Project Part-3 Report

rsanikom@asu.edu

Introduction: We are provided CIFAR-10 data, and we must utilize convolutional neural networks to categorize the data set. We frequently employ convolutional neural networks when working with visual data. The CIFAR -10 Data contains input images with a 32x32x3-sized image. They must be divided into ten classes.

Data Set: CIFR-10 Dataset

Libraries used: Keras, TensorFlow.

Problem Statement: To classify Image Data into different classes.

Initial Case: These are the test loss and test accuracy I got when I executed the baseline provided.

Test loss: 0.6167688369750977
Test accuracy: 0.8306000232696533

Method: We utilize a 12-hidden-layer convolutional neural network with kernel-sized layers, ReLu activation in some levels, Maxpool layer, drop-out in others, and batch normalization in others. The final layer categorizes 10 classes with 10 output nodes using the Softmax activation function. The question includes the neural network's setup.

Results and Observations:

By altering the network's hyperparameters and parameters, we examine how the network performs under various configurations while attempting to determine why it behaves that way.

Case A:

i) Learning Rate: 0.05: The neural network converges swiftly and reaches the least cost value soon if we raise the learning rate. The model may, however, miss the minimum and go over it if the learning rate is fast. When compared to the default situation, the test error and accuracy in this instance rose, suggesting that the model may have missed the minimum and continued anyhow.

Test loss: 1.5697132349014282
Test accuracy: 0.748199999332428

ii) Learning Rate: 0.0001: If we slow down the learning rate, the model takes too long to converge, converges slowly, and makes baby steps. In this scenario, compared to the default case, accuracy is decreased and test error is raised; hence, the model may not converge with the specified epochs and may need a few additional iterations and epochs to converge to a minimum.

Test loss: 0.516810417175293

Test accuracy: 0.8360000252723694

Case B:

Convolutional neural networks function locally; however, if the kernel size is big, the locality property will be lost when the convolutional neural network begins to behave normally and fails to capture the properties of the input image, leading to underfitting. However, because we applied the same padding to all of the levels, we essentially achieved the default outcome.

Test loss: 0.6492162942886353

Test accuracy: 0.8539999723434448

Case C:

RMS Prop is inferior than Adam Optimizer. While we change the parameters in the Rms prop using the first moment's mean, we also update the parameters in the Adam optimizer. This improves convergence speed and accuracy while requiring little parameter tweaking. As a result, using RMS prop gave us less accuracy.

Test loss: 0.5491887927055359

Test accuracy: 0.8483999967575073

Case D:

Data originating from a layer of the neural network is normalized using batch normalization. In normalization, the activation scales the input layer. Regularization improves learning efficiency and protects against overfitting. Therefore, when we remove batch normalization, accuracy suffers greatly and error increases.

Test loss: 2.303337812423706

Test accuracy: 0.10000000149011612

Case E: We randomly deactivate some of the neurons in the layers of the neural network as another regularization strategy to avoid overfitting. The general rule of thumb is to utilize drop out up to 50%; if we use drop out more than 50%, we may run into a situation where the accuracy is diminished and the error is raised, as in this instance.

Test loss: 0.7065795660018921

Test accuracy: 0.7649999856948853

Case F:

(i) **Batch size-16:** The weights update more quickly if the batch size is smaller, but because you can only train a small number of samples at once, this is not ideal. In some circumstances, this might only converge to the local minimum rather than the global minimum. We employ batch size to speed up the network, but smaller batches could not capture all of the image's patterns. As a result, as compared to the default scenario, test accuracy is lowered and loss is raised.

Test loss: 0.5705927014350891

Test accuracy: 0.8411999940872192

(ii) **Batch size-256:** In generally, accuracy should rise as batch size increases, but because we are maintaining the same learning parameter, the model will overfit and have somewhat lower accuracy than the default model.

Test loss: 0.5818068385124207

Test accuracy: 0.8432999849319458

Conclusion: From the various configurations tested here, we conclude that a decent learning rate that is neither too high nor too low is required to get the model to converge. Additionally, we shouldn't enlarge the kernel to the point where the locality of the pixels is not taken into account. Adam Optimizer is a better optimizer in this situation. Batch normalization is essential to prevent overfitting, and drop out has to be used appropriately to prevent underfitting the model. To achieve converge with few epochs and acceptable performance, an adequate batch size and learning rate should be maintained. With neural networks, you may experiment with several configurations to find the ideal one. It is crucial to understand how to fine-tune the parameters and hyperparameters to fast achieve the optimum outcome.