# Project 3: Classification

# Report

Project Member #1

Harshal Anil Patil
UBIT Name: harshala
UB Person# 5024 5727

Project Member #2

Savit Varchasvi Aluri
UBIT Name: savitvar
UB Person# 5024 7220

## Introduction

The aim of this project is to implement and evaluate classification algorithms to classify hand written images of digits into 0,1, 2,…. 9 by training with the MNIST dataset. The models we built are:

1. Logistic regression (simple python AND tensorflow implementation)
2. Multi-layer perceptron with one hidden layer (Self-Implemented Back Propagation )
3. Convolutional neural network

All the models were tested on both MNIST test data as well as USPS data and the ***No-Free-lunch theorem*** was verified.

Platforms used: **Google Cloud Platform** was used to train and validate the CNN and MLP.

## Data Overview and Pre-processing steps:

### Training Data:

All classification models in this project are trained over MNIST data. Each image in the dataset is of size 28x28 and grayscale handwritten digit image. Value of pixel varies between 0 to 1 where 0 represents white and 1 represents black. These files were provided to us in the zipped –binary ('ubyte.gz') format. The format of this data is mentioned at:
http://yann.lecun.com/exdb/mnist/

#### Preprocessing:

1. We read the data byte by byte and stored them in a numpy array.
2. Since the pixels can hold a value between 0 – 255 we normalized them to be in between 0 – 1.
3. Since the labels of the images are target classes, we converted them into one-hot-vectors and later used the np.argmax() function to return the predicted label.

### Testing Data:

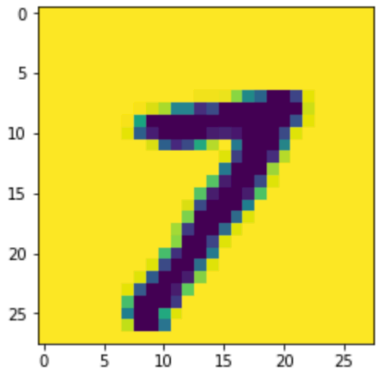
Two different types of testing data are used in this project, MNIST test data, and USPS data.

1. MNIST test data is similar to training data in nature, but USPS data is different.
2. Images in USPS data are of different size and each pixel is represented by the combination of three colors Red, Green, and Blue.
3. *Resizing of images into 28x28* and conversion of each image into grayscale was done before USPS data was used to test classification models in this project.

```
get_ipython().magic('matplotlib inline')
index = 40000
plt.imshow(train_set_X.reshape((60000, 28, 28))[index])
print(train_set_Y_one_hot[index])

# train_set_X.reshape((60000, 28, 28))[index]
```

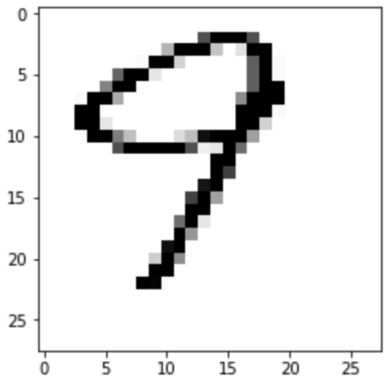
```
[ 0.  0.  0.  0.  0.  0.  0.  1.  0.  0.]
```



Preprocessing the USPS test-data:

1. Traversed iteratively through all the folders containing the labels and read the images into a list using the skimage library.
2. Each image was resized and centered to fit into a 28 X 28 box to match the dimensions of the MNIST data.
3. Since the pixels intensity values were different in USPS and MNIST, we inverted the pixel intensities to match those of MNIST. (255 – White -> 0 – White)
4. After this, similar pre-processing was done.

```
In [12]: plt.axis('on')
         plt.imshow(img_uniform[1000], cmap="gray")
         plt.show()
         print(labels_vt[1000])
```



```
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  1.]
```

| Data Set Type | # Images |
|---|---|
| MNIST training data | 60,000 |
| MNIST testing data | 10,000 |
| USPS testing data | 20,000 |

## Logistic Regression with TensorFlow:

TensorFlow was used to compute the model on MNIST training data to recognize digits by having it "look" at thousands of examples. Every image has a handwritten digit between 0 to 9. Thus, our model needs to classify each image into 10 different possibilities. Here softmax regression was used as opposed to the traditional sigmoidal regression.
A softmax regression contains two processes. First, we sum up the evidence of input image being in certain classes, and then we convert it into probabilities. Thus, each image is classified into one class for which it has maximum probability.
The Accuracy of the model is defined by how many correct classifications are done by model from total test data. During testing, it was found that our softmax regression model was giving accuracy around 0.92 for MNIST test data and 0.38 for USPS test data.
**Tuning the hyper parameters:** Hyper parameters like training steps and training data batch size also affected the accuracy of the model. During training, it was found that accuracy of model was slightly increased with increase in training steps and data batch size.

**Hyper-parameter tuning:**

Model accuracy based on various parameters:

| Training Iteration # | Training Steps | Data Batch Size | MNIST accuracy | USPS accuracy |
|---|---|---|---|---|
| 1 | 1000 | 100 | 0.9019 | 0.3816 |
| 2 | 2000 | 200 | 0.9115 | 0.3876 |
| 3 | 5000 | 500 | 0.9223 | 0.38921 |

## Logistic Regression using Numpy

This classification model is like our first regression model as both use softmax regression for classification of images. But this model is different from the previous one based on how calculations are performed to train the model. Functions are created from scratch to calculate the probability, cross entropy and accuracy of the regression.

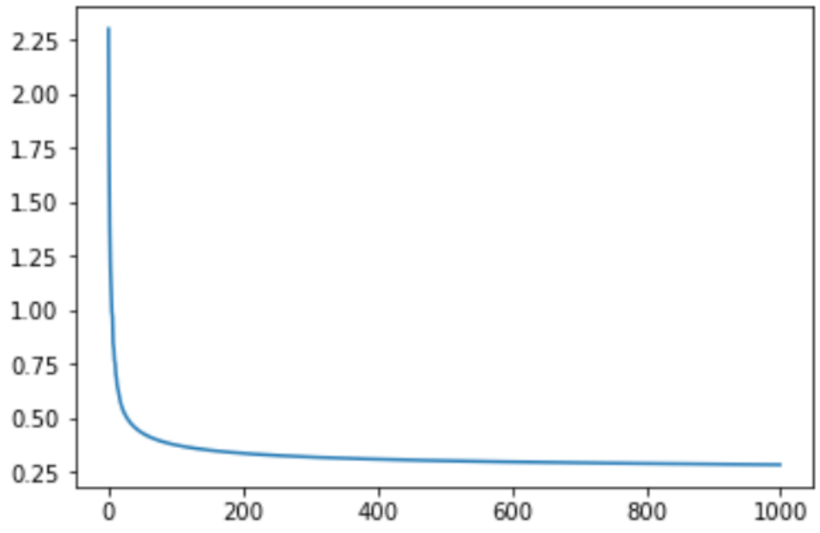
In this model, all calculations were done in python environment. Time taken by this model to train was slightly more as compared previous one because previous one was based on TensorFlow where calculations were done by creating graphs out of python environment. We optimized the error function using stochastic gradient descent and minimized the losses. Initially we used only the w for minimizing the error but later we used 'b' regularization also to increase the accuracy of the model.

After tuning the final hyper-parameters were:

1. Learning rate: 1e – 5
2. Lambda: 1
3. Iterations: 1000

The Accuracy of this model was **comparable** to the previous one. Model is giving 0.92 accuracy for MNIST test data and 0.36 for USPS data.

**Hyper-parameter tuning:**
Minimum loss: 0.286
Accuracy on MNIST test data:  0.9206
Accuracy on USPS data: 0.367268363418



Reduction in Loss with training steps

Model accuracy based on various parameters: (After many iterations)

| Training Iteration # | Training Steps | Learning rate | MNIST accuracy | USPS accuracy |
|---|---|---|---|---|
| 1 | 1000 | 1e-5 | 0.9009 | 0.3561 |
| 2 | 2000 | 1e-3 | 0.9113 | 0.3415 |
| 3 | 5000 | 1e-5 | 0.9206 | 0.36726 |

## MLP with 1 hidden layer:

This classification algorithm is a class of feedforward artificial neural network.

Completely implemented using numpy and no external libraries, this model contains one hidden layer apart from input and output layer. Except for the input nodes, each node is a neuron that uses a non-linear activation function.

MLP model too was initially done with 32 neurons in the hidden later and later this was also tuned, however it was found that this is mostly determined empirically and should lie between length of input layer and that of output layer.

## Backpropagation using Python:

Backpropagation is utilized by MLP as a supervised learning technique for training. Training occurs in the perceptron by changing connections weights after each batch of data is processed, based on the amount of error in the output compared to the expected result.

All the functions were defined in a class of NeuralNet, and other functions relating to activation and cost were also defined. Later gradient descent was used to determine the weights.

During testing, it was found that MLP based model gave more accuracy compared to logistic the regression-based model.

Tuned Hyperparameters:
Epochs: 50
Hidden layer perceptrons: 100
Mini-batch size: 20

### Hyper-parameter tuning:

Accuracy on MNIST test data: 0.9638
Accuracy on USPS data: 0.4356

Model accuracy based on various parameters:

| Training Iteration # | Epochs | Neurons in Hidden layer | MNIST accuracy | USPS accuracy |
|---|---|---|---|---|
| 1 | 20 | 32 | 0.9537 | 0.4202 |
| 2 | 10 | 100 | 0.9514 | 0.4037 |
| 3 | 50 | 200 | 0.9647 | 0.4381 |
| 4 | 30 | 100 | 0.9638 | 0.4356 |

During model training with different parameter values, it was found that the number of neurons had very less effect on accuracy on classification. The same can be seen in above table for Training iteration #2 and #3. #3 has double neurons as compared to 2 and still accuracy increased by some fraction. Therefore, we can assume that number of neurons for a model can be counted empirically.

## Convolution Neural Network

This model is a class of deep, feed-forward artificial neural networks. It uses a variation of multilayer perceptron designed to require minimal preprocessing. A CNN consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of convolutional layers, pooling layers, fully connected layers and normalization.

During testing on test data, it was found that CNN based model gave the highest accuracy for both MNIST test and USPS data.

The maximum accuracy was obtained using this model:
Accuracy on MNIST test: **99.27%**
Accuracy on USPS test: **69.44%**

Model accuracy based on various parameters:

| Training Iteration# | Training Steps | Data batch size | MNIST accuracy | USPS accuracy |
|---|---|---|---|---|
| 1 | 1000 | 50 | 0.9672 | 0.4367 |
| 2 | 5000 | 100 | 0.9710 | 0.4841 |
| 3 | 10000 | 50 | 0.9861 | 0.4914 |
| 4 | 20000 | 50 | 0.9927 | 0.6944 |

## Model Comparisons

Each model has some pros and cons.

1. CNN based classification model gave the highest accuracy for test data as compared to MLP with one hidden layer and logistic regression. Although accuracy was more, the time required to train the model was also high in multi folds.
2. The Logistic regression model with and without TensorFlow was quick to train but accuracy was poor as compared to other models.
3. Accuracy and time for training in MLP model was in between of CNN and Logistic regression.
4.

Therefore, we can say that CNN was the most accurate model than MLP with 1 hidden layer and lastly logistic regression.

## No-free lunch Theorem

There is a subtle issue that plagues all machine learning algorithms, summarized as the “no free lunch theorem”. The gist of this theorem is that you cannot get learning “for free” just by looking at training instances.

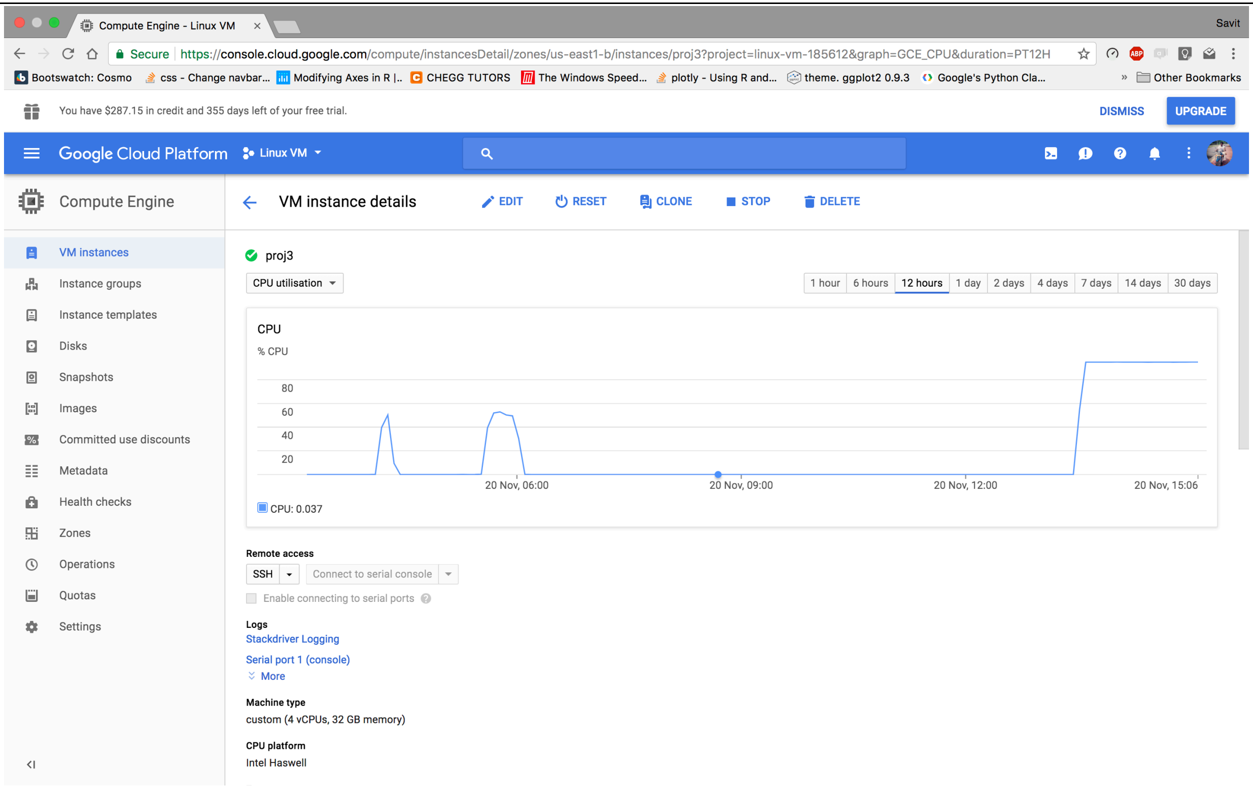
Similar is the case with four classification models in this project. Each model was trained on MNIST data and yet their accuracy was not perfect (zero error in classification) on MNIST test data. Not only this, accuracy drop by a huge margin on USPS data. ***If we take no free lunch theorem in perspective drop in accuracy on USPS data seems to be expected as models were trained on MNIST data and USPS data was new to them.***

Therefore, after training four models and testing them on two types of data with separate origin we can conclude **that our findings support No-free lunch theorem. The best results were produced in the CNN model with an error in classification as high as 52%.**

Results :

```
(tensorflow) savit@proj3:~/cse574_proj3/proj3_code$ python main.py
Project 3 : Classification
UBitnames : harshala & savitvar
UBit# : 50245727 & 50247220
Choose from below models:
1. Logistic Regression using TensorFlow
2. Logistic Regression using only Numpy
3. Multi layer Perceptron using Numpy and Back propagation
4. Convolutional Neural Network with 2 hidden layers
5. Exit the program
Enter Choice: 1
Logistic Regression on MNIST data using TensorFlow...
/home/savit/tensorflow/lib/python3.6/importlib/_bootstrap.py:219: RuntimeWarning: compiletime version 3.5
of module 'tensorflow.python.framework.fast_tensor_util' does not match runtime version 3.6
  return f(*args, **kwds)
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
Training the model on MNIST train data.. Please wait
Training Completed
The accuracy with MNIST test data:  0.9223
Let's see how this model performs with USPS data:
USPS test accuracy =  0.389219
Choose from below models:
1. Logistic Regression using TensorFlow
2. Logistic Regression using only Numpy
3. Multi layer Perceptron using Numpy and Back propagation
4. Convolutional Neural Network with 2 hidden layers
5. Exit the program
Which option do you want next? 2
Loss:  2.30258509299
Loss:  0.377436528238
Loss:  0.33871191589
Loss:  0.321567656313
Loss:  0.311294536003
Loss:  0.304230463837
```

```
Loss:  0.298965001492
Loss:  0.294825695412
Loss:  0.291447091438
Loss:  0.288611733147
The minimum loss is:  0.286203828486
Test Accuracy with MNIST Train:  0.920966666667
Test Accuracy with MNIST Test:  0.9206
Test Accuracy with USPS:  0.367268363418
Choose from below models:
1. Logistic Regression using TensorFlow
2. Logistic Regression using only Numpy
3. Multi layer Perceptron using Numpy and Back propagation
4. Convolutional Neural Network with 2 hidden layers
5. Exit the program
Which option do you want next? 3
MLP model with 100 neurons in the hidden layer..
Training and testing on MNIST dataset:
Accuracy is:  96.38
Testing on USPS Dataset:
Accuracy is:  43.56717835891794
Choose from below models:
1. Logistic Regression using TensorFlow
2. Logistic Regression using only Numpy
3. Multi layer Perceptron using Numpy and Back propagation
4. Convolutional Neural Network with 2 hidden layers
5. Exit the program
Which option do you want next? 5
Ran in Cloud:
(tensorflow) savit@proj3:~/cse574_proj3/proj3_code$ python main.py
Project 3 : Classification
UBitnames : harshala & savitvar
UBit# : 50245727 & 50247220
Choose from below models:
1. Logistic Regression using TensorFlow
2. Logistic Regression using only Numpy
3. Multi layer Perceptron using Numpy and Back propagation
4. Convolutional Neural Network with 2 hidden layers
5. Exit the program
Enter Choice: 4
'/home/savit/tensorflow/lib/python3.6/importlib/_bootstrap.py:219: RuntimeWarning: compiletime
version 3.5 of module 'tensorflow.python.framework.fast_tensor_util' does not match runtime version
3.6
  return f(*args, **kwds)
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
Training CNN begins..
Step 0, training accuracy 0.03
Step 100, training accuracy 0.87
Step 200, training accuracy 0.92
Step 300, training accuracy 0.93
Step 400, training accuracy 0.94
Step 500, training accuracy 0.97
Step 600, training accuracy 0.94
Step 700, training accuracy 0.98
Step 800, training accuracy 0.96
Step 900, training accuracy 0.98
Step 1000, training accuracy 0.99
Step 1100, training accuracy 0.97
Step 1200, training accuracy 0.98
Step 1300, training accuracy 0.99
Step 1400, training accuracy 0.97
Step 1500, training accuracy 0.98
Step 1600, training accuracy 0.98
Step 1700, training accuracy 0.98
Step 1800, training accuracy 1
Step 1900, training accuracy 0.98
Step 2000, training accuracy 0.98
Step 2100, training accuracy 0.99
Step 2200, training accuracy 1
Step 2300, training accuracy 0.97
Step 2400, training accuracy 1
Step 2500, training accuracy 0.97
Step 2600, training accuracy 1
Step 2700, training accuracy 0.99
Step 2800, training accuracy 0.99
Step 2900, training accuracy 1
Step 3000, training accuracy 0.99
Step 3100, training accuracy 0.98
Step 3200, training accuracy 0.98
Step 3300, training accuracy 0.98
Step 3400, training accuracy 0.98
Step 3500, training accuracy 0.98
Step 3600, training accuracy 0.98
Step 3700, training accuracy 1
Step 3800, training accuracy 0.99
Step 3900, training accuracy 0.99
Step 4000, training accuracy 0.98
Step 4100, training accuracy 0.99
Step 4200, training accuracy 1
Step 4300, training accuracy 1
Step 4400, training accuracy 0.99
Step 4500, training accuracy 1
Step 4600, training accuracy 1
Step 4700, training accuracy 0.99
Step 4800, training accuracy 0.98
Step 4900, training accuracy 0.99
Step 5000, training accuracy 1
Step 5100, training accuracy 0.99
Step 5200, training accuracy 0.99
Step 5300, training accuracy 1
Step 5400, training accuracy 0.99
Step 5500, training accuracy 0.99
Step 5600, training accuracy 1
Step 5700, training accuracy 0.99
Step 5800, training accuracy 0.99
Step 5900, training accuracy 1
Step 6000, training accuracy 0.99
Step 6100, training accuracy 1
Step 6200, training accuracy 1
Step 6300, training accuracy 1
Step 6400, training accuracy 0.99
Step 6500, training accuracy 1
Step 6600, training accuracy 1
Step 6700, training accuracy 0.98
Step 6800, training accuracy 1
Step 6900, training accuracy 1
Step 7000, training accuracy 0.99
Step 7100, training accuracy 1
Step 7200, training accuracy 1
Step 7300, training accuracy 1
Step 7400, training accuracy 1
Step 7500, training accuracy 0.99
Step 7600, training accuracy 0.98
Step 7700, training accuracy 1
Step 7800, training accuracy 1
Step 7900, training accuracy 1
Step 8000, training accuracy 1
Step 8100, training accuracy 1
Step 8200, training accuracy 1
Step 8300, training accuracy 0.99
Step 8400, training accuracy 0.99
Step 8500, training accuracy 1
Step 8600, training accuracy 1
Step 8700, training accuracy 1
Step 8800, training accuracy 1
Step 8900, training accuracy 1
Step 9000, training accuracy 0.99
Step 9100, training accuracy 1
Step 9200, training accuracy 1
Step 9300, training accuracy 1
Step 9400, training accuracy 1
Step 9500, training accuracy 1
Step 9600, training accuracy 1
Step 9700, training accuracy 1
Step 9800, training accuracy 1
Step 9900, training accuracy 1
Step 10000, training accuracy 1
Step 10100, training accuracy 1
Step 10200, training accuracy 1
Step 10300, training accuracy 1
Step 10400, training accuracy 1
Step 10500, training accuracy 1
Step 10600, training accuracy 1
Step 10700, training accuracy 0.99
Step 10800, training accuracy 0.99
Step 10900, training accuracy 1
Step 11000, training accuracy 1
Step 11100, training accuracy 1
Step 11200, training accuracy 1
Step 11300, training accuracy 1
Step 11400, training accuracy 1
Step 11500, training accuracy 1
Step 11600, training accuracy 1
Step 11700, training accuracy 1
Step 11800, training accuracy 1
Step 11900, training accuracy 0.99
Step 12000, training accuracy 1
Step 12100, training accuracy 1
Step 12200, training accuracy 1
Step 12300, training accuracy 1
Step 12400, training accuracy 1
Step 12500, training accuracy 1
Step 12600, training accuracy 1
Step 12700, training accuracy 1
Step 12800, training accuracy 1
Step 12900, training accuracy 1
Step 13000, training accuracy 1
Step 13100, training accuracy 1
Step 13200, training accuracy 1
Step 13300, training accuracy 1
Step 13400, training accuracy 1
Step 13500, training accuracy 1
Step 13600, training accuracy 1
Step 13700, training accuracy 1
Step 13800, training accuracy 1
Step 13900, training accuracy 1
Step 14000, training accuracy 1
Step 14100, training accuracy 1
Step 14200, training accuracy 1
Step 14300, training accuracy 1
Step 14400, training accuracy 1
Step 14500, training accuracy 1
Step 14600, training accuracy 1
Step 14700, training accuracy 1
Step 14800, training accuracy 1
Step 14900, training accuracy 1
Step 15000, training accuracy 1
Step 15100, training accuracy 1
Step 15200, training accuracy 1
Step 15300, training accuracy 1
Step 15400, training accuracy 1
Step 15500, training accuracy 1
Step 15600, training accuracy 1
Step 15700, training accuracy 1
Step 15800, training accuracy 1
Step 15900, training accuracy 1
Step 16000, training accuracy 1
Step 16100, training accuracy 1
Step 16200, training accuracy 1
Step 16300, training accuracy 1
Step 16400, training accuracy 1
Step 16500, training accuracy 1
Step 16600, training accuracy 1
Step 16700, training accuracy 1
Step 16800, training accuracy 1
Step 16900, training accuracy 1
Step 17000, training accuracy 1
Step 17100, training accuracy 1
Step 17200, training accuracy 1
Step 17300, training accuracy 1
Step 17400, training accuracy 1
Step 17500, training accuracy 1
Step 17600, training accuracy 1
Step 17700, training accuracy 1
Step 17800, training accuracy 1
Step 17900, training accuracy 1
Step 18000, training accuracy 1
Step 18100, training accuracy 1
Step 18200, training accuracy 1
Step 18300, training accuracy 1
Step 18400, training accuracy 1
Step 18500, training accuracy 1
Step 18600, training accuracy 1
Step 18700, training accuracy 1
Step 18800, training accuracy 1
Step 18900, training accuracy 1
Step 19000, training accuracy 1
Step 19100, training accuracy 1
Step 19200, training accuracy 1
Step 19300, training accuracy 1
Step 19400, training accuracy 1
Step 19500, training accuracy 1
Step 19600, training accuracy 1
Step 19700, training accuracy 1
Step 19800, training accuracy 1
Step 19900, training accuracy 1
Test accuracy on MNIST dataset: 0.9927
Test accuracy on USPS dataset: 0.693435
```



https://console.cloud.google.com/compute/instancesDetail/zones/us-east1-b/instances/proj3?project=linux-vm-185612&graph=GCE_CPU&duration=PT12H

You have $287.15 in credit and 355 days left of your free trial.

proj3

🟦 CPU: 0.037

**Remote access**

☐ Enable connecting to serial ports

**Logs**

**Machine type**

custom (4 vCPUs, 32 GB memory)

**CPU platform**

Intel Haswell

GCP Configured: 4 cores, 32 GB memory