Fundamentals of Statistical Learning and Pattern Recognition

CSE 569

Project Report on Deep Learning with CNN

Submitted To

Prof. Dr. Baoxin Li

Submitted By

Sai Vikhyath Kudhroli

ASU Id: 1225432689

Date: 08 December 2022

**INTRODUCTION**

**Problem Statement:** To perform hyperparameter tuning in a Convolutional Neural Network used to classify CIFAR-10 dataset. Convolutional Neural Network is used to classify the CIFAR-10 dataset into appropriate classes.

**Dataset Description:** The dataset used is the CIFAR-10 dataset which is included in the Keras framework’s datasets. It consists of 60000 colored images, each of the size 32 x 32 which implies the size of the dataset when read into NumPy array would be of the shape 32 x 32 x 3, where 3 represents the RGB channel of each image. The dataset consists of 10 classes. Out of the 60000 images in the dataset, 50000 are training images and the rest of the 10000 are testing images.

**PARAMETER DESCRIPTION**

**Learning Rate:** Learning Rate is a parameter that can be tuned in an optimization algorithm that determines the amount by which the weights of the neural network must be varied with respect to the minimum loss function.

If the learning rate of the algorithm is extremely low, the algorithm takes an extremely long time to converge to the global optimum. Whereas if the learning rate is too big, then the algorithm may overshoot the global optimum and may converge at some other point other than global optimum. So, a learning rate must be carefully defined for the algorithm to converge at global optimum within reasonable time.

**Kernel Size:** Kernel size is a two-dimensional matrix with each cell containing some value, which is used to perform convolution operation with the image. Kernel size specifies the window size for the convolution operation.

**Optimizer:** An optimizer is an algorithm that modifies the attributes of the neural network. Typically, an optimizer reduces the overall loss resulting in enhancing the accuracy of the neural network. Optimizer modifies the parameters such as weights to improve the accuracy of the model. Some of the prominent optimizers used are Adam, RMSprop (Root Mean Square propagation), SGD (Stochastic Gradient Descent).

**Batch Normalization:** Batch Normalization is a technique used to expedite the training process of the network by normalizing the layers’ inputs by using some normalization technique. Batch Normalization in general, re-centers and re-scales the data which ensures faster computation and faster training. Some of the popular normalization techniques are Min-Max Normalization, Log Scaling, Decimal Scaling, Feature Clipping, Z-Score.

**Dropout:** Dropout is a regularization technique that nullifies the contribution of some nodes that have least contribution to learning process. A percent of the nodes that contributes least to the training are cut off from the network by removing the forward and backward edges that pass through those nodes. Dropout is used to avoid overfitting of the model. When the model over fits, it performs well on training data and performs poorly on test data and when some of the least contributing nodes are removed from the network, it generates a more regularized function that fits well to the test data.

**Batch Size:** Batch size defines the number of training samples that will be propagated through the network. Once the samples of one batch are propagated through the network, the model is updated, hence controlling the accuracy of the error of the gradient when training the neural network.

**RESULTS AND OBSERVATIONS**

**Results with out any modifications:** Without any modifications to the network, the model performs well both during training and on test data. During training, it can be observed that the training accuracy, training loss, validation accuracy and validation loss are improving with each epoch. The following are the test accuracy and test loss obtained:

Test Accuracy: 85.49%

Test Loss: 0.5943

Text

Description automatically generated

**Results with 0.05 learning rate:** The model performs decently with 0.05 as the learning rate but the accuracy reported is less than that of with 0.01 learning rate. When the learning rate is increased to 0.05, the algorithm takes bigger steps to update weights to reach global optimum, but with high learning rate there is always a possibility that the algorithm may overshoot the global optima which could have happened in this case. The following are results reported:

Test Accuracy: 83.29%

Test Loss: 1.5227

Text

Description automatically generated

**Results with 0.0001 learning rate:** The model performs better when the learning rate is set to 0.0001 than when set to 0.01 and 0.05. When the learning rate is set to a lower value, the algorithm takes smaller steps in updating weights to reach the global optima. Thus, reducing the learning rate to 0.0001 may have reduced the step size of reaching global optima and may have appropriately reached global optima. However, setting the learning rate to a low value may not always be a good approach because the algorithm would take a lot of time to converge to global optimum. The following are the results obtained:

Test Accuracy: 86.74%

Test Loss: 0.6290

Text

Description automatically generated

**Results with 7x7 kernel size:** Model’s accuracy dropped when a kernel of 7x7 size was used. Increasing the kernel size increases the window of the convolution operation. Increasing the kernel size can increase the complexity of the model, however, with a large kernel size, more area of the image is convolved at once leading to important and unimportant features of the image to be convolved at the same time which can dampen the efficiency of the model. With a lower kernel size, small area of the image is convolved leading to important features to be convolved to a single cell in next layer. The following results were obtained:

Test Accuracy: 80.54%

Test Loss: 0.6462

Text

Description automatically generated

**Results with RMSprop optimizer:** The results obtained with Root Mean Square propagation optimizer are close to the results obtained with Adam’s optimizer with a minimal loss of accuracy. Both Adam and RMSprop are adaptive optimizers. However, RMSprop uses a decaying average of partial gradients to update the parameters, which simply is a way to update weights by considering some history of the gradient descent. While Adam adapts the parameters based on the mean and the variance of the past few gradients. So, Adam in most cases performs better than other optimizers. The following results were obtained:

Test Accuracy: 84.62%

Test Loss: 0.5240

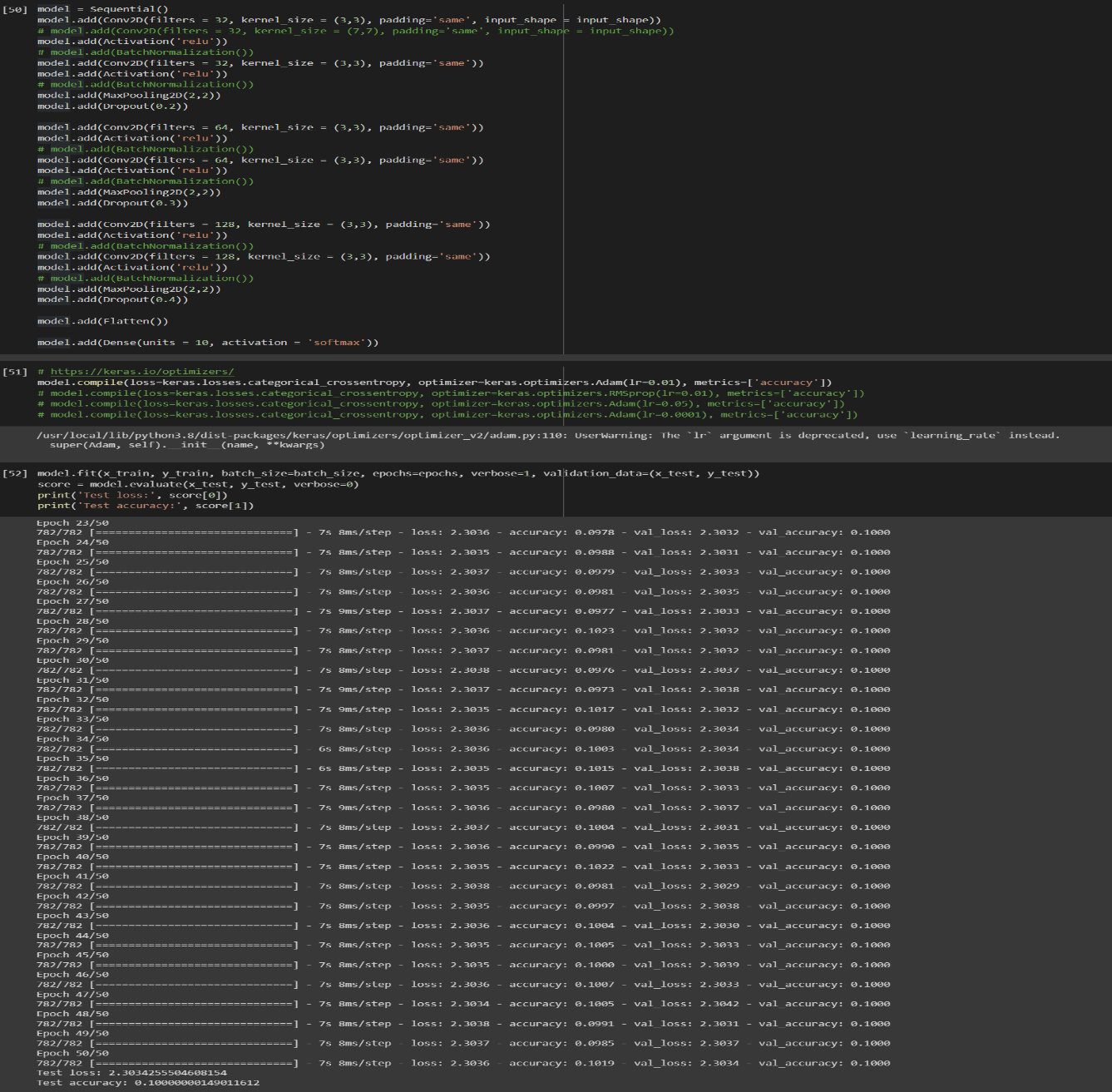
Text

Description automatically generated

**Results with batch normalization removed:** The results obtained after removing batch normalization are extremely poor. Removing batch normalization will not normalize the inputs at each layer resulting in the inputs being of very large values and would require an extremely low learning rate to converge to global optimum. Because the inputs are not normalized and the learning rate not modified, the model performs poorly. The following results are obtained:

Test Accuracy: 10%

Test Loss: 2.3034



**Results with 0.7 dropout:** The results obtained with 0.7 dropout are not as good as the one without any modifications. Dropout is used for regularization. When the model performs well on training data and does not perform well on test data, dropout is used to regularize by dropping out least contributing nodes from the network. But the percent of nodes dropped has an impact on the performance of the model. When more nodes are dropped, the network becomes sparse and may perform poorly.

Test Accuracy: 74.48%

Test Loss: 0.7895

Text

Description automatically generated

**Results with 16 batch size:** Batch size has an impact on the accuracy of the model by controlling the estimate of error in gradient descent. Lower batch size implies the weights will be updated several times in each epoch. A smaller batch size with a low learning rate works better on most of the scenarios. The results obtained with the batch size of 16 were as follows:

Test Accuracy: 82.62%

Test Loss: 0.6175

Text

Description automatically generated

**Results with 256 batch size:** Batch size has an impact on the accuracy of the model by controlling the estimate of error in gradient descent. Higher batch size implies the weights will be updated lesser number of times in each epoch. A larger batch size with a high learning rate generally works together. The results obtained with the batch size of 256 were as follows:

Test Accuracy: 83.25%

Test Loss: 0.5727

Text

Description automatically generated

**CONCLUSION**

This project was aimed at learning about Convolutional Neural Network and Hyperparameter tuning of the neural network to enhance accuracy. Key learning from the project was on how to build a convolutional neural network to perform image classification using Keras framework. It was observed on how the model’s accuracy varies with various parameters. It was also learnt what parameters to be used in a particular use case and how to tune those parameters to achieve better results.