Practicals to Convolutional Networks

Aufgabe P 2. *Convolutional Networks, due 14 December in class*

In this practical you will implement a convolutional neural network (CNN) for image classification in the CIFAR10 dataset containing $32 \times 32 \times 3$ images. The data import is provided in the stubs.

(a) Define the graph of the CNN with following convolutional, pooling, fully-connected layers:

  i) Convolutional layer with filter size $[3, 3]$, no strides, padding 'same', 32 feature maps, and a bias. Apply the rectified linear activation function (tf.nn.relu).

  ii) Convolutional layer with filter size $[3, 3]$, no strides, padding 'same', 32 feature maps, and a bias. Apply the rectified linear activation function (tf.nn.relu).

  iii) Max pool layer with filter size of $[2, 2]$, strides of 2 for the width and height of the image, padding 'same'.

  iv) Convolutional layer with filter size $[3, 3]$, no strides, padding 'same', 64 feature maps, and a bias. Apply the rectified linear activation function (tf.nn.relu).

  v) Max pool layer with filter size of $[2, 2]$,strides of 2 for the width and height of the image, padding 'same'.

  vi) Fully connected layer with output of 512 neurons. Input is the output of max-pool layer reshaped to a vector.

  vii) Fully connected layer with output of 10 neurons. Input is the output previous fully connected layer.

  viii) Use the $tf.losses.sparse\_cross\_entropy$ as loss. Use the sparse version of cross-entroy since the labels are imported as numbers and not as one-hot-encoded vectors. Implement the accuracy for the classification.

  ix) Train the model with tf.train.AdamOptimizer and try different learning rates for 10 epochs and 20 epochs. The stubs contain tensorboard loggings which you should turn in together with the accuracy in training and test.

(b) Introduce a drop-out layer after the first fully connected layer with a keep probability of 0.5. Use your pretrained network and start a refinement training. Is the test accuracy better than the previous one?

Note: initialize the filter variables with $tf.contrib.layers.xavier\_initializer\_conv2d()$ and the bias variables with $tf.constant\_initializer(0.0001)$.