Lab: CudaVision – Learning Vision Systems on Graphics Cards (MA-INF 4308)

Assignment 4

20.6.2016

Prof. Sven Behnke
Dr. Seongyong Koo

universitätbonn ais

Convolutional Neural Networks with MNIST dataset

- **Goal: How to build a Convolutional Neural Network (CNN) Models with TensorFlow**

- **Let's learn basic components to build CNN layer**
  `tf.reshape(), tf.nn.conv2d(), tf.nn.bias_add(), tf.nn.relu(), tf.nn.max_pool()`

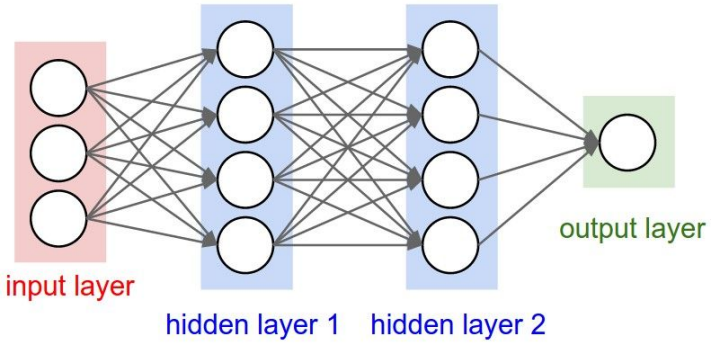
- **Design parameters**
  - patch_size, strides, padding, number of filters

- **Your task is,**
  - Build your CNN models
  - Find the best performance of MNIST classifier
  - Visualize trained filters

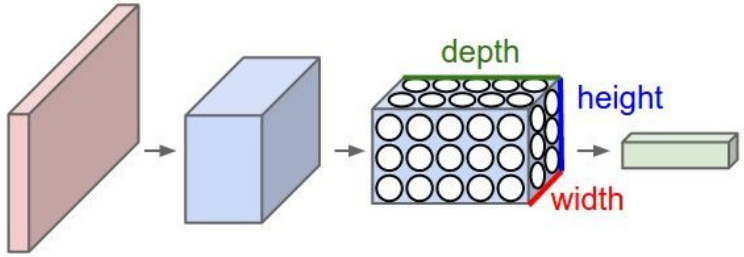
- **Theoritical reference**
  - http://cs231n.github.io/convolutional-networks/
  - http://www.deeplearningbook.org/contents/convnets.html

Convolutional Networks (ConvNet)

- Full connectivity in regular neural networks don't scale well to full images.
- Unlike a regular NN, the layers of a ConvNet have neurons arranged in 3 dimensions: width, height, channel.



[A regular 3-layer NN]



[A ConvNet]

- Three main types of layers to build ConvNet: Convolutional Layer, Pooling Layer, and Fully-connected layer

http://cs231n.github.io/convolutional-networks/

Example

- Input raw rgb pixel values of the image : [32x32x3]
- CONV layer will compute the output of neurons that are connected to local regions in the input : [32x32x12], if use 12 filters.
- RELU layer will apply an elementwise activation fulction [32x32x12]
- POOL layer will perform a downsampling operation along the spatial dimensions : [16x16x12], if use window size 2
- FC (fully-connected) layer will compute the class scores [1x1x10]

- Learning parameters
  - Weights and biasses in CONV/FC layers

- Hyper parameters of a ConvLayer
  - The number of filters
  - Filter size
  - Stride size: the number of pixels to move the filters at a time
  - Zero-padding size

http://cs231n.github.io/convolutional-networks/

1 ConvLayer implementation for MNIST (24 x 24 x 1) data

- in CNNmodel_single_MNIST.ipynb

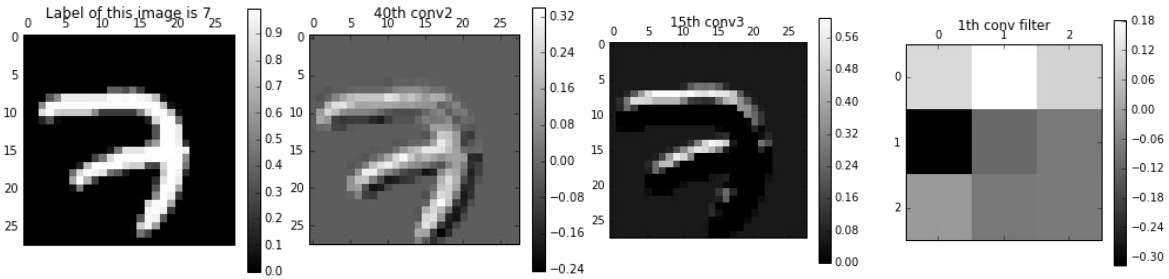
```
n_input_width = 28
n_input_height = 28
n_input_channel = 1

n_conv1_patch_size = 3
n_conv1_filter = 64

n_output  = 10 # e.g. MNIST total classes (0-9 digits)

# tf Graph input
x = tf.placeholder(tf.float32, [None, n_input_width * n_input_height])
y = tf.placeholder(tf.float32, [None, n_output])

wc1 = tf.Variable(tf.random_normal([n_conv1_patch_size, n_conv1_patch_size, n_input_channel, n_conv1_filter], stddev=0.1))
bc1 = tf.Variable(tf.random_normal([n_conv1_filter], stddev=0.1))

wf1 = tf.Variable(tf.random_normal([(n_input_width/2)*(n_input_height/2)*n_conv1_filter, n_output], stddev=0.1))
bf1 = tf.Variable(tf.random_normal([n_output], stddev=0.1))

# Reshape input
input_r = tf.reshape(x, shape=[-1, n_input_width, n_input_width, 1])
# Convolution
conv = tf.nn.conv2d(input_r, wc1, strides=[1, 1, 1, 1], padding='SAME')
# Add-bias
bias = tf.nn.bias_add(conv, bc1)
# Pass ReLu
relu = tf.nn.relu(bias)
# Max-pooling
pool  = tf.nn.max_pool(relu, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
# Vectorize
dense = tf.reshape(pool, [-1, wf1.get_shape().as_list()[0]])
# Fully-connected layer
out = tf.add(tf.matmul(dense, wf1), bf1)

cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(out, y))
```

Data reshaping

- `tf.reshape(tensor, shape, name=None)`
  - Given tensor, this operation returns a tensor that has the same values as tensor with shape shape.
  - If one component of shape is the special value -1, the size of that dimension is computed so that the total size remains constant. In particular, a shape of [-1] flattens into 1-D. At most one component of shape can be -1.
  - If shape is 1-D or higher, then the operation returns a tensor with shape filled with the values of tensor. In this case, the number of elements implied by shape must be the same as the number of elements in tensor.

Assignment 4

- **Finding your best CNN model for MNIST classification**
  - Train the given single ConvNet with MNIST dataset
    - Load MNIST
    - Construct a optimizer using `tf.train.AdamOptimizer`
    - Train the model by changing learning_rate, training_epochs, batch_size, the number and size of filters,
    - What is the test accuracy of the given model ? (I got 0.979)
  - Build and train deeper CNNs with `tf.nn.dropout`
    - Add more Conv layers
    - Add dropout function after each pooling layer
  - What is your best performance with a multi-layered CNN?
- Visualize images of input image, output image of each Conv filter, weight images of each Conv filter



universitätbonn ais