Handwritten Digit Classification

Project Domain background

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, I 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.

A multi-layered convolutional neural network comprising one input layer, hidden layers, and one output layer is developed and shown to detect handwritten digits. The identification of handwritten digits has piqued academics' curiosity. This issue has resulted in a great number of papers and articles being published in recent years. Deep Learning methods such as multilayer CNN that use Pytorch for higher accuracy have been demonstrated in study.

The purpose of this project is to see how accurate CNN is in classifying handwritten digits using different numbers of hidden layers and epochs. The Modified National Institute of Standards and Technology (MNIST) dataset was used in this experiment to evaluate the performance of CNN.[1]

Submitted By:

Mansi Chilwant

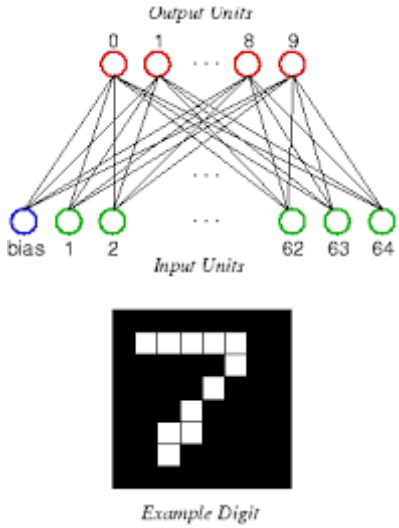


Figure 1: CNN for digit recognition

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.

The operation of a machine to train itself or recognise digits from various sources such as emails, bank checks, papers, images, and so on, and in various real-world scenarios such as online handwriting recognition on computer tablets or systems, recognise number plates of vehicles, process bank cheque amounts, numeric entries in forms filled out by hand (such as tax forms), and so on.

Submitted By:

Mansi Chilwant

Because handwritten digits varied in size, breadth, orientation, and justification to margins from person to person, the basic difficulty would be identifying the digits owing to the similarities between digits such as 1 and 7, 5 and 6, 3 and 8, 2 and 5, 2 and 7, and so on. This issue arises more frequently when a single digit is written by a large number of people in a diversity of handwriting styles. Finally, the construction and look of the digits are influenced by the individual's handwriting's originality and variation. Deep learning and machine learning ideas and methods are now introduced.

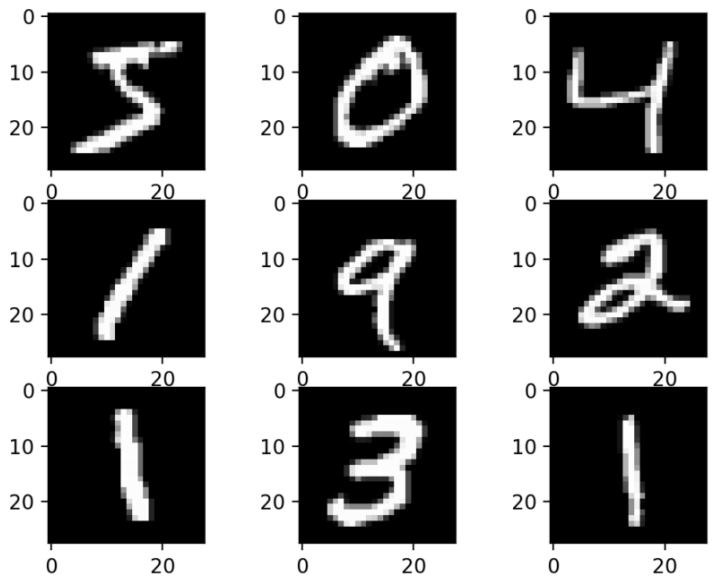


Figure 2: Example Handwritten Digit detection using machine learning

Submitted By:

Mansi Chilwant

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.

I entered AWS through the gateway in the course and open SageMaker Studio. Downloaded and made the dataset available.

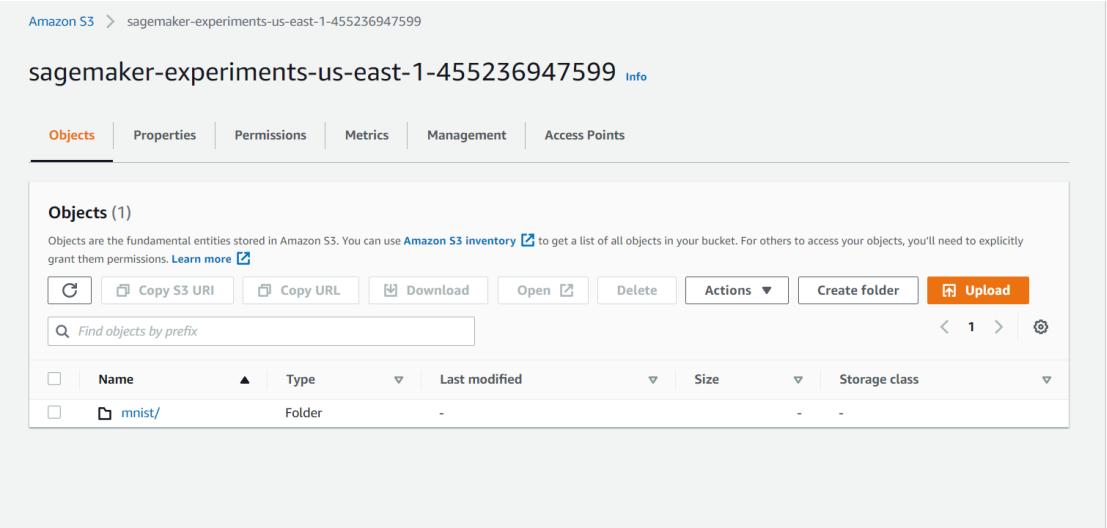


Figure 3: S3 Bucket Setup

Data exploration and visualisation:

After the data is uploaded to the s3 bucket, I normalise the data into the processable format.

Submitted By:

Mansi Chilwant

```
In [9]: plt.imshow(train_set.data[2].numpy())
```

```
Out[9]: <matplotlib.image.AxesImage at 0x7f63b4f78d50>
```

After transforming the images in the dataset, we upload it to s3.

```
In [10]: inputs = sagemaker.Session().upload_data(path="mnist", bucket=bucket, key_prefix=prefix)
         print("input spec: {}".format(inputs))
```

```
input spec: s3://sagemaker-experiments-us-east-1-455236947599/mnist
```

Now lets track the parameters from the data pre-processing step.

```
In [11]: with Tracker.create(display_name="Preprocessing", sagemaker_boto_client=sm) as tracker:
             tracker.log_parameters(
                 {
                     "normalization_mean": 0.1307,
                     "normalization_std": 0.3081,
                 }
             )
             # we can log the s3 uri to the dataset we just uploaded
             tracker.log_input(name="mnist-dataset", media_type="s3/uri", value=inputs)
```

Figure 4: Data exploration

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 the training script -- you can see how to access these values in the hpo.py script above.

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

Submitted By:

Mansi Chilwant

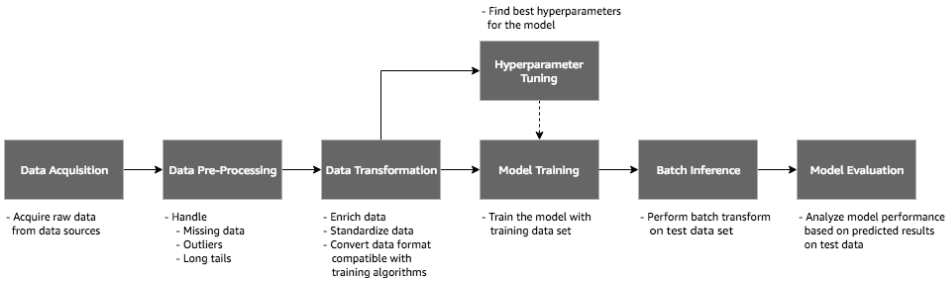


Figure 5: Amazon sagemaker machine learning workflow

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.

Evaluation Metrics

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

Submitted By:

Mansi Chilwant

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

Pre-processing:

- I entered AWS through the gateway in the course and open SageMaker Studio. Downloaded and made the dataset available.
- Started by installing all the requirements in the jupyter notebook " mnist-handwritten-digits-classification-experiment (1).ipynb."
- I used the basic "Python 3 (Data science - 01)"Kernel in this project. Also, essential PyTorch libraries have been installed.

Submitted By:

Mansi Chilwant

- The project run on SageMaker, such as Autopilot jobs and training jobs, will be logged automatically. You may also keep track of artefacts for other phases in an ML workflow that occur before or after model training, such as data pre-processing or model assessment.
- I trained a Convolutional Neural Network (CNN) model for this project. Adjust the number of hidden channels in the model by adjusting the hyperparameter. Using SageMaker Experiments, make track of parameter settings and model accuracy.
- I focused on various settings for the number of hidden channels in the CNN model while training it on SageMaker. We'll set up a Trial to keep track of each training task. I also made a Trial Component out of the tracker we made before and include it in the Trial. This will add the parameters we collected during the data pre-processing step to the Trial.
- I used 5 hidden channels i.e 2, 5, 10, 20, 32 to train the model
- Each Training job defines the individual hidden channel for training.

Submitted By:

Mansi Chilwant

Training code:

- Training using python script is performed. Script mode for PyTorch is a training script format that allows you to run any PyTorch training script in SageMaker with few changes.

- Transferring the script to a SageMaker training instance is handled by the SageMaker Python SDK. SageMaker's native PyTorch support on the training instance sets up training-related environment variables and runs the training script.

- I utilized the SageMaker Python SDK to initiate a training job and deploy the trained model in this tutorial.

```
# Based on https://github.com/pytorch/examples/blob/master/mnist/main.py
class Net(nn.Module):
    def __init__(self, hidden_channels, kernel_size, drop_out):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, hidden_channels, kernel_size=kernel_size)
        self.conv2 = nn.Conv2d(hidden_channels, 20, kernel_size=kernel_size)
        self.conv2_drop = nn.Dropout2d(p=drop_out)
        self.fc1 = nn.Linear(320, 50)
        self.fc2 = nn.Linear(50, 10)

    def forward(self, x):
        x = F.relu(F.max_pool2d(self.conv1(x), 2))
        x = F.relu(F.max_pool2d(self.conv2_drop(self.conv2(x)), 2))
        x = x.view(-1, 320)
        x = F.relu(self.fc1(x))
        x = F.dropout(x, training=self.training)
        x = self.fc2(x)
        return F.log_softmax(x, dim=1)


def _get_train_data_loader(batch_size, training_dir, is_distributed, **kwargs):
    logger.info("Get train data loader")
    dataset = datasets.MNIST(
        training_dir,
        train=True,
        transform=transforms.Compose(
            [transforms.ToTensor(), transforms.Normalize((0.1307,), (0.3081,))]
        ),
        download=False,
    )
    train_sampler = (
        torch.utils.data.distributed.DistributedSampler(dataset) if is_distributed else None
    )
    return torch.utils.data.DataLoader(
        dataset,
        batch_size=batch_size,
```

Submitted By:

Mansi Chilwant

Figure 6: Python script

- The mnist.py script contains all of the code required to train and host a SageMaker model (the model fn function is used to load a model). The training script is quite similar to a training script that you could run outside of SageMaker, however you may access important training environment attributes through several environment variables, such as:

❖ SM_MODEL_DIR: A string representing the path to the directory to write model artifacts to. These artifacts are uploaded to S3 for mode hosting.

❖ SM_CURRENT_HOST: The name of the current container on the container network.

❖ SM_HOSTS: JSON encoded list containing all the hosts.

❖ SM_HOSTS: JSON encoded list containing all the hosts.

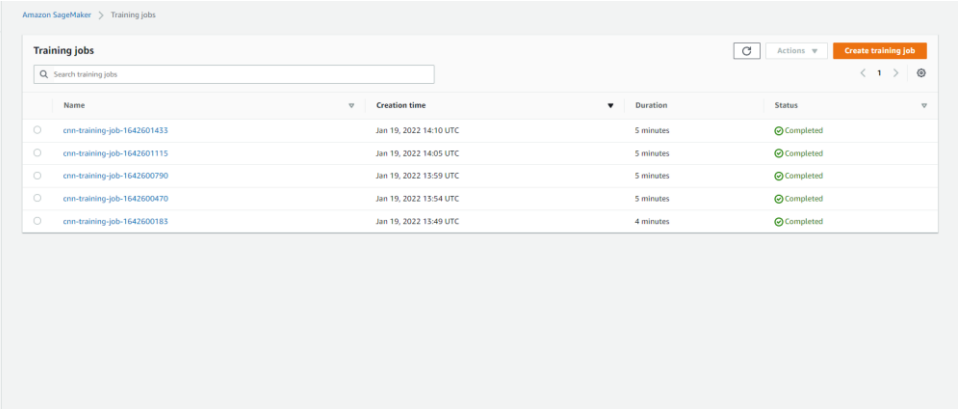


Figure 7: Training Jobs

Submitted By:

Mansi Chilwant

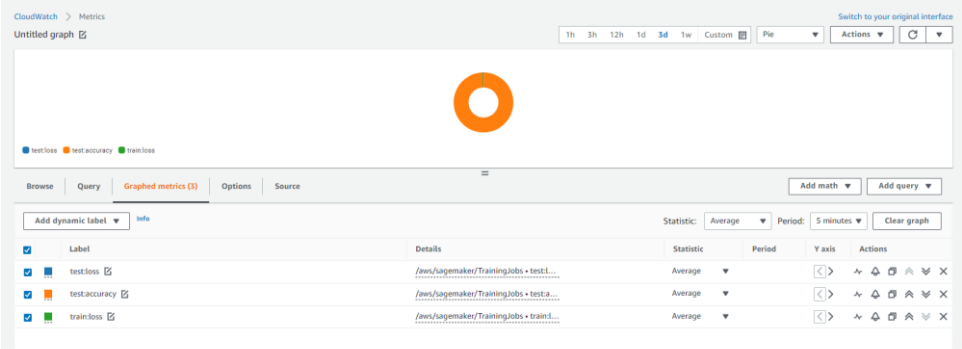
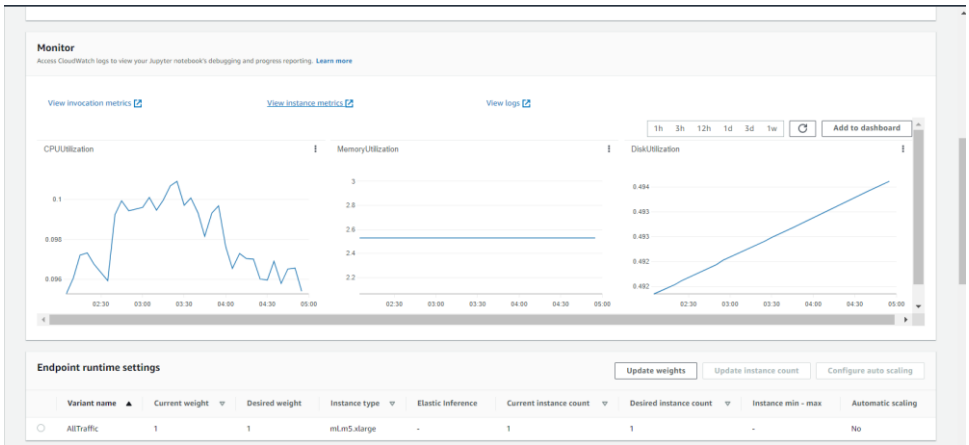


Figure 8: Training Jobs CloudWatch validation metrics

Deploying endpoints:

- The PyTorch class enables us to run training function on SageMaker infrastructure as a training task. I needed training script, an IAM role, the number of training instances, the kind of training instance, and hyperparameters to set it up. I executed training task on ml.m5.xlarge instance in this scenario.



Submitted By:

Mansi Chilwant

Figure 9: Deployed Endpoint CloudWatch validation metrics

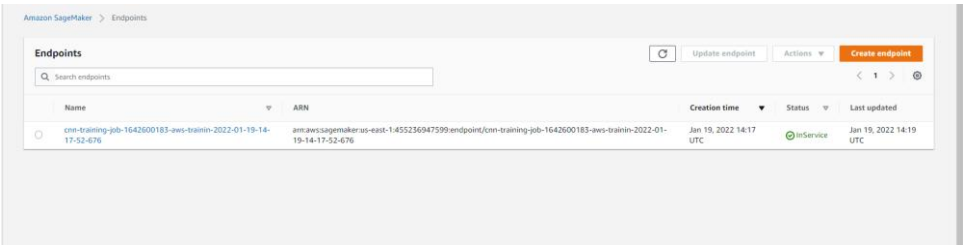


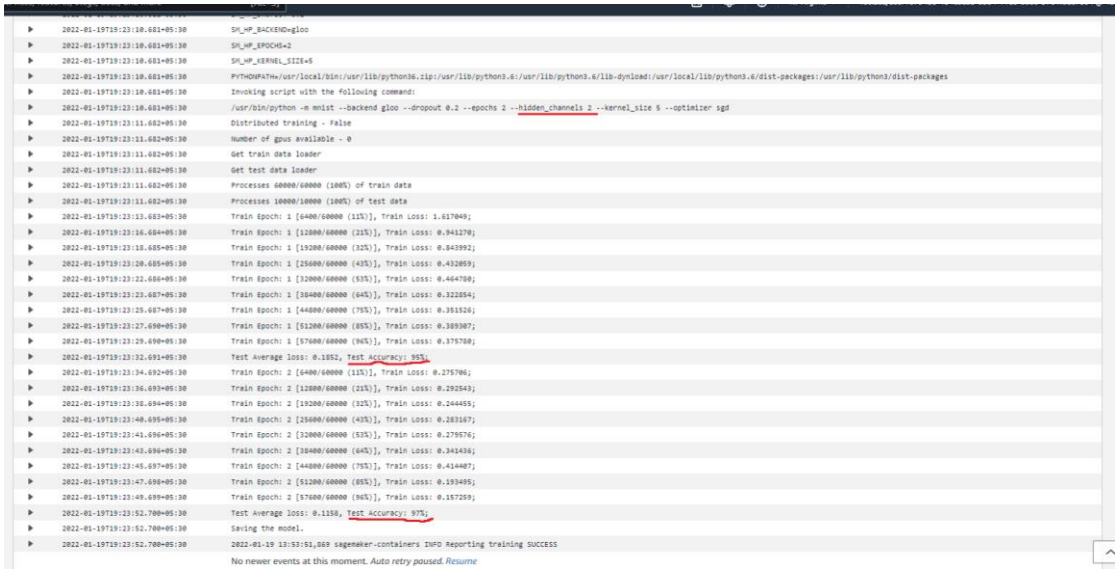
Figure 10: Endpoint

Evolutions:

- The summary of Training and endpoint alongside with CPU usage is provided in profiler report.
- Using several hidden channels, I was able to achieve ~ 97 percent accuracy for handwritten digit identification for two epochs.
- The aim of the project was to deliver the highest accuracy for handwritten digit recognition system suing Pytorch model. The tabular data confirms the highest accuracy using Pytorch model.

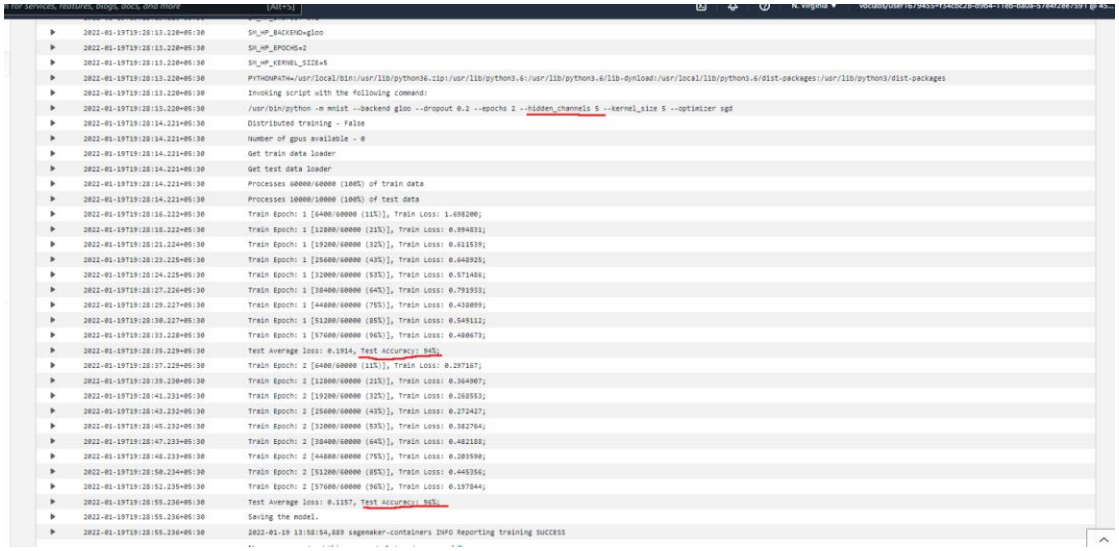
Submitted By:

Mansi Chilwant



```
2022-01-19T19:23:10.681+05:30   SM_HP_BACKEND=gloo
2022-01-19T19:23:10.681+05:30   SM_HP_EPOCHS=2
2022-01-19T19:23:10.681+05:30   SM_HP_KERNEL_SIZE=5
2022-01-19T19:23:10.681+05:30   PYTHONPATH=/usr/local/bin:/usr/lib/python36.zip:/usr/lib/python3.6:/usr/lib/python3.6/lib-dynload:/usr/local/lib/python3.6/dist-packages:/usr/lib/python3/dist-packages
2022-01-19T19:23:10.681+05:30   Invoking script with the following command:
2022-01-19T19:23:10.681+05:30   /usr/bin/python -m mnist --backend gloo --dropout 0.2 --epochs 2 --hidden_channels 2 --kernel_size 5 --optimizer sgd
2022-01-19T19:23:11.682+05:30   Distributed training - False
2022-01-19T19:23:11.682+05:30   Number of gpus available - 0
2022-01-19T19:23:11.682+05:30   Get train data loader
2022-01-19T19:23:11.682+05:30   Get test data loader
2022-01-19T19:23:11.682+05:30   Processes 60000/60000 (100%) of train data
2022-01-19T19:23:11.682+05:30   Processes 10000/10000 (100%) of test data
2022-01-19T19:23:13.683+05:30   Train Epoch: 1 [6400/60000 (11%)], Train Loss: 1.617049;
2022-01-19T19:23:16.684+05:30   Train Epoch: 1 [12800/60000 (21%)], Train Loss: 0.941270;
2022-01-19T19:23:18.685+05:30   Train Epoch: 1 [19200/60000 (32%)], Train Loss: 0.843992;
2022-01-19T19:23:20.685+05:30   Train Epoch: 1 [25600/60000 (43%)], Train Loss: 0.432059;
2022-01-19T19:23:22.686+05:30   Train Epoch: 1 [32000/60000 (53%)], Train Loss: 0.464780;
2022-01-19T19:23:23.687+05:30   Train Epoch: 1 [38400/60000 (64%)], Train Loss: 0.322854;
2022-01-19T19:23:25.687+05:30   Train Epoch: 1 [44800/60000 (75%)], Train Loss: 0.351526;
2022-01-19T19:23:27.690+05:30   Train Epoch: 1 [51200/60000 (85%)], Train Loss: 0.389307;
2022-01-19T19:23:29.690+05:30   Train Epoch: 1 [57600/60000 (96%)], Train Loss: 0.375700;
2022-01-19T19:23:32.691+05:30   Test Average loss: 0.1852, Test Accuracy: 95%;
2022-01-19T19:23:34.692+05:30   Train Epoch: 2 [6400/60000 (11%)], Train Loss: 0.275706;
2022-01-19T19:23:36.693+05:30   Train Epoch: 2 [12800/60000 (21%)], Train Loss: 0.292543;
2022-01-19T19:23:38.694+05:30   Train Epoch: 2 [19200/60000 (32%)], Train Loss: 0.244455;
2022-01-19T19:23:40.695+05:30   Train Epoch: 2 [25600/60000 (43%)], Train Loss: 0.283167;
2022-01-19T19:23:41.696+05:30   Train Epoch: 2 [32000/60000 (53%)], Train Loss: 0.279576;
2022-01-19T19:23:43.696+05:30   Train Epoch: 2 [38400/60000 (64%)], Train Loss: 0.341436;
2022-01-19T19:23:45.697+05:30   Train Epoch: 2 [44800/60000 (75%)], Train Loss: 0.414407;
2022-01-19T19:23:47.698+05:30   Train Epoch: 2 [51200/60000 (85%)], Train Loss: 0.193495;
2022-01-19T19:23:49.699+05:30   Train Epoch: 2 [57600/60000 (96%)], Train Loss: 0.157259;
2022-01-19T19:23:52.700+05:30   Test Average loss: 0.1158, Test Accuracy: 97%;
2022-01-19T19:23:52.700+05:30   Saving the model.
2022-01-19T19:23:52.700+05:30   2022-01-19 13:53:51,869 sagemaker-containers INFO Reporting training SUCCESS
                                No newer events at this moment. Auto retry paused. Resume
```

Figure 11: Training Job 1: Hidden Channel: 2, Test Accuracy of min.95% max 97%

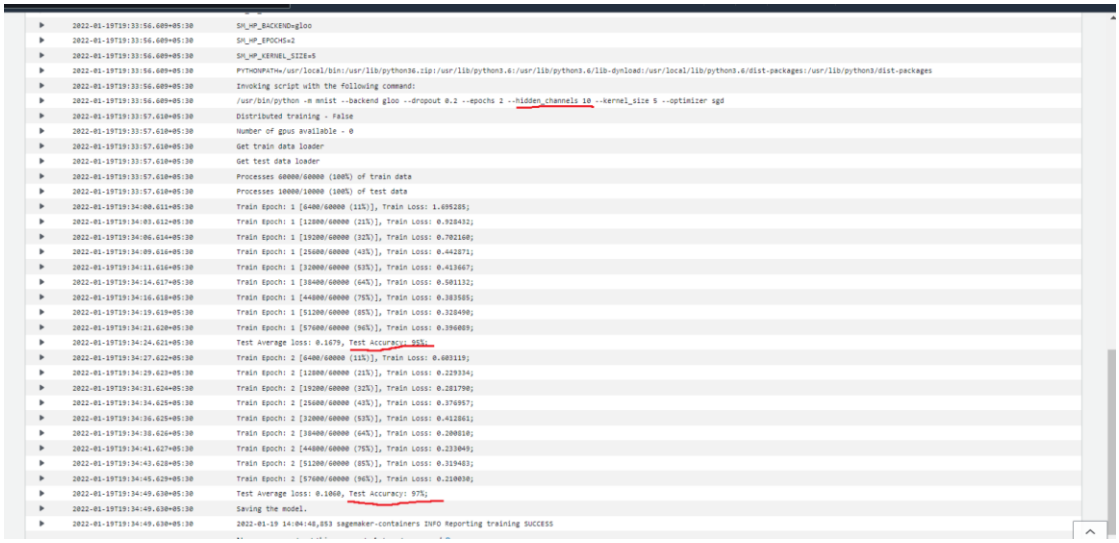


```
2022-01-19T19:28:13.220+05:30   SM_HP_BACKEND=gloo
2022-01-19T19:28:13.220+05:30   SM_HP_EPOCHS=2
2022-01-19T19:28:13.220+05:30   SM_HP_KERNEL_SIZE=5
2022-01-19T19:28:13.220+05:30   PYTHONPATH=/usr/local/bin:/usr/lib/python36.zip:/usr/lib/python3.6:/usr/lib/python3.6/lib-dynload:/usr/local/lib/python3.6/dist-packages:/usr/lib/python3/dist-packages
2022-01-19T19:28:13.220+05:30   Invoking script with the following command:
2022-01-19T19:28:13.220+05:30   /usr/bin/python -m mnist --backend gloo --dropout 0.2 --epochs 2 --hidden_channels 5 --kernel_size 5 --optimizer sgd
2022-01-19T19:28:14.221+05:30   Distributed training - False
2022-01-19T19:28:14.221+05:30   Number of gpus available - 0
2022-01-19T19:28:14.221+05:30   Get train data loader
2022-01-19T19:28:14.221+05:30   Get test data loader
2022-01-19T19:28:14.221+05:30   Processes 60000/60000 (100%) of train data
2022-01-19T19:28:14.221+05:30   Processes 10000/10000 (100%) of test data
2022-01-19T19:28:16.222+05:30   Train Epoch: 1 [6400/60000 (11%)], Train Loss: 1.698200;
2022-01-19T19:28:18.222+05:30   Train Epoch: 1 [12800/60000 (21%)], Train Loss: 0.994831;
2022-01-19T19:28:21.224+05:30   Train Epoch: 1 [19200/60000 (32%)], Train Loss: 0.611539;
2022-01-19T19:28:23.225+05:30   Train Epoch: 1 [25600/60000 (43%)], Train Loss: 0.648925;
2022-01-19T19:28:24.225+05:30   Train Epoch: 1 [32000/60000 (53%)], Train Loss: 0.571486;
2022-01-19T19:28:27.226+05:30   Train Epoch: 1 [38400/60000 (64%)], Train Loss: 0.791933;
2022-01-19T19:28:29.227+05:30   Train Epoch: 1 [44800/60000 (75%)], Train Loss: 0.430099;
2022-01-19T19:28:30.227+05:30   Train Epoch: 1 [51200/60000 (85%)], Train Loss: 0.549112;
2022-01-19T19:28:33.228+05:30   Train Epoch: 1 [57600/60000 (96%)], Train Loss: 0.480673;
2022-01-19T19:28:35.229+05:30   Test Average loss: 0.1914, Test Accuracy: 94%;
2022-01-19T19:28:37.229+05:30   Train Epoch: 2 [6400/60000 (11%)], Train Loss: 0.297167;
2022-01-19T19:28:39.230+05:30   Train Epoch: 2 [12800/60000 (21%)], Train Loss: 0.364907;
2022-01-19T19:28:41.231+05:30   Train Epoch: 2 [19200/60000 (32%)], Train Loss: 0.268553;
2022-01-19T19:28:43.232+05:30   Train Epoch: 2 [25600/60000 (43%)], Train Loss: 0.272427;
2022-01-19T19:28:45.232+05:30   Train Epoch: 2 [32000/60000 (53%)], Train Loss: 0.382764;
2022-01-19T19:28:47.233+05:30   Train Epoch: 2 [38400/60000 (64%)], Train Loss: 0.482188;
2022-01-19T19:28:48.233+05:30   Train Epoch: 2 [44800/60000 (75%)], Train Loss: 0.203590;
2022-01-19T19:28:50.234+05:30   Train Epoch: 2 [51200/60000 (85%)], Train Loss: 0.445356;
2022-01-19T19:28:52.235+05:30   Train Epoch: 2 [57600/60000 (96%)], Train Loss: 0.197844;
2022-01-19T19:28:55.236+05:30   Test Average loss: 0.1157, Test Accuracy: 96%;
2022-01-19T19:28:55.236+05:30   Saving the model.
2022-01-19T19:28:55.236+05:30   2022-01-19 13:58:54,889 sagemaker-containers INFO Reporting training SUCCESS
```

Submitted By:

Mansi Chilwant

Figure 12: Training Job 2: Hidden Channel: 5, Test Accuracy of min.94% max 96%

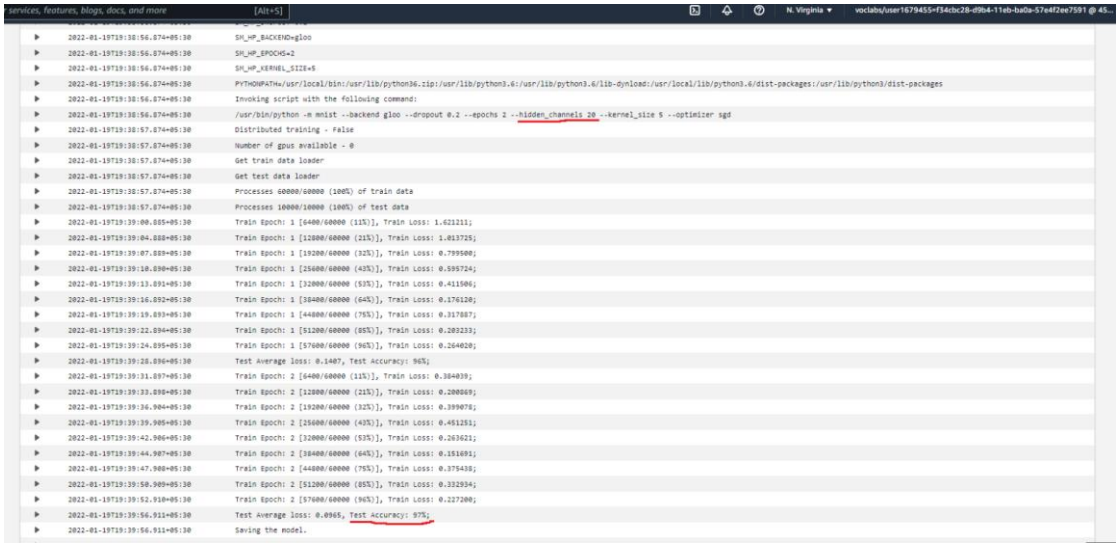


```
2022-01-19T19:33:56.609+05:30   SM_HP_BACKEND=gloo
2022-01-19T19:33:56.609+05:30   SM_HP_EPOCHS=2
2022-01-19T19:33:56.609+05:30   SM_HP_KERNEL_SIZE=5
2022-01-19T19:33:56.609+05:30   PYTHONPATH=/usr/local/bin:/usr/lib/python36.zip:/usr/lib/python3.6:/usr/lib/python3.6/lib-dynload:/usr/local/lib/python3.6/dist-packages:/usr/lib/python3/dist-packages
2022-01-19T19:33:56.609+05:30   Invoking script with the following command:
2022-01-19T19:33:56.609+05:30   /usr/bin/python -m mnist --backend gloo --dropout 0.2 --epochs 2 --hidden_channels 10 --kernel_size 5 --optimizer sgd
2022-01-19T19:33:57.610+05:30   Distributed training - False
2022-01-19T19:33:57.610+05:30   Number of gpus available - 0
2022-01-19T19:33:57.610+05:30   Get train data loader
2022-01-19T19:33:57.610+05:30   Get test data loader
2022-01-19T19:33:57.610+05:30   Processes 60000/60000 (100%) of train data
2022-01-19T19:33:57.610+05:30   Processes 10000/10000 (100%) of test data
2022-01-19T19:34:00.611+05:30   Train Epoch: 1 [6400/60000 (11%)], Train Loss: 1.695285;
2022-01-19T19:34:03.612+05:30   Train Epoch: 1 [12800/60000 (21%)], Train Loss: 0.928432;
2022-01-19T19:34:06.614+05:30   Train Epoch: 1 [19200/60000 (32%)], Train Loss: 0.702160;
2022-01-19T19:34:09.616+05:30   Train Epoch: 1 [25600/60000 (43%)], Train Loss: 0.442871;
2022-01-19T19:34:11.616+05:30   Train Epoch: 1 [32000/60000 (53%)], Train Loss: 0.413667;
2022-01-19T19:34:14.617+05:30   Train Epoch: 1 [38400/60000 (64%)], Train Loss: 0.501132;
2022-01-19T19:34:16.618+05:30   Train Epoch: 1 [44800/60000 (75%)], Train Loss: 0.383585;
2022-01-19T19:34:19.619+05:30   Train Epoch: 1 [51200/60000 (85%)], Train Loss: 0.328490;
2022-01-19T19:34:21.620+05:30   Train Epoch: 1 [57600/60000 (96%)], Train Loss: 0.396089;
2022-01-19T19:34:24.621+05:30   Test Average loss: 0.1679, Test Accuracy: 95%;
2022-01-19T19:34:27.622+05:30   Train Epoch: 2 [6400/60000 (11%)], Train Loss: 0.603119;
2022-01-19T19:34:29.623+05:30   Train Epoch: 2 [12800/60000 (21%)], Train Loss: 0.229334;
2022-01-19T19:34:31.624+05:30   Train Epoch: 2 [19200/60000 (32%)], Train Loss: 0.281790;
2022-01-19T19:34:34.625+05:30   Train Epoch: 2 [25600/60000 (43%)], Train Loss: 0.376957;
2022-01-19T19:34:36.625+05:30   Train Epoch: 2 [32000/60000 (53%)], Train Loss: 0.412861;
2022-01-19T19:34:38.626+05:30   Train Epoch: 2 [38400/60000 (64%)], Train Loss: 0.200810;
2022-01-19T19:34:41.627+05:30   Train Epoch: 2 [44800/60000 (75%)], Train Loss: 0.233049;
2022-01-19T19:34:43.628+05:30   Train Epoch: 2 [51200/60000 (85%)], Train Loss: 0.319483;
2022-01-19T19:34:45.629+05:30   Train Epoch: 2 [57600/60000 (96%)], Train Loss: 0.210030;
2022-01-19T19:34:49.630+05:30   Test Average loss: 0.1060, Test Accuracy: 97%;
2022-01-19T19:34:49.630+05:30   Saving the model.
2022-01-19T19:34:49.630+05:30   2022-01-19 14:04:48,853 sagemaker-containers INFO Reporting training SUCCESS
```

Figure 13: Training Job 3: Hidden Channel: 10, Test Accuracy of min.95% max 97%

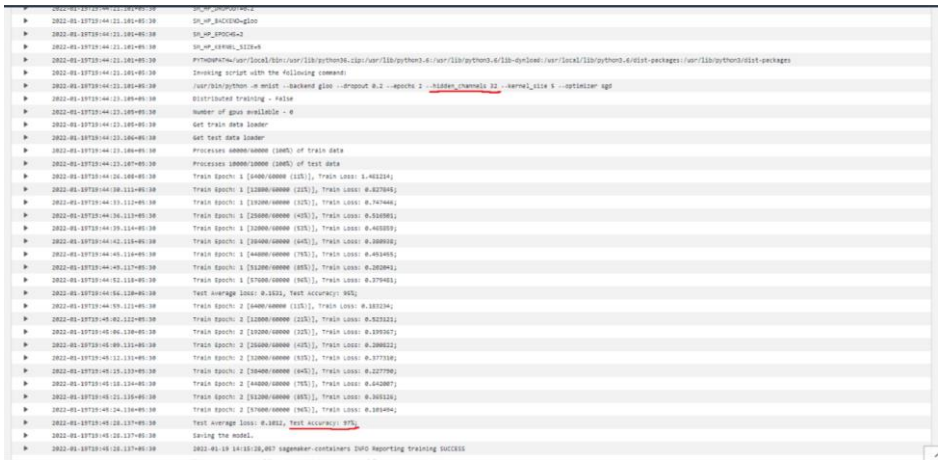
Submitted By:

Mansi Chilwant



```
services, features, blogs, docs, and more [Alt+S]                N. Virginia ▾   voclabs/user1679455=f34cbc28-d9b4-11eb-ba0a-57e4f2ee7591 @ 45...
2022-01-19T19:38:56.074+05:30   SM_HP_BACKEND=gloo
2022-01-19T19:38:56.074+05:30   SM_HP_EPOCHS=2
2022-01-19T19:38:56.074+05:30   SM_HP_KERNEL_SIZE=5
2022-01-19T19:38:56.074+05:30   PYTHONPATH=/usr/local/bin:/usr/lib/python36.zip:/usr/lib/python3.6:/usr/lib/python3.6/lib-dynload:/usr/local/lib/python3.6/dist-packages:/usr/lib/python3/dist-packages
2022-01-19T19:38:56.074+05:30   Invoking script with the following command:
2022-01-19T19:38:56.074+05:30   /usr/bin/python -m mnist --backend gloo --dropout 0.2 --epochs 2 --hidden_channels 20 --kernel_size 5 --optimizer sgd
2022-01-19T19:38:57.074+05:30   Distributed training - False
2022-01-19T19:38:57.074+05:30   Number of gpus available - 0
2022-01-19T19:38:57.074+05:30   Get train data loader
2022-01-19T19:38:57.074+05:30   Get test data loader
2022-01-19T19:38:57.074+05:30   Processes 60000/60000 (100%) of train data
2022-01-19T19:38:57.074+05:30   Processes 10000/10000 (100%) of test data
2022-01-19T19:39:00.885+05:30   Train Epoch: 1 [6400/60000 (11%)], Train Loss: 1.621211;
2022-01-19T19:39:04.888+05:30   Train Epoch: 1 [12800/60000 (21%)], Train Loss: 1.013725;
2022-01-19T19:39:07.889+05:30   Train Epoch: 1 [19200/60000 (32%)], Train Loss: 0.799500;
2022-01-19T19:39:10.890+05:30   Train Epoch: 1 [25600/60000 (43%)], Train Loss: 0.595724;
2022-01-19T19:39:13.891+05:30   Train Epoch: 1 [32000/60000 (53%)], Train Loss: 0.411506;
2022-01-19T19:39:16.892+05:30   Train Epoch: 1 [38400/60000 (64%)], Train Loss: 0.176120;
2022-01-19T19:39:19.893+05:30   Train Epoch: 1 [44800/60000 (75%)], Train Loss: 0.317887;
2022-01-19T19:39:22.894+05:30   Train Epoch: 1 [51200/60000 (85%)], Train Loss: 0.203233;
2022-01-19T19:39:24.895+05:30   Train Epoch: 1 [57600/60000 (96%)], Train Loss: 0.264020;
2022-01-19T19:39:28.896+05:30   Test Average loss: 0.1407, Test Accuracy: 96%;
2022-01-19T19:39:31.897+05:30   Train Epoch: 2 [6400/60000 (11%)], Train Loss: 0.384039;
2022-01-19T19:39:33.898+05:30   Train Epoch: 2 [12800/60000 (21%)], Train Loss: 0.200869;
2022-01-19T19:39:36.904+05:30   Train Epoch: 2 [19200/60000 (32%)], Train Loss: 0.399076;
2022-01-19T19:39:39.905+05:30   Train Epoch: 2 [25600/60000 (43%)], Train Loss: 0.451251;
2022-01-19T19:39:42.906+05:30   Train Epoch: 2 [32000/60000 (53%)], Train Loss: 0.263621;
2022-01-19T19:39:44.907+05:30   Train Epoch: 2 [38400/60000 (64%)], Train Loss: 0.151691;
2022-01-19T19:39:47.908+05:30   Train Epoch: 2 [44800/60000 (75%)], Train Loss: 0.375438;
2022-01-19T19:39:50.909+05:30   Train Epoch: 2 [51200/60000 (85%)], Train Loss: 0.332934;
2022-01-19T19:39:52.910+05:30   Train Epoch: 2 [57600/60000 (96%)], Train Loss: 0.227200;
2022-01-19T19:39:56.911+05:30   Test Average loss: 0.0965, Test Accuracy: 97%;
2022-01-19T19:39:56.911+05:30   Saving the model.
```

Figure 9: Training Job 4: Hidden Channel: 20, Test Accuracy of min.96% max 97%



```
2022-01-19T19:44:21.101+05:30   SM_HP_BACKEND=gloo
2022-01-19T19:44:21.101+05:30   SM_HP_EPOCHS=2
2022-01-19T19:44:21.101+05:30   SM_HP_KERNEL_SIZE=5
2022-01-19T19:44:21.101+05:30   PYTHONPATH=/usr/local/bin:/usr/lib/python36.zip:/usr/lib/python3.6:/usr/lib/python3.6/lib-dynload:/usr/local/lib/python3.6/dist-packages:/usr/lib/python3/dist-packages
2022-01-19T19:44:21.101+05:30   Invoking script with the following command:
2022-01-19T19:44:21.101+05:30   /usr/bin/python -m mnist --backend gloo --dropout 0.2 --epochs 2 --hidden_channels 32 --kernel_size 5 --optimizer sgd
2022-01-19T19:44:23.105+05:30   Distributed training - False
2022-01-19T19:44:23.105+05:30   Number of gpus available - 0
2022-01-19T19:44:23.105+05:30   Get train data loader
2022-01-19T19:44:23.106+05:30   Get test data loader
2022-01-19T19:44:23.106+05:30   Processes 60000/60000 (100%) of train data
2022-01-19T19:44:23.107+05:30   Processes 10000/10000 (100%) of test data
2022-01-19T19:44:26.108+05:30   Train Epoch: 1 [6400/60000 (11%)], Train Loss: 1.461214;
2022-01-19T19:44:30.111+05:30   Train Epoch: 1 [12800/60000 (21%)], Train Loss: 0.827845;
2022-01-19T19:44:33.112+05:30   Train Epoch: 1 [19200/60000 (32%)], Train Loss: 0.747446;
2022-01-19T19:44:36.113+05:30   Train Epoch: 1 [25600/60000 (43%)], Train Loss: 0.516581;
2022-01-19T19:44:39.114+05:30   Train Epoch: 1 [32000/60000 (53%)], Train Loss: 0.403859;
2022-01-19T19:44:42.115+05:30   Train Epoch: 1 [38400/60000 (64%)], Train Loss: 0.268938;
2022-01-19T19:44:45.116+05:30   Train Epoch: 1 [44800/60000 (75%)], Train Loss: 0.461455;
2022-01-19T19:44:49.117+05:30   Train Epoch: 1 [51200/60000 (85%)], Train Loss: 0.282841;
2022-01-19T19:44:52.118+05:30   Train Epoch: 1 [57600/60000 (96%)], Train Loss: 0.379481;
2022-01-19T19:44:56.120+05:30   Test Average loss: 0.1521, Test Accuracy: 95%;
2022-01-19T19:44:59.121+05:30   Train Epoch: 2 [6400/60000 (11%)], Train Loss: 0.183234;
2022-01-19T19:45:02.122+05:30   Train Epoch: 2 [12800/60000 (21%)], Train Loss: 0.525321;
2022-01-19T19:45:06.130+05:30   Train Epoch: 2 [19200/60000 (32%)], Train Loss: 0.195367;
2022-01-19T19:45:09.131+05:30   Train Epoch: 2 [25600/60000 (43%)], Train Loss: 0.200822;
2022-01-19T19:45:12.131+05:30   Train Epoch: 2 [32000/60000 (53%)], Train Loss: 0.377316;
2022-01-19T19:45:15.133+05:30   Train Epoch: 2 [38400/60000 (64%)], Train Loss: 0.227750;
2022-01-19T19:45:18.134+05:30   Train Epoch: 2 [44800/60000 (75%)], Train Loss: 0.642087;
2022-01-19T19:45:21.135+05:30   Train Epoch: 2 [51200/60000 (85%)], Train Loss: 0.365126;
2022-01-19T19:45:24.136+05:30   Train Epoch: 2 [57600/60000 (96%)], Train Loss: 0.181454;
2022-01-19T19:45:28.137+05:30   Test Average loss: 0.1012, Test Accuracy: 97%;
2022-01-19T19:45:28.137+05:30   Saving the model.
2022-01-19 14:15:28,057 sagemaker-containers INFO Reporting training SUCCESS
```

Figure 10: Training Job 5: Hidden Channel: 32, Test Accuracy of min.95% max 97%

Submitted By:

Mansi Chilwant

Reference:

1. Recognition of Handwritten Digit using Convolutional Neural Network in Python with Tensorflow and Comparison of Performance for Various Hidden Layers: Fathma Siddique1# , Shadman Sakib2⋆ , Md. Abu Bakr Siddique2$ 1 Department of CSE, International University of Business Agriculture and Technology, Dhaka 1230, Bangladesh 2 Department of EEE, International University of Business Agriculture and Technology, Dhaka 1230, Bangladesh siddiquefathma@gmail.com# , sakibshadman15@gmail.com⋆ , absiddique@iubat.edu

Submitted By:

Mansi Chilwant