MLP Coursework 2: Investigating the Implementation and Optimization of Convolutional Networks

S1793292

Abstract

1. Introduction

Convolutional Neural Networks (CNN) are a Deep Learning algorithms which can take in an input image, assign learnable weights and biases to various aspects of the image and be able to differentiate one from the other. The preprocessing required in a CNN is much lower as compared to other classification algorithms. Training such networks is a complex task and sometimes we encounter issues that either arise during the training or after the completion of it. On this report we explore the way CNNs are implemented and how we can optimize their performance of them. We also try to improve the generalization performance of them using different methods.

2. Implementing convolutional networks

When programming a convolutional neural network(CNN), each convolutional layer consist of a set of filters (2d matrices). When the input passes through the layer, the input has to pass through stages. 1) Overlaying the filter on top of the input at some random location. 2) Performing element-wise multiplication between the values in the filter and their corresponding values of the given input. 3) Sum up all the element-wise products.

3. Optimization problems in convolutional networks

By analyzing the output files of the two networks (VGG_08, VGG_38) along with the graphs provided, we can see that the VGG_38 doesn't improve the accuracy or minimize the loss which leads us to believe that the gradients are vanishing and won't converge. To tackle this we can try to introduce the concept of the ResNet(Residual Network)(He et al., 2015) which creates a shortcut between layers which adds the output from the previous layer to the layer ahead. ResNet enables us to use deep CNN and larger learning rates without the problem of the vanishing gradients (Bengio et al., 1994). To achieve this we need to normalize the input of the layers before the Leaky ReLu and in the end we add the input of the previous layer. We implemented this on the ConvolutionalProcessingBlockRes and the ConvolutionalDimensionalityReductionBlockRes class. Another way to minimize the vanighing gradients is to introduce Batch Normalization(BN) (Ioffe & Szegedy, 2015). The idea is to normalise the inputs of each layer in such a way that they have a mean output activation of zero and standard deviation of one. This is analogous to how the inputs to networks are standardised. BN enables us to train a network faster and also increase the learning rate. The BN must happen before the activation of the layer. We implemented this method on the ConvolutionalProcessingBlockNorm and ConvolutionalDimensionalityReductionBlockNorm class.

4. Improving the generalization performance of convolutional networks

Generalization is referred to the ability of an algorithm is this case, the CNN, to be able to work efficient in multiple data. If the CNN overfits a given dataset then it's not gonna perform the same on other datasets.To minimize this we can introduce a few methods that helps our model with analyzing multiple datasets with similar accuracy score. To do this, we proposed the data augmentation and weight decay. Data augmentation introduces an extra step while or model is training. It modifies the input image so that the model can have a wider range of data. The second method, weight decay,follows a different approach. When training neural networks, after each update, the weights of the network are multiplied by a factor slightly less than 1. this can prevent the weights from becoming too large and overfit our data. We chose those two methods as there the data augmentation is very helpful when dealing with images on a CNN as it randomly rotates the image, zooming in, adding a color filter among other to create a range of images that are slightly different from the original ones. The weight decay method was used because when the model is overfitting the data, a good way to minimize that is to limit the weights that are being used during the training.

5. Experiments

Seeing that the network has some signs of overfitting we decided to run an experiment to find out if by enabling weight-decay and monitor if there is any significant changes on the final model. We noticed that the accuracy of the model increased and loss minimized faster using weight-decay.

6. Discussion

Although weight-decay improved the overall performance but from the results we gathered we can tell that the difference is not that significant and we probably need to combine different methods to significantly improve the model.

7. Conclusions

Having implemented different methods to tackle the vanishing gradients problem and trying to prevent the network from overfitting the data we found out that is important to take into consideration optimization techniques in order to minimize similar issues. By applying some methods we also improved the generalization performance of the model so it can perform similar to other datasets.

References

Bengio, Yoshua, Simard, Patrice, Frasconi, Paolo, et al. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2): 157–166, 1994.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

Ioffe, Sergey and Szegedy, Christian. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning*, pp. 448–456, June 2015. URL http://proceedings.mlr.press/v37/ioffe15.html.