Homework 3 – ConvNets

For this homework, rely on functions from tensorflow's `tf.layers` and `tf.nn` toolbox where possible.

1) continue to practice building and running tensorflow graphs

. . . try advanced things, e.g.: create a graph, save it, and load it twice under different names. Train one of them. Copy the contents of variables from one to the other. Etc.

2) training data

We will continue using the data from homework 2.

3) 1D Convolutional Neural Network

Implement a routine that, given values (k, l), creates a 1D multilayer convolutional neural networks (ConvNet) with the following architecture:

- input size 768
- k convolutional layers, each consisting of
  - 32 convolution filters of spatial size 5 (but taking into account all channels) with no padding
  - ReLu activations
  - per-channel max-pooling over a neighborhood of size 3, applied with a stride of 2
- l fully connected layer (connected to all inputs) with 100 outputs and ReLU activation function
- one fully connected layer with 10 outputs
- 10-class *softmax* output

Hint: you can add dimensions to a tensor using `tf.expand_dims`.

Hint: functions sometimes change packages between different versions of tensorflow. For example, depending on which version of tensorflow you use, you might want to look up either `tf.contrib.layers.flatten` or `tf.layers.flatten`

4) training

Create 1d ConvNets for all combinations of $k = 0, \ldots, 5$ and $l = 0, \ldots, 3$. Train each of them for 10 epochs with *cross-entropy* loss on the data as loaded in 1). Use any optimization method and parameter choice you like.

What's the best validation error you can achieve?

5) 2D Convolutional Neural Network (ConvNet)

Implement a routine that given values (k, l) creates 2D multilayer convolutional neural networks (ConvNet) with the following architecture:

- input size 28x28
- k convolutional layer, each consisting of
  - 32 2d-convolution filters of size $5x5$ (but taking into account all channels) with `SAME` padding
  - ReLU activation function
  - per channel max pool over a neighborhood of size $3x3$, applied with a stride of 2, also with `SAME` padding
- l fully connected layer with 100 outputs and ReLU activation function
- one fully connected layer with 10 outputs
- 10-class *softmax* output

6) training

Convert the data from 1) into 2-dimensional inputs by first padding 10 constant zero entries at the beginning and 6 constant zero entries at the end of each data vector, and then reshaping it such that each data point has size 28x28.

Create 2d ConvNets for all combinations of $k = 0, \ldots, 3$ and $l = 0, \ldots, 3$. Train each of them for 10 epochs using *cross-entropy* loss. You can use any optimizer and parameter settings you like.

What's the best validation error you can achieve?

Hint: It should be lower than the one from homework 2.

7) descriptive analysis

Answer in one sentences each: What differences do you observe between last week's network, 3)/4), and 5)/6)? What do you suspect is the reason? What could be a procedure to test if your suspicion is correct?

8) quantiative analysis

Take the first 10 elements of the $28x28$ data you created in 6) and visualize them using `np.matshow`. What do you see?

Create a new training set by randomly shuffling the column of the downloaded data before applying the padding and reshaping of 6). Again visualize the first 10 elements. What is the difference?

Train the network from 5) on this shuffled data instead of the original one. What changed compared to 6)?

9) (optional) a bit of fun

tensorflow has a surprising number of `softmax` routines, distributed over different packages and with input arguments of different formats. How many can you find?

The `tf.nn.softmax_cross_entropy_with_logits` has a peculiar signature: its first argument is a `_sentinel` that defaults to `None` and prevents the use of arguments without specifying their names. Try to find out (from any source you like) why that is the case.

Hand-in requirements

1. upload your code to 3)-6),8) with reasonable parameters to the IST *git* server
2. pick exactly **one** of your trained models and use it to predict labels for the data available at:

   `https://cvml.ist.ac.at/courses/DLWT_W18/data/hw2-test-data.npy`
3. write the resulting class predictions to a file "hw3-test-labels.txt" (one number 0-9 per line in the same order as the test examples) and upload it.

To fulfill the requirement, your results have to be better than the ones from homework 2.