Image Classification
Capstone Project Proposal

Thien Ngan Doan
Udacity ML Engineer Nanodegree
Jan 2021

Introduction

This Capstone project is part of the Udacity Machine Learning Engineer Nanodegree. When learning this program, we studied the problems of image classification with 2 classes and with dog breed project of 133 classes. This project is also an image classification with the famous image dataset CIFAR-10 containing 10 classes, we will go from the basic knowledge of image data to the process of training model and create its endpoint for predicting.

Domain Background

Cifar-10 is a very well-known database for deep learning example. There are many solved examples of this image classification problem, pytorch and keras were used popularly to teach learners for this problem.

This project is focused on how to create image data for a problem of image classification and step to step from executing hyperparameter tuning to training and deploying endpoint for predicting. This is a wonderful example and an awesome case of study.

My goal is studying each step from analyzing this type of data until predicting model to apply the same ideas in other similar projects in real productions on AWS.

Problem Statement

We use Pytorch to perform our models. We will use pretrained model densenet for our data. In this project we will use densenet161 model on AWS to get the most powerful prediction of our models.

We want to know the original byte data from cifar-10 and get all the images to organize three directory train, test, and valid for the problem of image classification. We will determine the process on AWS:

1. Extract data from original byte-format data.
2. Create the image data directories train, valid, and test.
3. Try some pretrained models on local machine to have a best selected model for perform on AWS.

4. Copy the image data directories *train*, *valid*, and *test* into s3 bucket.
5. Tuning hyperparameters on AWS.
6. Training on AWS
7. Create endpoint to predict.
8. Perform full prediction of 10,000 test images to get the accuracy and the result of each class accuracy.

Datasets and Inputs

We use the dataset provided by CIFAR-10 and CIFAR-100 datasets (toronto.edu).

The original data is an zip file of format tar.gz, it consists of 8 pickled files containing data and related information. Six important files of data are: *data_batch_1*, *data_batch_2*, *data_batch_3*, *data_batch_4*, *data_batch_5*, *test_batch*. From these files, we extract and process to transform them into three image data directories for training, validation, and testing. There are ten classes in our data.

class_id	class_name
0	airplane
1	automobile
2	bird
3	cat
4	deer
5	dog
6	frog
7	horse
8	ship
9	truck

- Training data: 40,000 images.
- Validation data: 10,000 images.
- Testing data: 10,000 images.

The distributions of ten classes in three folders are almost uniform.

Solution Statement

We use Pytorch to perform our models. Ours is model for the image classification problem. There are many popular famous pretrained models are used to build this kind of model, such

as: resnet, vgg, convNet, densenet, … In this project we will use densenet161 model on AWS to get the most powerful prediction of our models.

Benchmark Model

This project creates the image classification model. We can use many famous model such as: cnn, resnet, vgg, convNet, densenet, … We can compare the results with the same data CIFAR-10.

Evaluation Metrics

With this type of image classification problem, we use the best compatible metric is accuracy classification score.

Project Design

Our project gets original CIFAR-10 data - *cifar-10-python.tar.gz* - from the link Original CIFAR-10 Data. We perform the data processing as following:

I. Part I: Data processing
   1. Examine the original files cifar-10 and logically transform them into a whole package of images, related labels and save in 3 separated directories train, valid, and test.
   2. We create csv compatible files to help us get related information data and as an effective method of data backup.
   3. From this image data, we copy it to s3 bucket for training our model.

II. Part II: Modeling and predicting
   1. Pytorch pretrained desnet161 model is used for this problem.
   2. Use *sagemaker.tuner* to find best hyperparameters for our model.
   3. Use best hyperparameters for our *sagemaker.pytorch estimator*.
   4. Training on AWS Sagemaker.
   5. Use the training job to create endpoint for predicting.
   6. Get the predicted results to plot some charts and have conclusion.

References

1. CIFAR-10 and CIFAR-100 datasets (toronto.edu).
2. Original CIFAR-10 Data.
3. torchvision.models — Torchvision 0.11.0 documentation.
4. densenet-Torchvision 0.11.0-pytorch.org.