Neural Networks Midterm Project

Benjamin Klybor, Damian Creasy

October 23, 2018

1 Introduction

In this paper we set out to improve on the convolutional CIFAR-10 model provided by Keras.[1] The CIFAR-10 dataset contains 60000 32 by 32 color images for training data and 6000 32 by 32 color images for testing data. The neural network provided by Keras, of which the CNN (convolutional neural network) is the backbone, obtains an accuracy 75% over 25 epochs and 79% over 50 epochs. We set out to experiment with various adjustments in order to examine their effects on the accuracy of the CNN on the CIFAR-10 dataset. We find that with a few minor adjustments to the network, we can typically improve accuracy. Our experiments include adding an upsampling layer, changing the activations to exponential linear units (ELU), and combining those two with a convolutional 2D transpose. Most of these minor changes do not increase the computational complexity by a significant amount, with upsampling being the exception.

2 Related Work

There have been many breakthroughs in state-of-the-art performance for the CIFAR-10 classification problem, however, one sticks out as particularly beneficial. Clevert et. al. introduced a novel activation function, the exponential linear unit (ELU), in order to improve upon the largely popular rectified linear unit (ReLU) activation. The ELU uses the following piecewise function:

$$f(x) = \begin{cases} x & x > 0 \\ \alpha(exp(x) - 1) & x \leq 0 \end{cases}$$

where α is a hyperparamter which the ELU will push to for negative values. As they show, this simple change to the piecewise function greatly improves generalizability and speed without sacrificing computational simplicity. This is due to the function's ability to push the mean closer to zero through the inclusion of negative values, which in turn allows the gradient to take on values which decrease the time to learn.

[1] `https://github.com/keras-team/keras/blob/master/examples/cifar10_cnn.py`

3 Approach

Our first improvement to the Keras example code comes from finding the best optimizer for the neural network. We considered testing the rms, sgd, adagrad, adadelta, and adam optimizers for optimization. Several trials of the Keras example code using different optimizers lead us to use the adam optimizer for best accuracy (Fig. 1) , although it should be noted that the rms optimizer produced the quickest results. After finding the optimal optimizer, we set out to determine the best activation function to use for each layer of the network. Starting with the rectified linear units that Keras uses for each layer, we optimized for the best activation function. Optimizing for the activation function, again on the vanilla Keras example code, lead us to use the adam activation function for each layer. After finding the best activation function, we added in new layers and changed the ordering of some of the layers in an effort to improve performance. The greatest increase in accuracy came from inserting a 2D upsampling layer in the middle of the network.

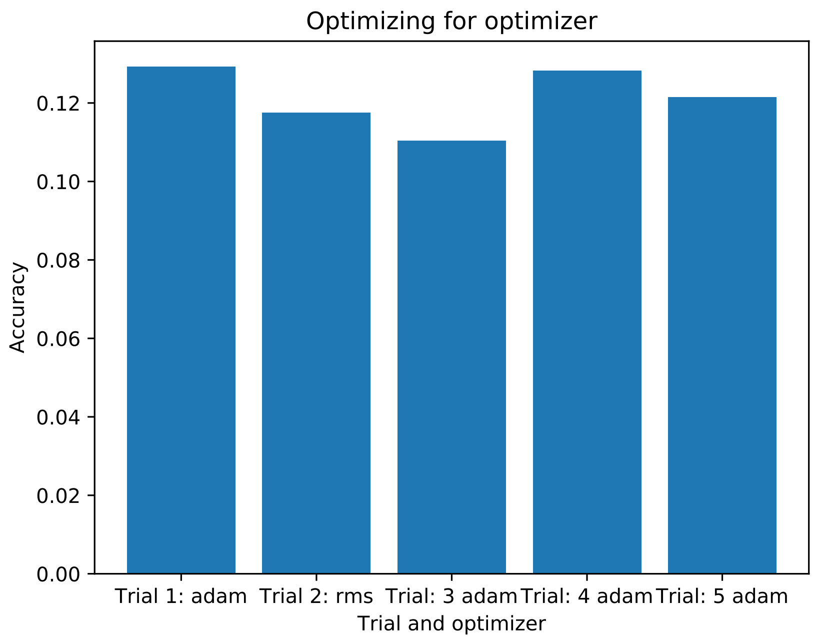

Figure 1: The optimizing process for the ideal optimizer. We used the Keras CIFAR-10 example code using CIFAR-100 data for this process, however we found the ideal optimizer to be `adam` for the CIFAR-10 dataset as well. Each trial lasted for one epoch with the same learning rate and batch size.

The final network architecture was two 2D CNN's followed by a pooling layer; the upsampling layer; two more 2D CNN's followed by a pooling layer; and finally a flattening layer followed by two dense layers. All other layers besides the final dense layer used an exponential linear (ELU) unit for the activation function. Other types of layers supported by Keras were tested, but produced sub-optimal results. Rearranging the layers did not seem to have much effect on improving the accuracy of the network either,so the final network keeps the same ordering as the Keras example code, aside from the addition of the upsampling layer. Our implementation of this network can be found at `https://github.com/DaCreasy/DLMidterm/midterm3.py`.

4 Results

All experiments we run are compared to our baseline, represented by the CNN architecture provided by Keras. This CNN achieves an accuracy of 75% in 25 epochs. The average time per epoch is 10 seconds, establishing our baseline. The first experiment we run is the upsampling network. This network achieves an accuracy of 77% in 25 epochs, with an average time of 40 seconds. The next experiment is the ELU activation network, which achieves an accuracy of 76% with an average time of 10 seconds. It is worth noting that at 25 epochs, this network still improves validation accuracy. As such, we continue to run the network for a total of 100 epochs, at which point it achieves an accuracy of 81%. Despite this, we wish to control for number of epochs, so we only consider the accuracy of the network after 25 epochs. Finally, we run the network that includes upsampling, ELU activations, and convolutional 2D-transpositions. This networks achieves an accuracy of 74% with an average time of 50 seconds per epoch.

5 Analysis

Our results show that the additions we have made to the Keras example code have had a measureable, though not great, effect on the accuracy of the neural network. Replacing the ReLU activations with ELU activations was a surprisingly simple modification that gave us a decent increase in performance. The addition of the upsampling layer was another simple improvement to the code that ended up giving us the best increase in performance from the 75% baseline. Replacing the 2D convolutional layers with 2D convolutional transposition layers should have, at the very least, not negatively impacted accuracy. So it was unexpected to find that the switch in fact decreased accuracy. Regardless of the layer architecture we tested, however, for some reason using the `adam` optimizer for each layer always gave the better accuracy.

6 Conclusion

Our understanding of the convolutional neural network was at the writing of this paper amateurish. Given that, we felt the best use of our time would be optimizing the Keras exmaple code for parameters such as the optimizer. In addition, while we know that adding the upsampling layer increases accuracy, we do not at this time understand why. The same is true of why the substitution of the 2D convolutional layers for 2D convulutional transpose layers negatively affected accuracy so much. While our experimentation has not produces marked improvement in accuracy over the Keras example neural network, we believe we are on the right track to understanding how to implement a convolutional neural network.

7 References

[1] Clevert, D., Unterthiner, T. & Hochreite S. (2016)
Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUS). URL https://arxiv.org/pdf/1511.07289.pdf