Handwritten Digit Classification

Project Overview

This project is based on Handwritten Digit Classification. Difference between handwritten and computerised digits can be predicted using this project. For character recognition, neural networks are commonly utilised [1, 2, 3, 4, 5, 6, 7, as well as 8]. We utilised the MNIST database to train and test a neural network classifier. Learning is a key stage in recognition; thus, we employed the gradient descent approach. The synaptic weight of the connections between the neurons is increased during the training phase modified. Attached binary neurons with no modifiable connections make up the initial layer.

Problem Statement

Handwritten digit identification is becoming increasingly important in a variety of new applications. Computer vision and machine learning researchers utilise it extensively for practical applications like scanning computerised bank check numbers. However, putting a computerised system in place to do certain tasks is a difficult and time-consuming task. It's difficult to distinguish one person's number handwriting from another since everyone writes differently. The number of characteristics and the classifiers used have a big influence in getting the greatest possible classification accuracy.

Submitted By:

Mansi Chilwant

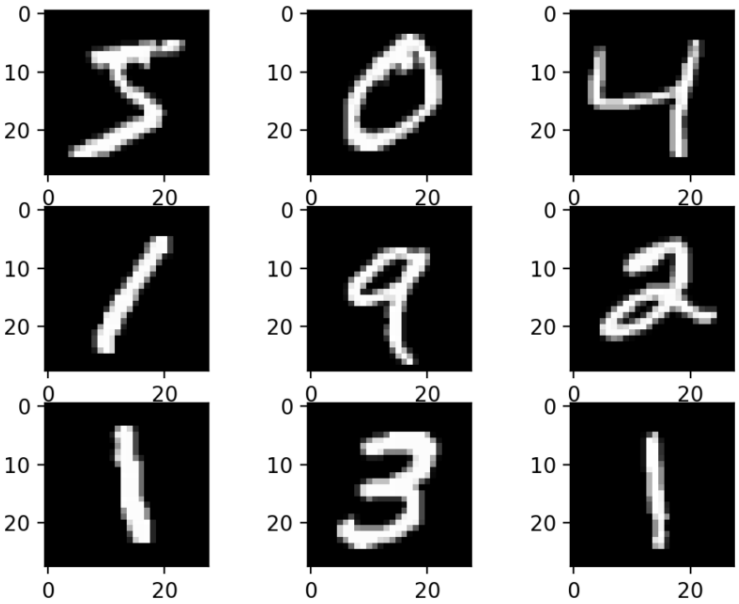

Figure 1: Example Handwritten Digit detection using machine learning

Datasets and Inputs

MNIST is a widely used dataset for handwritten digit classification. It consists of 70,000 labeled 28x28 pixel grayscale images of hand-written digits. The dataset is split into 60,000 training images and 10,000 test images. There are 10 classes (one for each of the 10 digits). This tutorial will show how to train and test an MNIST model on SageMaker using PyTorch.

Submitted By:

Mansi Chilwant

Solution Statement

We will be using sagemaker "Script mode". Script mode for PyTorch is a training script format that allows you to run any PyTorch training script in SageMaker. This example can be ran on one or multiple, cpu or gpu instances. The hyperparameters parameter is a dict of values that will be passed to your training script -- you can see how to access these values in the hpo.py script above.

After we've constructed our PyTorch object, we can fit it using the data we uploaded to S3. SageMaker makes sure our data is available in the local filesystem of each worker, so our training script can simply read the data from disk.

Benchmark Model:

The handwritten digit recognition following article. The author came up with some good analysis for using MNIST dataset for detection of handwritten digits.

The MNIST Database of Handwritten Digit Images for Machine Learning Research

Li Deng, Microsoft Research, Redmond, WA, USA

This project is a try to determine Handwritten Digit Images using the knowledge of the article.

Submitted By:

Mansi Chilwant

Evaluation Metrics

Since this is detection/classification problem, the accuracy of the analysis and detection of handwritten images will be considered for evaluation purpose.

Project Design

The project's concept is simple: initially, we'll develop a Sagemaker notebook instance that performs quite well. Download the dataset to the instance's environment, then upload to an S3 bucket for training purposes. Because the dataset is clean and includes manifest files. After that, we'll tune the model using the best available hyperparameters using Sagemaker tuner, and after that's done, we'll train the model on a GPU enabled instance using Sagemaker estimator. The model will then be deployed to a Sagemaker instance, and lambdas functions with proper permissions will be created to conveniently use the model.

These are steps will be followed.

- Create the SageMaker Session
- Training Data
- Train with SageMaker PyTorch Estimator
- Perform Batch Predictions

Submitted By:

Mansi Chilwant