In and delete all those data points

In my experiment, I train a multilayer CNN for street view housenumbers recognition and check the accuracy of test data.

The coding is done inpython using Tensorflow, a powerful library for implementation and trainingdeep neural networks. The central unit of data in TensorFlow is the tensor. Atensor consists of a set of primitive values shaped into an array of any numberof dimensions. A tensor’s rank is its number of dimensions. 20 Along withTensorFlow used some other library function such as Numpy, Mathplotlib, SciPyetc. Firstly, as I have technical resource limitation I perform my analysisonly using the train and test dataset.

And omit extra dataset which is 2. 7GB. Secondly, to make the analysis simpler I find and delete all those data pointswhich have more than 5 digits in the image. For the implementation, I randomlyshuffle valid dataset I have used the pickle file svhn_multi which I created bypreprocessing the data from the original SVHN dataset.

Then used the picklefile and train a 7-layer Convoluted Neural Network. Finally, I cast off thetest data to check for accuracy of the trained model to detect number fromstreet house number image. At the verybeginning of my experiment, first convolution layer I used 16 feature maps with5x5 filters, and originate 28x28x16 output. A few ReLU layers are also addedafter each layer to add more non-linearity to the decision-making process. After first sub-sampling the output size decrease in 14x14x10.

The secondconvolution has 512 feature maps with 5×5 filters and produces 10x10x32 output. In this moment applied sub-sampling second time and

shrink the output size to5x5x32. Finally, the third convolution has 2048 feature maps with same filtersize.

It is mentionable that the stride size = 1 in my experiment along withthis zero padding also used here. During my experiment, I used dropouttechnique to reduce the overfitting. Finally, the last layer is SoftMaxregression layer. Weights are initialized randomly using Xavier initializationwhich keeps the weights in the right range. It automatically scales theinitialization based on the number of output and input neurons. Now I train thenetwork and log the accuracy, loss and validation accuracy in steps of 500. Initially, we used a static learning rate of 0.

01 butlater on switched to exponential decay learning rate with an initial learningrate of 0. 05 which decays every 10000 steps with a base of 0. 95. Also usedAdagrad Optimizer to minimize loss. We stop learning when we reach adequateaccuracy level for the test dataset and we save the hyperparameters incnn_multi checkpoint file so that it can be loaded later when we need toperform detection without training the model again.

Refinement The initialmodel produced an accuracy of 89% with just 15000 steps. It’s a great startingpoint and certainly after a few hours of training the accuracy will reach mybenchmark of 90%. However, I further made some simple improvements to furtherincrease the accuracy of few number of learning steps. 1. Added a dropout layerto the network after the third convolution layer just before fully connectedlayer, which randomly drops weights from the network with a keep probability of0.

9375 to add more redundancy to the network. This allows the network to becomemore robust and prevents overfitting. 2. Introduced exponential decay tolearning rate instead of keeping it constant. This helps the network to takebigger steps at first so that it learns fast but over time as we move closer tothe global minimum, take smaller noisier steps.

With these changes, the modelis now able to produce an accuracy of 92. 9% on the test set with 15000 steps. Since there are 230070 images in training set and about 13068 images in thetest set, the model is expected to improve further if it is trained for alonger duration.