Convolutional Neural Network Report

Ruby Link

Machine Learning

November 30th, 2023

Introduction

Convolutional neural networks are a common architecture for image classification. They are superior to a simple feed forward neural network because their data doesn't need to be compressed. Patterns still exist when in a compressed form but it is much more difficult to decipher. CNNs maintain spatial structure and each pixel becomes a feature in the model. In order to improve CNNs, preprocessing is required in order to ensure that the model can successfully interpret the data. Additionally, there are regularization methods that can further enhance the performance and hyperparameters can be tuned to train a model. In the dataset, CIFAR-10, there are 10 potential image classifications. We trained models using various regularization methods and tuned hyperparameters in a CNN in order to successfully classify the images in the dataset.

Batch Normalization

The first regularization/normalization technique that was implemented in the convolutional neural network is batch normalization. Earlier in the model, we normalized the data by centering it so the distribution of the data follows a normal distribution. Batch normalization is similar in that it is used to lessen the impact of variance between mini-batches. Batches are chosen at random, but this can lead to an Internal Covariate Shift in the model when the distribution of input data varies between mini-batches. Maintaining consistency in the data that reaches the kernels helps improve the overall training of the model and, therefore, the results. However, as seen in the code, there was nearly a 10% decrease in the accuracy (~55%) when compared to the base model, which had an accuracy of 64.4%. A reason for this could be that the batches are too small and may not be representative enough of the entire dataset. In the future, we might want to play around with the batch size and see if that improves the

performance. We also may need to tune our hyperparameters to support the use of batch normalization. The model could also simply not benefit from the use of batch normalization and the addition is not necessary to achieve better performance.

L2 Regularization

The second regularization technique implemented in the convolutional neural network was L2 regularization. L2 regularization is when weights are penalized and pulled close to 0. This enforces the idea of making the larger weights have less of an impact on the output and making sure that each of the weights are used and some weights aren't held to a higher importance in the model. This is useful because it prevents overfitting of the model. In the code, the results drop by about 5% compared to the base model (64.4% initial, 58.9% with L2). A potential reason for the decline in accuracy could be that this model doesn't benefit from the penalization of weights due to complexity reduction. Although L2 (ridge regression) does drag the weights to zero unlike L1, it may be penalizing the weights too heavily and reducing too much of the complexity in the model that is needed to accurately classify the images.

Dropout

Dropout is a technique that is applied per layer in a neural network, excluding the classifier. For each forward pass, a random percent of nodes in a layer has their activation set to zero, canceling any output that they have. In a feed forward network, this is used to promote sparsity and ensure that the model isn't reliant on certain paths or nodes. In a convolutional neural network, this regularization method still shows to improve the model even though in theory it is difficult to explain why. When used on a CNN, it isn't promoting spasticity in the same way that it does with a FFNN, but we are still able to see the accuracy increase to over 70%

for the validation data, an improvement from the first model. This shows that it is improving the model and making it more generalizable to data that it hasn't seen before.

Kernel Size + Dropout

In all of the previous models, we were using a 3x3 kernel. To see how the model would react if the kernel size were different, we updated the size in another model to be a 4x4. The performance of the model slightly decreased in both the training and the validation data. It decreased from 72.78% in the model solely utilizing dropout to 68.5% with the increased filter size. This is likely due to the fact that with a larger kernel size, the filter is being spread over a larger area of the image at any given time. Larger kernel sizes will have a harder time picking up on smaller features and detecting specific/fine patterns in the images. The model may then have a harder time identifying those patterns in images that it has not been trained on.

Learning Rate + Dropout

The learning rate is the step size when performing gradient descent and a hyperparameter in this coding example when using the adam optimizer. It correlates the step size when descending down the gradient. A small learning rate minimizes noise, but there is a risk of converging at local minima in the loss function. With a large learning rate, there is increased potential of skipping over the minima of the loss function and noisy convergence. In previous models, the learning rate was set to 0.0001 as shown in the online example. Changing the learning rate to 0.01 yielded less accurate results, lowering it to 68.7%. There may be more noise with the larger learning rate and although the convergence may have been quicker, the model appears to do better with a slower learning rate to avoid landing at a suboptimal solution.

Conclusion

Each of the adjustments made to the original model all had an impact one way or another on the performance of the convolutional neural network's ability to accurately classify images. While many of these changes – including batch normalization, dropout, L2 regularization, and hyperparameter tuning – had the potential to improve results by decreasing convergence time, limiting complexity, and helping to reduce overfitting, many hindered the performance despite potential benefits. It is important to experiment and test various regularization methods and hyperparameters as each dataset responds differently to the model's architecture and parameters.