CNN Classification on CIFAR-10

Introduction

Problem to Be Solved

The goal of this program is to accurately classify natural images using a multi-layered model trained on a specific dataset, CIFAR-10.

Relevance of Problem

Being able to classify natural images could lead to benefits in the real world. For example, sightings of endagered or near-extinct animals could be detected through the classifier when processing large amounts of images online, leading to possible better protection of animals.

Plan to Solve Problem

To solve the problem, a pipeline will be developed that is capable of preprocessing, augmentation, training, prediction, and validation, and uses a CNN-based DNN for classification. This pipeline will be a deep learning solution to classifying natural images and will be able to uniquely classify into 10 classes.

Justification for a DL Solution

A random forest classifier was tested on the CIFAR-10 dataset, and the accuracy scores for validation and testing were 0.4555 and 0.4685 respectively. These scores were very poor in classifying the images from the CIFAR-10 dataset, no matter which hyperparameters were changed. This lead to the conclusion that the random forest classifier was not well-versed enough to solve this problem, and that a deep learning solution was required.

Data

Describing the Data

The CIFAR-10 dataset is a collection of 60,000 color images labeled into 10 classes. There is an even split among the classes, so there are 6,000 images per class. The dataset has 50,000 training images and 10,000 testing images. The classes mentioned above are the following: airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks.

Below is the result from exploratory data analysis, where the first 100 images from the training dataset and their labels are displayed.

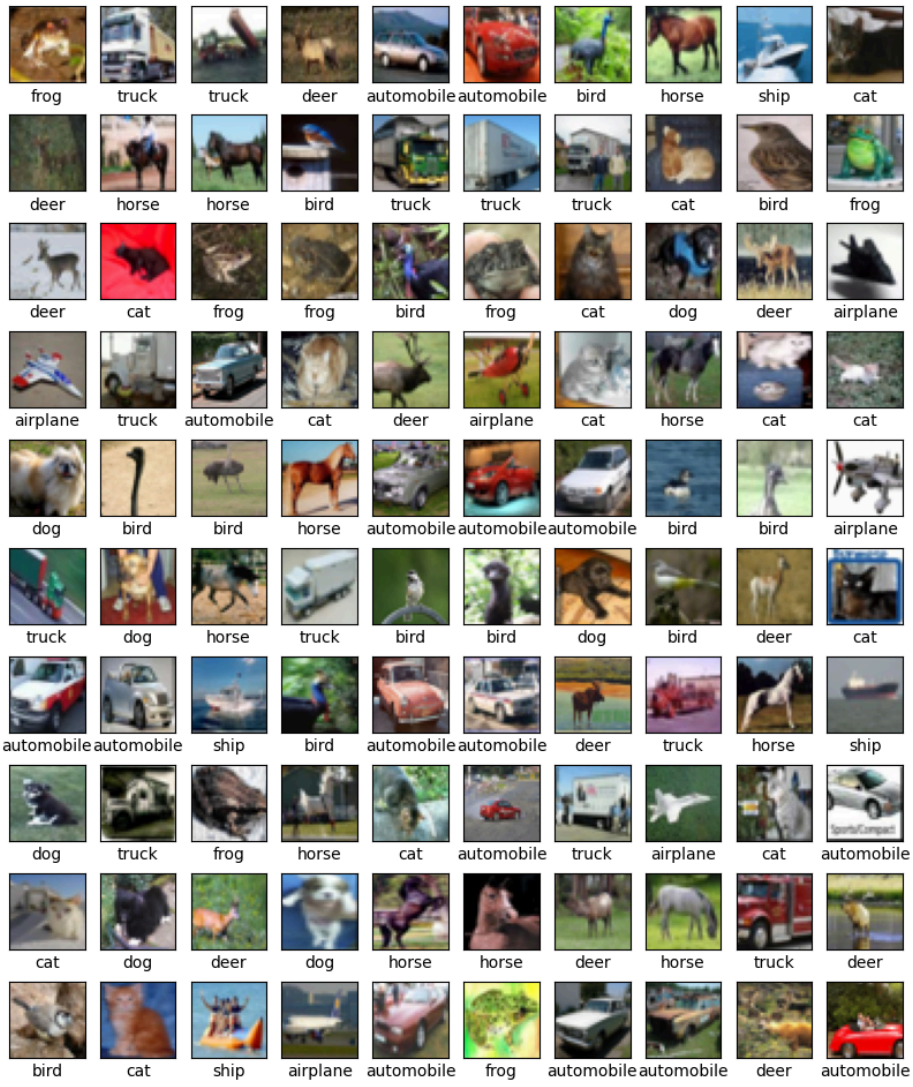

Figure 1: 'Figure 1'

Methods

Preprocessing and Augmentation

Several steps were taken to preprocess and augment the data. First, the data was loaded into training and testing variables. After loading in the data, the train and test variables for x were divided by 255 to normalize the pixel values between 0 and 1. This essentially greyscales the images. Next, the target labels are one-hot encoded. This was a new technique I learned which essentially creates a way to represent categorical variables as numerical values in a machine learning model; a perfect technique for the CIFAR-10 dataset. For augmentation, numerous changes were applied to the training set to alter the images such as: rotation, horizontal shift, vertical shift, shear angle change, zoom change, and flipping. The augmentations improved the model by allowing for new possible images it sees to be classified even if they are altered.

Model

The model used was a Convolutional Neural Network configured for image classification, where in this case, was applied to the CIFAR-10 dataset. The model has two sets of convolutional layers, each followed by batch normalization for improved training stability. The initial set involves a Conv2D layer with 64 filters and a (3, 3) kernel, coupled with a ReLU activation function. The subsequent set replicates the first but introduces a MaxPooling2D layer for spatial downsampling. Both sets integrate dropout layers (rate of 0.25) to mitigate overfitting. A Flatten layer precedes two fully connected layers, the first with 256 units, ReLU activation, batch normalization, and 0.5 dropout, and the second with 10 units and a softmax activation for class probabilities. The model is compiled using the Adam optimizer, categorical crossentropy loss, and accuracy as the evaluation metric. Training spans 20 epochs with a batch size of 16. A lower batch size than the default was chosen to free up memory when training the model, and did not seem to have a large impact on accuracy when compared to a batch size of 32.

Results and Discussion

Initially the accuracy of the model was around 0.65 through 10 epochs. After altering the hyperparameters and epoch count, the accuracy dropped drastically to a score 0.09. One-hot encoding was implemented to improve the score and it managed to improve it significantly. After including this improvement, the CNN model achieved an accuracy score of 0.81 through 20 epochs and a batch size of 16. This score is fairly good, but could have definitely been better. To improve it, more hyperparameter tuning and altering the layers in the model was required.

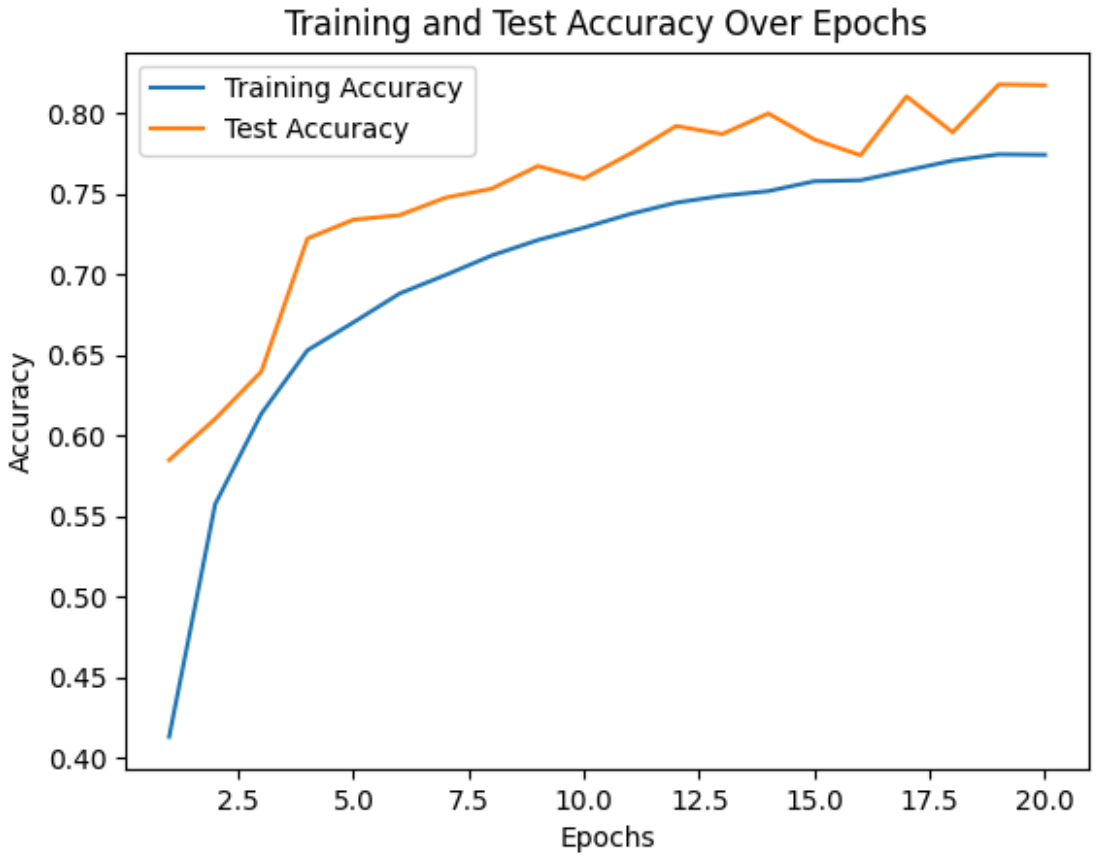

Figure 2: 'Figure 2'

The model aboves depicts the training and test accuracy curves over each epoch. As expected, the test accuracy starts and remains higher than the training accuracy over each epoch, but towards the end of the epochs, the training accuracy reaches very close to the test accuracy.

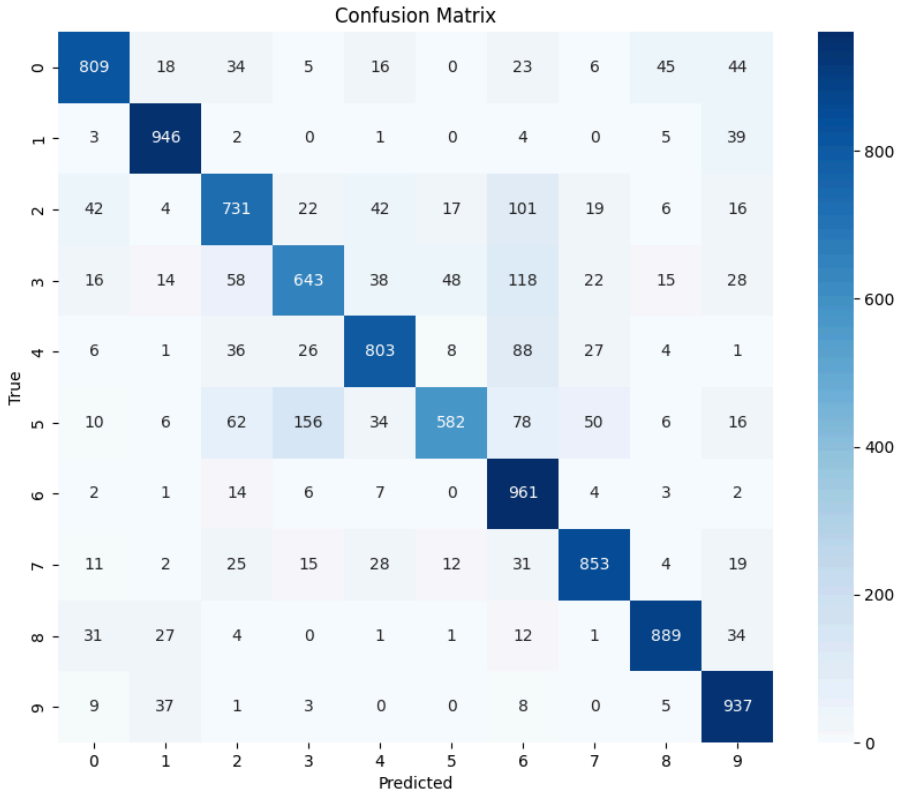

Figure 3: 'Figure 3'

The confusion matrix above implies that the model was effective in predicting images correctly. The higher values along the main diagonal imply good performance. However, there is a lack of consistency in the matrix, indicating that the model is not completely accurate.

Model Effectiveness

With an accuracy score of 0.81 and a confusion matrix which partially supports the model's ability to accurately predict images, it can be determined that the model is somewhat effective. The accuracy and effectiveness of the model has room to improve, however it still provides effective classification of images to a significant degree.

Future Works and Lessons Learned

To improve the model in the future, I would focus on adjusting the layers rather than the hyperparameters first, as this was the biggest impact on the models accuracy. As this is the first time building a model, I struggled a lot on it but learned a lot along the way. I had to reference a lot of resources and a couple of my classmates along the way as this was all foreign to me. Learning it was a tricky process but was extremely beneficial.