Introduction to Neural Networks 67103

Assignment 1: CNN

Due: 2/12/2019 at 23:59

Practical Questions

In this assignment you will implement a simple Convolutional Neural Network (CNN), train it to perform a simple classification task, and experiment with different architectures and hyper-parameters. Your implementation and experiments should be done using TensorFlow 2. As a basis you should start from a simple CNN-based MNIST classifier as described in this tutorial (MNIST database can be found here).

1. **Train and Test Error.**
    1.1. Create a new model for the MNIST classification task, with the following architecture:
    - convolution layer: 5*5 kernel, 32 filters
    - 2*2 max pooling
    - convolution layer: 5*5 kernel, 64 filters
    - 2*2 max pooling
    - Fully connected layer: output shape 1024
    - Fully connected layer: output shape 10

    (*use ReLU for the activations in the convolutional and fully connected layers)

    (*other arguments should be left as default, e.g. stride in the convolutional layers is 1)

    This model is able to achieve an accuracy of above 97% after 20000 iterations. **How many free parameters does this network have?** Count manually, and show your steps.

    1.2. The training process is often visualized/monitored by plotting the training accuracy and the test accuracy as a function of the number of training iterations. **Plot these accuracy measures (perhaps using a data point every 500 iterations).**

2. **Linear vs. Non-Linear**.
    2.1. Identify what layers/components of the network are non-linear and remove them

(except for the logits/softmax in the loss function, also assume the max-pooling is a uniform stride - or change it to one if you wish).

2.2. Compare the performance of this linear network to its original non-linear counterpart.

2.3. What is the effect on the training and the final test accuracy of the network?

3. **Deep vs. Shallow**.

3.1. Construct a shallow CNN consisting of a single convolutional layer, followed by a fully connected layer, such that it is able to achieve accuracy above 95% using the smallest number of free parameters.

3.1.1. How many parameters does your network have? (explain how you count them)

3.1.2. Plot the loss function of the training of this network

3.2. Reduce the number of parameters of the original deeper network as much as you can, while still being able to achieve 95% accuracy.

3.2.1. How many parameters does this network have? (explain how you count them)

3.2.2. Plot the training of this network.

3.3. Compare the training speed (time to convergence) of the two networks and explain your findings

4. **Overfitting.**

4.1. Train the original network (without dropout, i.e., parameter rate in the code should be set to 0) over the complete set of the examples as well as a reduced set of only 250 examples. Plot the training and testing accuracies as well as their ratio. In order to reduce the over fitting of the reduced training dataset, reduce the network's size (fewer parameters) until a higher ratio and test accuracy is obtained. Report the reduction and size and gain in performance in this experiment. As an alternative insert **dropout layers** to the outputs of all the hidden layers. Report which keep-probability resulted in the best ratio/test accuracy.

A dropout layer randomly blocks neurons by setting them to zero and transfers the rest at probability *keep_prob*. It is used to cause the network to learn the filters with redundancies (let one neuron compensate for the possible loss of others), thus effectively reducing the number of neurons in the network, or alternatively reducing the number of "free" parameters. We will discuss this layer in a future class.

5. **Two Digits Sum.**

5.1. Design, implement and train a CNN architecture that takes as input two MNIST images (concatenated horizontally or vertically) and predicts their sum.

5.1.1. Plot the training steps (train and test of each iteration) of this network.

5.2. Design, implement and train a cnn architecture that takes as input two MNIST images, predicts each number separately and sums the two numbers. Notice that the network should predict a pair of labels (a vector of dimension 2). This should be done in a single forward pass through the network.

5.2.1. Plot the training steps (train and test of each iteration) of this network.

5.3. Which strategy worked better and why?

Theoretical Questions

1. **Convolution.**

   1.1. Show that convolution is indeed a translation-invariant operation, i.e., (f[k+t]*g[k])[n] = (f[k]*g[k])[n+t] and (f[k]*g[k+t])[n] = (f[k]*g[k])[n+t]. Express the convolution operation as a dot-product (and its translations).

   1.2. When convolving a signal of length n with a filter of length k<=n and discarding all the cases where the signal is not fully included in the support of the filter,

   1.2.1. what is the length of the output signal?

   1.2.2. What would that be when n=k?

2. **ReLU.** Assume the filter f and the bias b produce a negative response on an example datapoint I, i.e., (f*I)[n]+b<0 for all n. Assume the next layer is a RelU layer, followed by the remaining layers and loss, expressed together by F(.), i.e., the network's loss is given by F(RelU(f*I+b)).

   2.1. What would be the partial derivative of this loss term with respect to f and b?

   2.2. What would happen to these filters and biases in a gradient-descent process consisting solely on examples which produce negative responses?

3. **Backpropagation.**

   3.1. Compute the partial derivative of the convolution layer, f*I with respect to the filter f (express the complete matrix whose columns correspond to the filter elements, and rows to the output image pixels - row per pixel - big matrix!).

   3.2. Explain how this matrix is efficiently multiplied by the gradient vector of the previous layer being propagated through this layer.

   - You may assume 1D signals, and that the convolution layer discards points in which the filter f is not fully included within the support of the signal I.

4. **Parameters vs. Constraints.**

   4.1. How many parameters are involved in a convolutional layer (+ its biases) consisting of 96, 5-by-5 pixel filters that operate on an RGB image?

   4.2. How many parameters are involved in a fully-connected layer (plus its biases) mapping 4-by-4 images of 256 channels (reshaped into a vector) to the same dimension?

   4.3. Assuming each loss term is fulfilled, i.e., rather than minimizing the loss, we set each loss term as an equation (or constraint). How many constraints a dataset of 10000 images with 1000 categories produce?

Submission

Submit (via moodle) a ZIP archive file named:
”Ex1 _(student1name)[_student2name].zip"

For example, if your names are Yann LeCun and Alex Krizhevsky, the file name should be:

“Ex1_YannLeCun_AlexKrizhevsky.zip”

The ZIP file should contain only your code (only .py files), and one PDF document named:

”Ex1 _(student1name)[_student2name].pdf"

which clearly describes your solutions and reports your results.

Preferably, create a file called “Ex1_main_(student1name)[_student2name].py” which is a compact file, with a function for every answer or plot included in your pdf report.

Remarks:

- The "”Ex1 _(student1name)[_student2name].pdf" document should be a pdf, not a docx. Note that only one PDF is to be submitted, additional documents will not be checked.
- If you are doing the assignment in pairs, only one of you should submit one zip file.
- Please write your names inside the pdf file.
- Please write some documentation explaining your code.
- Please don’t submit png files.

Grading

- Practical questions, 12 points each
- Theoretical questions, 10 points each