EECE 5644 Project
CIFAR-10 - Object Recognition in Images

Team Members: Yuxuan Cai, Jiahui Zhang, Heng Zhou

Programming Language: Python

1 Question Description

On Kaggle, CIFAR-10 dataset is used for an object recognition competition. The competition aims to train machine learning model and develop algorithms for classifying the images and predicting the label for each image in the test set, as accurate as possible. So our team want to participate in the Kaggle challenge and see how much accuracy we can get.

Hopefully our model will generate the labels that match the official labels. We will evaluate and visualize our object recognition results.

2 Source of Data

CIFAR-10 data was collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The dataset can be found on Kaggle.com and cs.toronto.edu (Alex Krizhevsky's personal website)

In this project, we just use the dataset from Kaggle.com and the link is shown below:
`https://www.kaggle.com/c/cifar-10/data`

3 Data description

CIFAR-10 (Canadian Institute For Advanced Research) is an established computer-vision dataset used for object recognition. The CIFAR-10 data consists of 60,000 32x32 color images in 10 classes, with 6000 images per class. There are 50,000 training images and 10,000 test images in the official data. Kaggle have preserved the train/test split from the original dataset.

Kaggle provide train and test files:

- train.7z - a folder containing the training images in png format
- test.7z - a folder containing the test images in png format
- trainLabels.csv - the training labels

The label classes in the dataset are:

- airplane
- automobile
- bird
- cat
- deer
- dog

- frog
- horse
- ship
- truck

The classes are completely mutually exclusive. There is no overlap between automobiles and trucks. "Automobile" includes sedans, SUVs, things of that sort. "Truck" includes only big trucks. Neither includes pickup trucks.

4 Outline of approach

We are going to train a neuron networks to do classification.

Before load data, we will first do a preprocessing step. The 32×32 images have to be reshape to match the neuron networks input.

Then we can apply image augmentation to prevent overfitting. We can normalize the RGB values by dividing the x_train and x_test values by 255. Also, we could convert the images to grayscale, apply discrete histogram flattening, apply de-noising filters, or separate the phase and magnitude of the images to create more features if needed. These tasks are based on the error rate of neuron networks output.

After that, we will use CNN to generate an efficient classification model. We will find an appropriate activation function and the number of convolution layers and pooling layers. The fully-connected output layer needs a softmax activation to calculate the loss. To test how the model trains with respect to epochs, we can train the model on 20, 50, or more epochs.

Finally, we will validate our model on the testing data, evaluate the loss and accuracy and visualize the result.