Deep Learning laboratory Report

Assignment 2

Mohamed AbouHussein
Louay Abdelgawad

Department of Machine Learning
University of Freiburg
November 2016

Summary

This report is related to the work of implementing a convolutional neural network using `tensorflow as tf` for image label classification. The report is divided into 3 sections as follows: Section 1 represent a simple explanation of the implementation, section 2 represents the designed convolutional neural network and section 3 represents the results.

Report

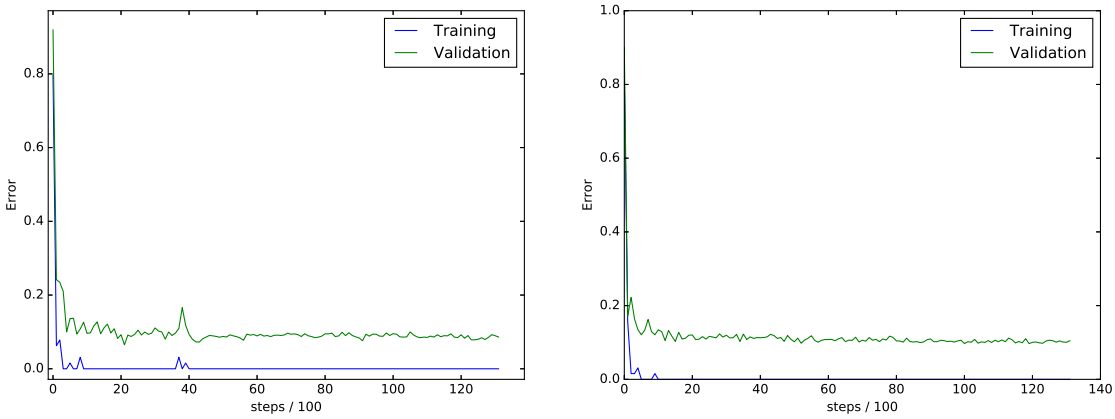
1 Convolutional Neural Network Implementation

The algorithm begins by loading the `data_input` and saving the input and their respective output in variables. After that, `tf.placeholder` nodes are created for the `train_data_node`, `train_labels_node` and `eval_data`. We then define functions to create the `weight_variable`, `bias_variable`, `conv2d` and `max_pool_2x2`. Afterwards, we initialize the weight and the bias variables of our model (to be mentioned in section 2). We created a `model` function to use it afterwards to define the training and validation model. Later, we define the `AdamOptimizer` and we use the softmax output for the output layer. Finally we iterate on the train batches and the validation dataset. Results are presented in section 3.

2 Designed Convolutional Neural Network

For the design of the conv network, the procedure utilized was to create a single convolutional layer and pool layer at a time. The layer parameters was randomly tried out from a range of values. The best estimated random values was the one implemented. Afterwards, another layer is added and the same procedure is tried out. In addition, the hypermaraters is also set in similar manner. The previous procedure has resulted in the following: The first conv layer included 32 `CONV_LAYER1_FEATURES`, 6x6 `filter_size` and finally a 2x2 `max_pool` is used. As for the second layer, it included 64 `CONV_LAYER2_FEATURES`, a 6x6 `filter_size` and also a 2x2 `max_pool`. The final `fully_connected_layer` layer is of size 1024. `softmax` function is used on the output layer. `AdamOptimizer` is selected as the optimization algorithm for gradient descent. The `epoch_size` selected was 30. The results of the conv network is discussed in section 3.

3 Results and Performance Evaluation



(a) 3 Channels Validation vs Training error

(b) 4 Channels Validation vs Training error

Figure 1: Training and validation error for different convolutional neural networks

The same CNN described in the 2 was trained two times once with 3 Channels (RGB) as seen in 1a, and the other time considering the Depth factor as seen in 1b. Both the 3 Channels and 4 Channels CNNs resulted in 0% `training_error` in about 1000-1500 steps. In addition, both of them resulted in the same `validation_error` range; around 8% and 10%. The difference between the two is noticed in the `test_error`, where the 3 Channels model resulted in a `test_error` of 9%, however the 4 Channels model resulted in a `test_error` of 4%. It can be interpreted that adding an extra input dimension (the depth channel) has resulted in a significant improvement in the result, about 5%.