A CNN model on the CIFAR 10 Dataset: M3 Final Report

Justin Harris
Computer Science
Python for Machine Learning
Conputer Science
CS 477 - Spring 2021

CNN with CIFAR 10

Justin Harris
Computer Science
University of Idaho
Moscow, Idaho
harr1433@vandals.uidaho.edu

***Abstract*—This electronic document is a "live" template and already defines the components of your paper [title, text, heads, etc.] in its style sheet. **CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.* (*Abstract*)**

Keywords—component, formatting, style, styling, insert (*key words****)***

I. Introduction

The CIFAR 10 dataset contains 60,000 images that are 32 x 32 pixels in three channels, separated into 10 classes. The dataset is split by default into 50,000 images for training, and 10,000 images for testing. The goal of this project is to train a Convolutional Neural Network (CNN) model using data from the training set, and then recognize and classify a new image to one of the 10 classes.

The CIFAR 10 dataset used for this project can be found at the following link: https://www.cs.toronto.edu/~kriz/cifar.html.

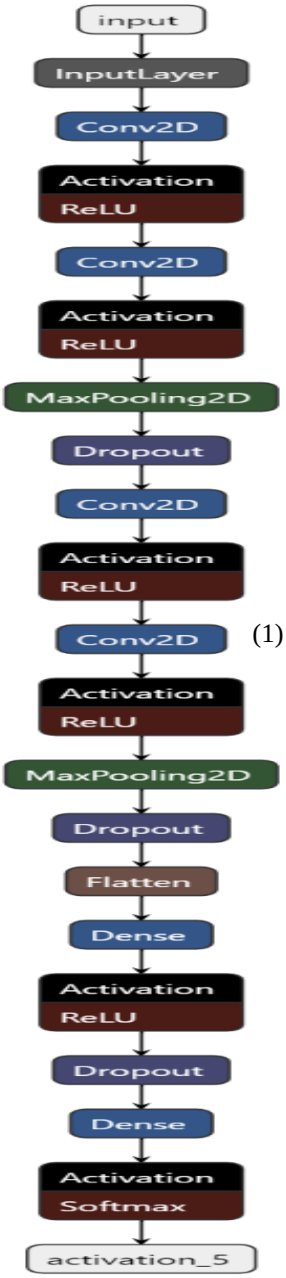
II. METHOD

A. *Model Architecture*

The approach used for this model is to use a CNN model with a sequential convent architecture. To start, I normalized the data and convert all image intensities to range from {0 – 1], instead of [0 – 255]. The data is already split between training data and testing data in the CIFAR 10 dataset, to this was left in its default state.

To build the CNN architecture, I am using the Sequential model type in Keras to build the model layer by layer. The model is currently set up with a total of 4 convnet layers using Max Pooling on two of the layers, one layer to flatten the Feature Maps and one Dense layer and one output layer. Figure 1 shows a visual representation of this architecture.

After looking though multiple models on the CIFAR 10 dataset, this type of architecture appeared to be a simple implementation while providing accurate results for the problem.



(1)

B. *Model Parameters*

Each convnet layer learns a specified number of convolutional filters of 3 x 3 size and uses ReLU as the activation. The first and second of these layers, the model will learn 32 convolutional filters followed by a Max Pooling of size 2 x 2 and an additional Dropout of 25% to attempt to limit an over-fitting problem with the model.

The third and fourth layers follow a similar pattern to the first and second layers, but the model will learn 64 convolutional filters in each of these layers. These layers are followed by the same Max Pooling and Dropout of the previous two layers.

The next layer flattens the Feature Maps and adds a Dense layer of 512 nodes with a Dropout of 50% for retraining the output of each hidden layer. The last layer is the Output layer with the 10 classes as its output and "softmax" activation.

Two trainings were run on this model, each with a different optimizer. The first run used the RMSprop optimizer and the second used the Adam optimizer.

Additionally, the model uses a default batch size of 32, an epoch size of 100 and a learning rate of 0.0001 on both optimizers.

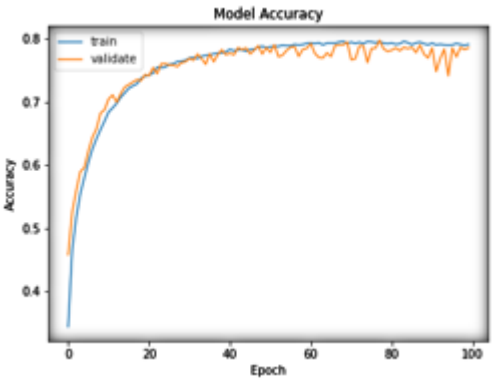
III. EXPERIMENTAL RESULTS

As mentioned above, this model was trained and tested multiple times using different optimizers. Each time the model was trained and tested; the same parameters were used.

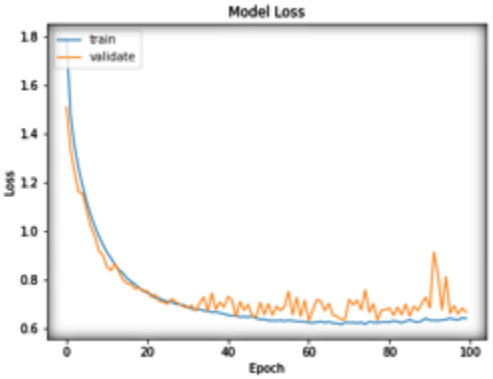
A. *Training Results*

Training the model with the RMSprop optimizer resulted in yielding the best results after running approximately 60 epochs. Running additional epochs past this point, the model does not improve in accuracy, as shown in Figure 2.

The model loss on the RMSprop optimizer decreases as the model accuracy increases and is optimal at approximately 60 epochs, shown in Figure 3. Since both the model accuracy and model loss remain close to the training line in the curves, it does not appear that this model running with the RMSprop optimizer has an over-fitting problem.

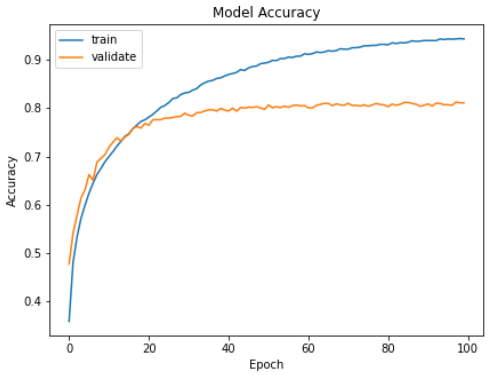


(2)

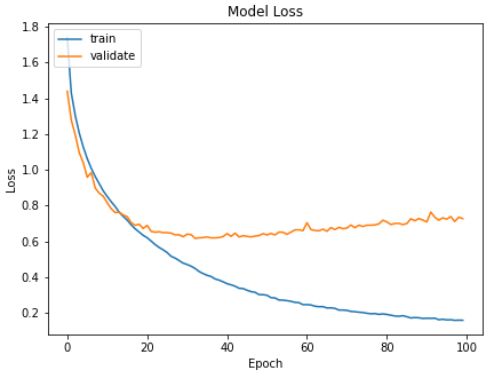


(3)

Comparatively, when training the model using the Adam optimizer, the accuracy curve shows that the model yields the best results at approximately 20 epochs, shown in Figure 4. Additionally, the model loss on the when training the model on the Adam optimizer follows the same patter as the model accuracy and is optimal at approximately 20 epochs, show in Figure 5.



(4)



(5)

Looking at the curves for both the model accuracy and the model loss when the model is trained with the current parameters, there is an over-fitting problem with the model after running approximately 20 epochs.

Identify applicable funding agency here. If none, delete this text box.

B. *Testing Results*

Testing was conducted by randomly selecting an image from the CIFAR 10 testing dataset. When the model was trained on the RMSprop optimizer, the best accuracy achieved was approximately 78.6%, shown in Figure 6. Testing the model when it was trained on the Adam optimizer resulted in a higher accuracy than the RMSprop optimizer, approximately 81.1%, shown in Figure 7.

```
313/313 [==============================] - 2s 6ms/step - loss: 0.6645 - accuracy: 0.7857
Test Loss:  0.6644977331161499
Test Acc:  0.7857000231742859
```

(6)

```
313/313 [==============================] - 2s 6ms/step - loss: 0.7264 - accuracy: 0.8110
Test Loss:  0.7264375686645508
Test Acc:  0.8109999895095825
```

(7)

IV. CONCLUSION

Overall, both optimizers on the model achieve a good base accuracy and can be further improved upon. Considering the two training of the model on each optimizer, the Adam optimizer at around 20 epochs of training would achieve more accurate classification of the images in the CIFAR 10 dataset 2.5% more often than the RMSprop optimizer running around 60 epochs of training. Possibilities for improvement may include adjusting the convnet layers or adding data augmentation to the model.

REFERENCES

[1] "CIFAR-10 and CIFAR-100 datasets", Cs.toronto.edu, 2021. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html. [Accessed: 10- May- 2021].

[2] "CIFAR-10 - Object Recognition in Images | Kaggle", Kaggle.com, 2021. [Online]. Available: https://www.kaggle.com/c/cifar-10/code. [Accessed: 10- May- 2021].

[3] J. Brownlee, "A Gentle Introduction to Dropout for Regularizing Deep Neural Networks", Machine Learning Mastery, 2018. [Online]. Available: https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/#:~:text=regularizers%20are%20present.-,%E2%80%94%20Dropout%3A%20A%20Simple%20Way%20to%20Prevent%20Neural%20Networks%20from%20Overfitting,be%20required%20when%20using%20dropout. [Accessed: 10- May- 2021].

[4] A. Brahmane, "Deep Learning with CIFAR-10 Image Classification", Medium, 2020. [Online]. Available: https://towardsdatascience.com/deep-learning-with-cifar-10-image-classification-64ab92110d79. [Accessed: 10- May- 2021].

[5] E. Allibhai, "Building a Convolutional Neural Network (CNN) in Keras", Medium, 2018. [Online]. Available: https://towardsdatascience.com/building-a-convolutional-neural-network-cnn-in-keras-329fbbadc5f5. [Accessed: 10- May- 2021].

[6] "🚗 🐎✈⛵Cifar-10 Object Recognition(CNN) Explained", Kaggle.com, 2021. [Online]. Available: https://www.kaggle.com/datajameson/cifar-10-object-recognition-cnn-explained. [Accessed: 10- May- 2021].