weight-decay (preventing weights of becoming large),
- Batch normalization which has been shown to improve convergence speed and reach better minima,
- Use dropout which forces the network to not be dependent on a few nodes,
- Perform data augmentation.

During the next session you are to experiment with all of these.

I. Design a new network

Unzip the tarball *DL_session2.zip* in the same matlab directory as the matlab files of last week. Copy file *BatchNorm.m* to directory: /matconvnet-1.0-beta17/matlab/+dagnn/. Overwrite the existing *BatchNorm.m* file.

Start by opening *session02_script.m*. You will design your own network. At the bottom you see several example networks.You will have to take a number of decisions.

- Which layers do you want to combine in your network.
- How do you choose the filter size, padding, and stride to reduce the number of parameters in the higher layers (use the mynet.print function to assess the number of parameters of your network).
- How many filters and feature maps do you want at every layer.
- Experiment with learning rate and weight-decay to find a configuration which converges (that is a good starting point).

NETWORK 1: mini ALEX	NETWORK 2: mini VGG	NETWORK 3: mini NinN
1.conv 5x5 2.relu 3. max-pool 4.conv 5x5 5.relu 6.max-pool 7.fc	1.conv 3x3 2.relu 3. max-pool 4.conv 3x3 5.relu 3. max-pool 6.conv 3x3 7.relu 3. max-pool 8.fc	1.conv 5x5 2.relu 3. max-pool 4.conv 1x1 5.conv 3x3 6.relu 7.max-pool 8.fc

EXERCISE 3: The main exercise of this week is design your own network. We are interested to see what is the best possible score you can get without any pre-training. It is forbidden to use any data other than the training set and to use any pre-trained networks. You are allowed to play with the input image size.

II. Initialization, dropout and Batch-normalization

Go through the steps in *session02_script.m* and look at the example code for dropout and batch-normalization.

The dropout can be introduced as follows in your network (e.g):

```
net.addLayer('drop1', dagnn.DropOut('rate', 0.5), {'pool1'}, {'drop1'}, {});
```

Make sure to change the output layer name of the layer before the dropout layer, and the input layer name of the layer before. The rate indicates the percentages of neurons which will be set to zero.

Batch-normalization can be introduced as follows in your network:

```
net.addLayer('bn2', dagnn.BatchNorm('numChannels', 24), {'pool2'}, {'bn2'}, {'bn2f',
'bn2b', 'bn2m'});
```

Again change the names according to its position in the network. Important is also to correctly fill in the number of channels of the input (here 24). Keep in mind that batch-normalization makes more sense before layers which have many weights.

EXERCISE 4: Change the initializations. Did any work better for you ? Show the effects of applying dropout and batch-normalization to your network. What positions in your network you found optimal to introduce these layers ? What about stochastic gradient descent with momentum ?

III. Homework

As homework for next week Monday 11/4 (deadline at 9:00) we would like you to submit a short presentation in which you show your results for the different exercises (the exercises with number 1-4). Also include a copy of your best network (mynet_train.m).