Lab 5 Exercise - A little Linear Regression

1. Introduction

This exercise is to study an application of CNN to a little linear regression—given a scatter plot whose scatter points are drawn from an underlying line, predict the parameters of the underlying line on each image.

To begin with, it is necessary to specify a loss function. On this task, the output is a 2-dim vector and our task will be to bring our predictions close to the targets. Therefore, Mean Square Error will be a decent choice since it simply measures the difference i.e. distance between two vectors.

2. Initial Attempt

In this attempt, a simple CNN is implemented which mainly composes of a convolution layer and 2 full connected layers. During the training stage, there is a continuous decrease in training set error from 62.37 initially to 2.23 at the end of 100th epoch. However, the validation set error descends following it from around 53 to 22 until half way, and dramatically switch to fluctuating, finally turning out 23.28. As for the test set error, it is 17.19 slightly lower than val-error. From the difference between training error and val-error, we can infer that there is a severe overfitting.

From the visualised results, the most of rebuilt lines moderately deviates their targets and a small part slightly. The average deviation is apparent.

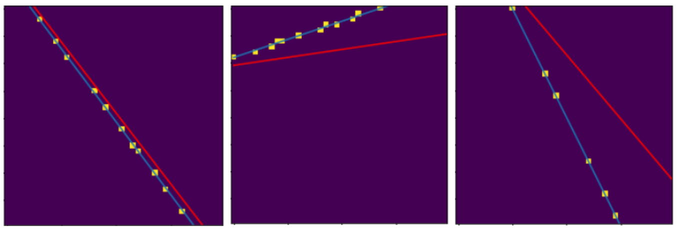

Figure 1: Examples of Rebuilt Line by the Initial Attempt

3. CNN with Global Pooling

With a max pooling layer in front of full connected layers, a CNN is able to yield better results. The training error falls from 59.85 to 3.77 not much different from the previous one. On the other hand, validation error reaches about 4.2 by 77th epoch and begins fluctuates, which is way more less than simple CNN's counterpart. Likewise, test error declines by 12.28, which is a great progress. Because of the difference between val-set and test set results, it can be inferred that the overfitting is significantly mitigated.

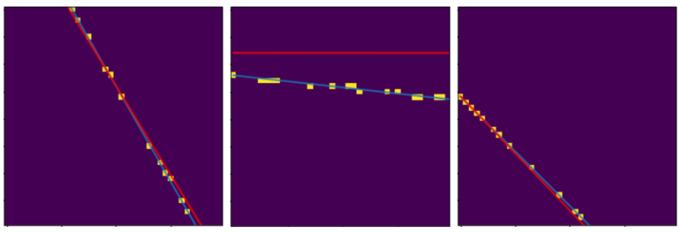

Figure 2: The Model Performance

The upper figure shows 3 random examples. Commonly, resultant lines are deviating but with less degrees. There are still outliers as second image shown. Overall, this model has a great improvement compared to the first one, reflected in their test errors and rebuilt lines.

4. Multi-channel Input

In this attempt, the channel number of each image is augmented from 1 to 3 before being pushed into the network architecture. Incredibly the final validation error turns out to be 1.62 with training error being 1.39, after it was fluctuating between approximately 1.4 to 3.0. As far as the test set is concerned, it has only 3.11 error, decreased by 1.80. Again, the overfitting is further mitigated.

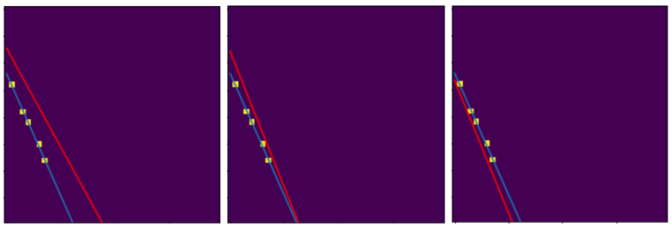

Figure 3: A Comparison among Three Models' Performance

These 3 images are 3 models' predictions for a same target. The distances on the normal direction are lessening, so do the angle distances. So, this model makes a prediction with higher precision than previous ones.

The rationale of this progress may be: when 3 channels' activation map is stacked together, (imagine each activation map as an altitude map) the peak of the activation (i.e. where the scatter points are) will be much higher so that they are distinct from other activation. This makes their position easier to be located by ReLU. For example, the activation of a scatter point will be a single pixel rather than a patch. This character allows these locations to propagate deeper in the network, thereby training deep parameters to be more precise.