Assignment 3

In this assignment, you will design a feed-forward neural network (also popularly known as a Multilayer Perceptron) classifier for the task. The ideal assignment to understand the working of neural networks would be to code up the backpropagation algorithm, taking into account variations in activation functions and loss functions.

Required Imports

Imports:

1. keras - Prediction Models
2. numpy : Support for Pandas and calculations
3. Matplotlib and seaborn - For visualization (Plotting graphs)
4. Sklearn : For evaluation

In [2]:
```
import numpy as np
# Use command line arguements for Task 1 (sys.argv)
import sys

# the answer to life, universe and everything. also for reproducibility
np.random.seed(42)
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
import seaborn as sns
from keras.models import Sequential
from keras.layers import Dense, Activation
from keras.utils import np_utils
from keras.optimizers import SGD
from keras.datasets import mnist
import matplotlib.pyplot as plt
%matplotlib inline
```

Loading the Dataset

About Dataset:

MNIST dataset has the following features:

Dataset size 60,000 samples of handwritten images.

1. The size of each image is 28x28 pixels.
2. Each image has only 1 color channel, i.e., grayscale image.
3. Each pixel has value in the range of [0,255] where 0 represents black, and 255 represents white.
4. Each image has labeled from 0-9.

In [3]:
```
# Load pre-shuffled MNIST data into train and test sets
(X_train_1, train_labels_1), (X_test, test_labels_1) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.
npz
11493376/11490434 [==============================] - 0s 0us/step
```

Preparing Training and Validation data and Normalizing the pixel data

It requires a few steps:

MNIST provides a total 70,000 examples, divided into a test set of 10,000 images and a training set of 60,000 images. In this assignment, I curve out a validation set of 10,000 images from the MNIST training set, and use the remaining 50,000 examples for training.

The pixel data is also normalized. Each pixel values lies between [0,255]. This value range is too high and it will be difficult for any model to learn. The best approach is normalize the data. In this case, as the pixel value is in the known range it sufficient to scale the pixel values in range [0,1] by simply dividing the array by 255.

In [4]:
```
 # Preprocess input data
X_train_1 = X_train_1.reshape(60000, 784)
X_test = X_test.reshape(10000, 784)
X_train_1 = X_train_1.astype('float32')
X_test = X_test.astype('float32')

X_train_1 /= 255
X_test /= 255
```

In [5]:
```
# Divides the dataset into train and validation sets
X_valid = X_train_1[50000:60000]
X_train = X_train_1[:50000]
print(X_train.shape[0], 'train samples')
print(X_valid.shape[0], 'validation samples')
```

```
50000 train samples
10000 validation samples
```

In [47]:
```
# Preprocess class labels
train_labels = np_utils.to_categorical(train_labels_1, 10)
test_labels = np_utils.to_categorical(test_labels_1, 10)
valid_labels = train_labels[50000:60000]
train_labels = train_labels[:50000]
```

Buliding Model

In [7]:
```
# Define model architecture
model = Sequential()

model.add(Dense( 100, input_shape=(784,)))
model.add(Activation('relu'))

model.add(Dense(100))
model.add(Activation('relu'))

model.add(Dense(10))
model.add(Activation('softmax'))
```

Training

model.fit() is used to train the model. It takes training data, batch_size, no of epochs, validation data. There are several more parameters. I am are taking the epochs = 3 and batch size = 32.

Loss Functions:

Keras provides all of the well-known loss functions which work well for most of the time. I am discussing 3 loss function here.

1. binary_crossentropy: This loss function is used for the binary classification task. The single-node output layer is required. 0 and 1 is used for classification.
2. categorical_crossentropy: Used for Used for Multi-class classification
3. sparse_categorical_crossentropy: Used for Multi-class classification.

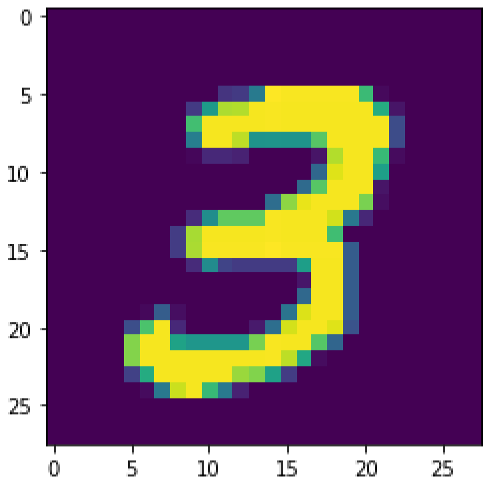
Difference between categorical_crossentropy and sparse_categorical_crossentropy is if your targets are one-hot encoded, use categorical_crossentropy. But if your targets are integers, use sparse_categorical_crossentropy.

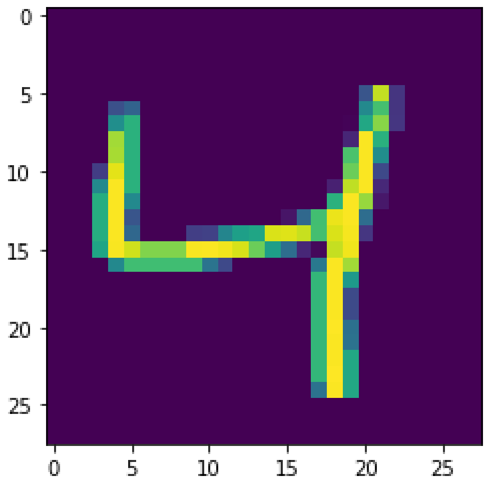
In [14]:
```
# Compile model
sgd = SGD(lr=0.01) # Sets learning rate.
model.compile(loss='categorical_crossentropy',
              optimizer=sgd,
              metrics=['accuracy'])

# Fit model on training data
model.fit(X_train, train_labels,batch_size=32, epochs=3, verbose=1)

# Evaluate model on test data.
score = model.evaluate(X_valid, valid_labels, verbose=0)
# This returns only a score, so you will need to use another function for
# extracting predicted labels for your confusion matrix. Use this line for that:
classes = model.predict_classes(X_test, batch_size=32)
print(classes)
print('Validation score:', score[0])
print('Validation accuracy:', score[1])
f=open("results-<parameter_name>.csv","a")
f.write(sys.argv[1]+","+str(score[1])+"\n" )
f.close()
```

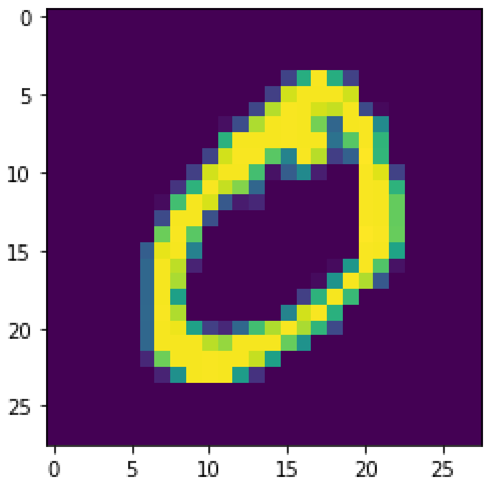
```
Epoch 1/3
1563/1563 [==============================] - 3s 2ms/step - loss: 0.1343 - accuracy: 0.96
01
Epoch 2/3
1563/1563 [==============================] - 3s 2ms/step - loss: 0.1250 - accuracy: 0.96
34
Epoch 3/3
1563/1563 [==============================] - 3s 2ms/step - loss: 0.1167 - accuracy: 0.96
55
[7 2 1 ... 4 5 6]
Validation score: 0.1269664615392685
Validation accuracy: 0.964900016784668
```

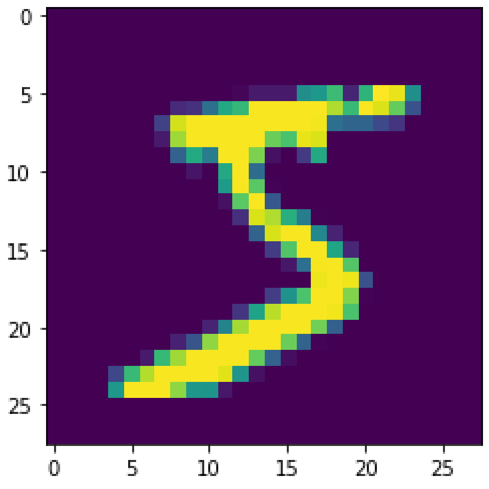
In [15]:
```
# A few random samples
#use_samples = [5, 38, 3939, 27389]
samples_to_predict = []

# Generate plots for samples
```

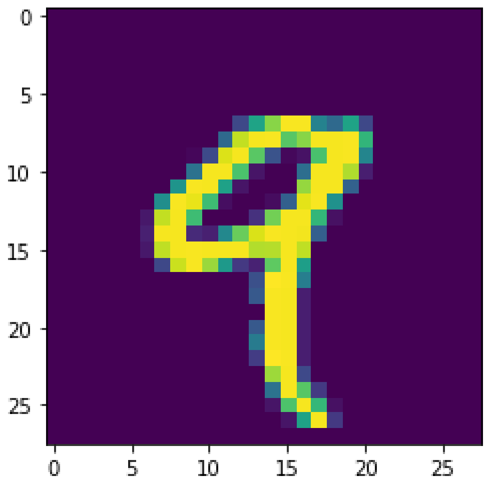
```
i = 0
for sample in classes:
  # Generate a plot
  reshaped_image = X_train[sample].reshape((28, 28))
  plt.imshow(reshaped_image)
  plt.show()
  # Add sample to array for prediction
  samples_to_predict.append(X_train[sample])
  i += 1
  if i == 10:
    break

# Convert into Numpy array
samples_to_predict = np.array(samples_to_predict)
print(samples_to_predict.shape)

# Generate predictions for samples
predictions = model.predict(samples_to_predict)
print(predictions)

# Generate arg maxes for predictions
classes = np.argmax(predictions, axis = 1)
print(classes)
```

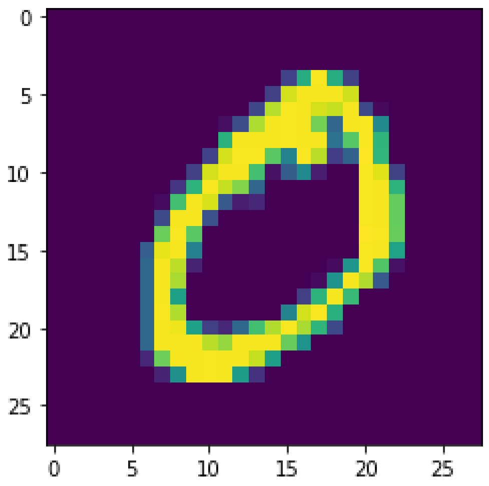


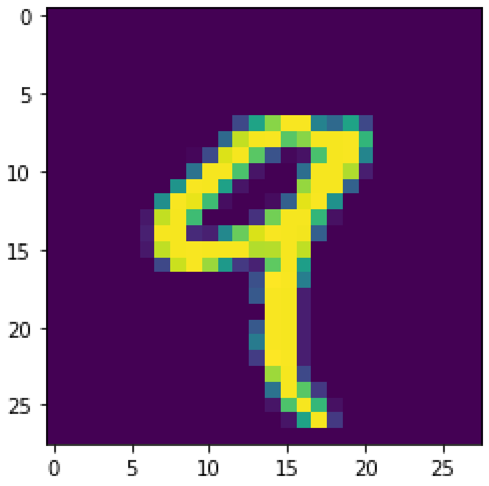


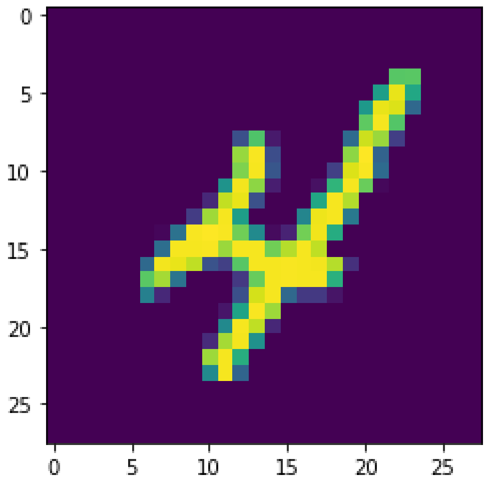

0
5
10
15
20
25
0
5
10
15
20
25

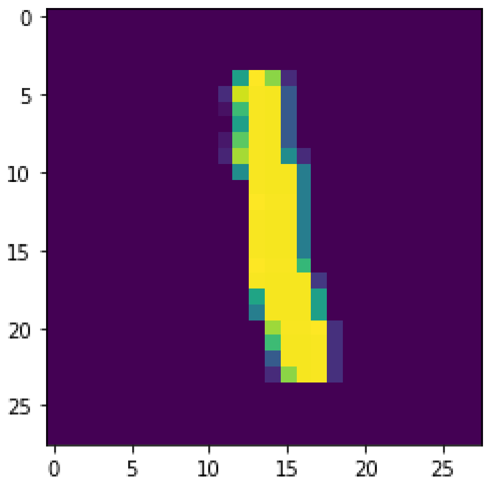

0
5
10
15
20
25
0
5
10
15
20
25

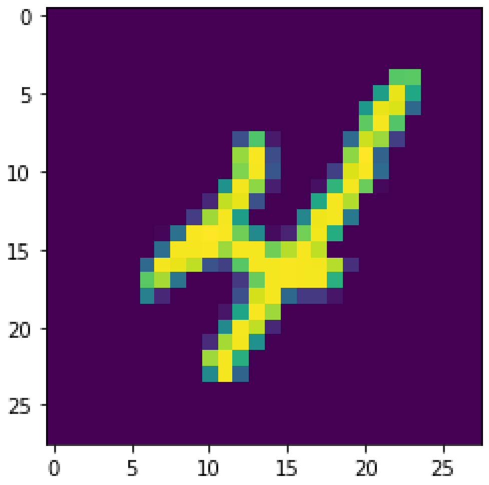

0
5
10
15
20
25
0
5
10
15
20
25


0
5
10
15
20
25
0
5
10
15
20
25


0
5
10
15
20
25
0
5
10
15
20
25


0
5
10
15
20
25
0
5
10
15
20
25





```
(10, 784)
[[3.81284099e-06 1.63422467e-08 8.91583622e-05 9.99717414e-01
  3.89885624e-11 1.06450570e-05 2.27254063e-10 1.32220555e-06
  8.94269469e-05 8.83098473e-05]
 [2.67994154e-04 1.05582665e-04 9.47761908e-03 5.68763213e-03
  9.33882892e-01 1.02023862e-03 2.34547933e-03 2.32850891e-02
  6.75440067e-04 2.32521147e-02]
 [9.99246240e-01 5.98922156e-09 6.71401678e-04 2.18753339e-05
  2.02497716e-07 2.76428750e-06 1.54422960e-05 1.12880043e-05
  1.12770604e-05 1.94912773e-05]
 [4.00830658e-07 2.69610132e-06 3.78836776e-05 3.87113206e-02
  9.40921386e-12 9.61237609e-01 8.45115711e-09 9.43654140e-06
  1.66527613e-07 3.76339500e-07]
 [6.27208976e-07 4.14763490e-05 8.34278751e-07 6.29684218e-05
  2.43696831e-02 9.11149691e-05 1.90619957e-07 6.68731751e-04
  7.61848642e-04 9.74002600e-01]
 [9.99246240e-01 5.98922156e-09 6.71401678e-04 2.18753339e-05
  2.02497716e-07 2.76428750e-06 1.54422960e-05 1.12880043e-05
  1.12770604e-05 1.94912773e-05]
 [6.27208976e-07 4.14763490e-05 8.34278751e-07 6.29684218e-05
  2.43696831e-02 9.11149691e-05 1.90619957e-07 6.68731751e-04
  7.61848642e-04 9.74002600e-01]
 [6.54729683e-06 8.06818719e-08 1.14714185e-05 1.05621768e-06
  9.99702752e-01 5.84225745e-05 1.76044763e-04 2.70650598e-05
  4.58959767e-06 1.19822480e-05]
 [8.10106537e-10 9.99006450e-01 5.62435926e-05 5.74549194e-04
  7.11463463e-06 4.20026163e-06 7.97856683e-06 9.14016437e-06
  3.30778625e-04 3.60915078e-06]
 [6.54729683e-06 8.06818719e-08 1.14714185e-05 1.05621768e-06
  9.99702752e-01 5.84225745e-05 1.76044763e-04 2.70650598e-05
  4.58959767e-06 1.19822471e-05]]
[3 4 0 5 9 0 9 4 1 4]
```

TASK 1 : Analysing performance

In this task, I will evaluate the performance of your network for varying values of hyperparameters. Keeping the rest of the values constant (and equal to the default values), adjust the values of parameters as described below. Find the performance (accuracy) of your model on the validation set and plot a trend graph for each of the following.

Batch size: 1, 2, 4, 8, 16, 32, 64, 128 (default 32).

Number of hidden layers: 1, 2, 4, 6, 8 (default 2).

Learning Rate: 0.01, 0.05, 0.1, 0.2, 0.4, 0.8 (default 0.01).

In [16]:

```
# Varying batch size
sgd = SGD(lr=0.01) # Sets learning rate.
model.compile(loss='categorical_crossentropy',
              optimizer=sgd,
              metrics=['accuracy'])

# Fit model on training data
batch = [1, 2, 4, 8, 16, 32, 64, 128]
batch_par = []
batch_val = []
for b in batch:
  model.fit(X_train, train_labels,batch_size=b, epochs=3, verbose=1)

  # Evaluate model on test data.
  score = model.evaluate(X_valid, valid_labels, verbose=0)
  # This returns only a score, so you will need to use another function for
  # extracting predicted labels for your confusion matrix. Use this line for that:
  classes = model.predict_classes(X_test, batch_size=b)
  print(classes)
  print('Validation score for batch ', b, ' is ', score[0])
  print('Validation accuracy:', score[1])
  batch_par.append(b)
  batch_val.append(score[1])
```

```
Epoch 1/3
50000/50000 [==============================] - 50s 1ms/step - loss: 0.1878 - accuracy:
0.9431
Epoch 2/3
50000/50000 [==============================] - 50s 999us/step - loss: 0.1046 - accuracy:
0.9680
Epoch 3/3
50000/50000 [==============================] - 50s 1ms/step - loss: 0.0788 - accuracy:
0.9754
[7 2 1 ... 4 5 6]
Validation score for batch  1  is  0.1077638566493988
Validation accuracy: 0.968500018119812
Epoch 1/3
25000/25000 [==============================] - 39s 2ms/step - loss: 0.0378 - accuracy:
0.9883
Epoch 2/3
25000/25000 [==============================] - 37s 1ms/step - loss: 0.0241 - accuracy:
0.9927
Epoch 3/3
25000/25000 [==============================] - 36s 1ms/step - loss: 0.0183 - accuracy:
```

```
0.9945
[7 2 1 ... 4 5 6]
Validation score for batch  2  is  0.0783177837729454
Validation accuracy: 0.9793999791145325
Epoch 1/3
12500/12500 [==============================] - 18s 1ms/step - loss: 0.0100 - accuracy:
0.9976
Epoch 2/3
12500/12500 [==============================] - 18s 1ms/step - loss: 0.0067 - accuracy:
0.9986
Epoch 3/3
12500/12500 [==============================] - 18s 1ms/step - loss: 0.0053 - accuracy:
0.9990
[7 2 1 ... 4 5 6]
Validation score for batch  4  is  0.0776519924402237
Validation accuracy: 0.9818999767303467
Epoch 1/3
6250/6250 [==============================] - 10s 2ms/step - loss: 0.0039 - accuracy: 0.9
994
Epoch 2/3
6250/6250 [==============================] - 10s 2ms/step - loss: 0.0034 - accuracy: 0.9
996
Epoch 3/3
6250/6250 [==============================] - 10s 2ms/step - loss: 0.0031 - accuracy: 0.9
996
[7 2 1 ... 4 5 6]
Validation score for batch  8  is  0.07973223924636841
Validation accuracy: 0.9821000099182129
Epoch 1/3
3125/3125 [==============================] - 5s 2ms/step - loss: 0.0027 - accuracy: 0.99
98
Epoch 2/3
3125/3125 [==============================] - 5s 2ms/step - loss: 0.0026 - accuracy: 0.99
98
Epoch 3/3
3125/3125 [==============================] - 5s 2ms/step - loss: 0.0025 - accuracy: 0.99
98
[7 2 1 ... 4 5 6]
Validation score for batch  16  is  0.08066857606172562
Validation accuracy: 0.9818000197410583
Epoch 1/3
1563/1563 [==============================] - 3s 2ms/step - loss: 0.0024 - accuracy: 0.99
99
Epoch 2/3
1563/1563 [==============================] - 3s 2ms/step - loss: 0.0023 - accuracy: 0.99
99
Epoch 3/3
1563/1563 [==============================] - 3s 2ms/step - loss: 0.0023 - accuracy: 0.99
99
[7 2 1 ... 4 5 6]
Validation score for batch  32  is  0.0812508836388588
Validation accuracy: 0.9817000031471252
Epoch 1/3
782/782 [==============================] - 2s 2ms/step - loss: 0.0022 - accuracy: 0.9999
Epoch 2/3
782/782 [==============================] - 2s 2ms/step - loss: 0.0022 - accuracy: 0.9999
Epoch 3/3
782/782 [==============================] - 2s 2ms/step - loss: 0.0022 - accuracy: 0.9999
[7 2 1 ... 4 5 6]
Validation score for batch  64  is  0.08153489977121353
Validation accuracy: 0.9818000197410583
Epoch 1/3
391/391 [==============================] - 1s 3ms/step - loss: 0.0022 - accuracy: 0.9999
Epoch 2/3
391/391 [==============================] - 1s 3ms/step - loss: 0.0021 - accuracy: 0.9999
```

```
Epoch 3/3
391/391 [==============================] - 1s 3ms/step - loss: 0.0021 - accuracy: 0.9999
[7 2 1 ... 4 5 6]
Validation score for batch  128  is  0.08166895806789398
Validation accuracy: 0.9818000197410583
```

In [17]:
```
# Varying learning rate

# Fit model on training data
l_rate = [0.01, 0.05, 0.1, 0.2, 0.4, 0.8]
lr_par = []
lr_val = []
for l in l_rate:
  model.compile(loss='categorical_crossentropy',
              optimizer=SGD(lr = l),
              metrics=['accuracy'])
  model.fit(X_train, train_labels,batch_size=32, epochs=3, verbose=1)

  # Evaluate model on test data.
  score = model.evaluate(X_valid, valid_labels, verbose=0)
  # This returns only a score, so you will need to use another function for
  # extracting predicted labels for your confusion matrix. Use this line for that:
  classes = model.predict_classes(X_test, batch_size=32)
  print(classes)
  print('Validation score for learning rate ', l, ' is', score[0])
  print('Validation accuracy:', score[1])
  lr_par.append(l)
  lr_val.append(score[1])
```

```
Epoch 1/3
1563/1563 [==============================] - 2s 2ms/step - loss: 0.0022 - accuracy: 0.99
99
Epoch 2/3
1563/1563 [==============================] - 2s 2ms/step - loss: 0.0021 - accuracy: 0.99
99
Epoch 3/3
1563/1563 [==============================] - 2s 1ms/step - loss: 0.0021 - accuracy: 0.99
99
[7 2 1 ... 4 5 6]
Validation score for learning rate  0.01  is 0.08221888542175293
Validation accuracy: 0.9818000197410583
Epoch 1/3
1563/1563 [==============================] - 2s 2ms/step - loss: 0.0023 - accuracy: 0.99
98
Epoch 2/3
1563/1563 [==============================] - 2s 1ms/step - loss: 0.0021 - accuracy: 0.99
99
Epoch 3/3
1563/1563 [==============================] - 2s 2ms/step - loss: 0.0019 - accuracy: 0.99
99
[7 2 1 ... 4 5 6]
Validation score for learning rate  0.05  is 0.08507594466209412
Validation accuracy: 0.982200026512146
Epoch 1/3
1563/1563 [==============================] - 2s 2ms/step - loss: 0.0021 - accuracy: 0.99
98
Epoch 2/3
1563/1563 [==============================] - 2s 1ms/step - loss: 0.0018 - accuracy: 0.99
99
Epoch 3/3
1563/1563 [==============================] - 2s 2ms/step - loss: 0.0015 - accuracy: 0.99
99
[7 2 1 ... 4 5 6]
Validation score for learning rate  0.1  is 0.08845045417547226
Validation accuracy: 0.9817000031471252
Epoch 1/3
1563/1563 [==============================] - 2s 2ms/step - loss: 0.0020 - accuracy: 0.99
99
Epoch 2/3
1563/1563 [==============================] - 2s 2ms/step - loss: 0.0102 - accuracy: 0.99
68
Epoch 3/3
1563/1563 [==============================] - 2s 1ms/step - loss: 0.0230 - accuracy: 0.99
23
[7 2 1 ... 4 5 6]
Validation score for learning rate  0.2  is 0.11874248087406158
Validation accuracy: 0.9754999876022339
Epoch 1/3
1563/1563 [==============================] - 2s 2ms/step - loss: 0.1048 - accuracy: 0.96
98
Epoch 2/3
1563/1563 [==============================] - 2s 1ms/step - loss: 0.0673 - accuracy: 0.97
96
Epoch 3/3
1563/1563 [==============================] - 2s 2ms/step - loss: 0.0607 - accuracy: 0.98
21
[7 2 1 ... 4 5 6]
Validation score for learning rate  0.4  is 0.11265822499990463
Validation accuracy: 0.9697999954223633
Epoch 1/3
1563/1563 [==============================] - 2s 2ms/step - loss: 1184.6309 - accuracy:
0.2290
Epoch 2/3
1563/1563 [==============================] - 2s 2ms/step - loss: 2.0690 - accuracy: 0.18
71
Epoch 3/3
1563/1563 [==============================] - 2s 1ms/step - loss: 2.0007 - accuracy: 0.19
08
[7 0 7 ... 7 0 0]
Validation score for learning rate  0.8  is 1.9619991779327393
Validation accuracy: 0.20640000700950623
```

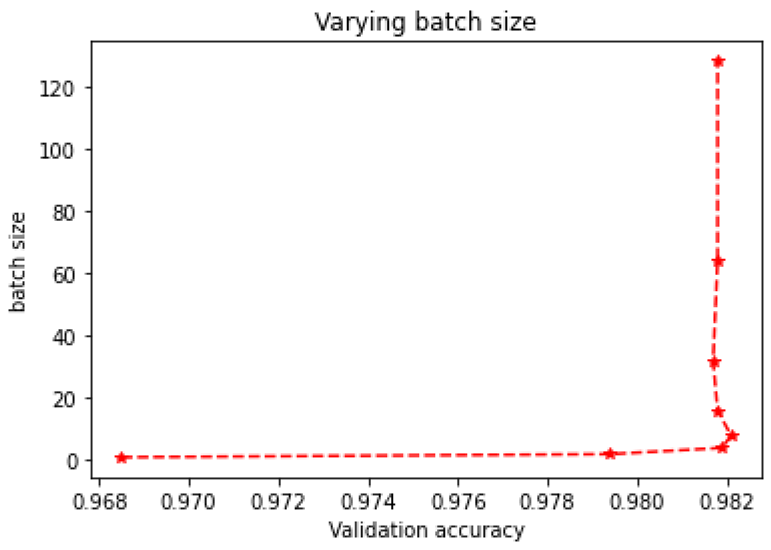
In [18]:
```python
# Varying hidden layer
# Fit model on training data
h_layer = [1, 2, 4, 6, 8]
hl_par = []
hl_val = []
for h in h_layer:
  model = Sequential()

  model.add(Dense( 100, input_shape=(784,)))
  model.add(Activation('relu'))
  for i in range(h):
    model.add(Dense(100))
    model.add(Activation('relu'))

  model.add(Dense(10))
  model.add(Activation('softmax'))
  sgd = SGD(lr=0.01) # Sets learning rate.
  model.compile(loss='categorical_crossentropy',
            optimizer=sgd,
            metrics=['accuracy'])
  model.fit(X_train, train_labels,batch_size=32, epochs=3, verbose=1)

  # Evaluate model on test data.
  score = model.evaluate(X_valid, valid_labels, verbose=0)
  # This returns only a score, so you will need to use another function for
```

```
    # extracting predicted labels for your confusion matrix. Use this line for that:
    classes = model.predict_classes(X_test, batch_size=32)
    print(classes)
    print('Validation score for hidden layer ', h, ' is', score[0])
    print('Validation accuracy:', score[1])
    hl_par.append(h)
    hl_val.append(score[1])
```

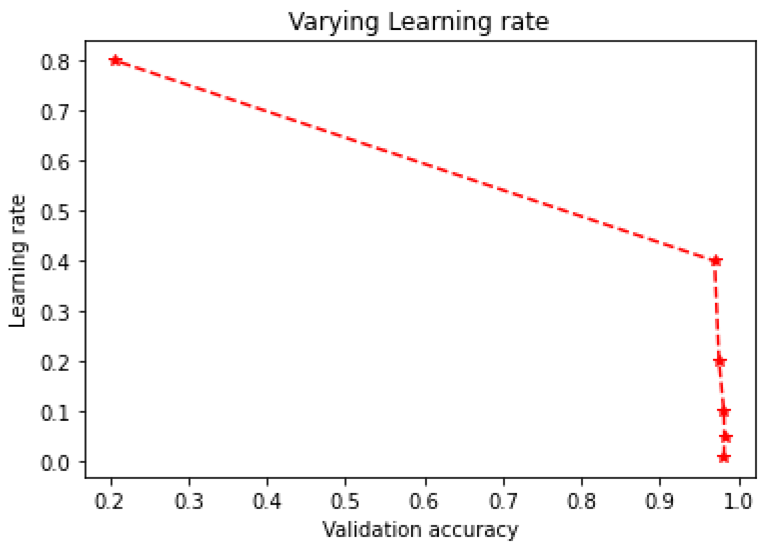
```
Epoch 1/3
1563/1563 [==============================] - 2s 2ms/step - loss: 0.7114 - accuracy: 0.81
23
Epoch 2/3
1563/1563 [==============================] - 2s 1ms/step - loss: 0.3211 - accuracy: 0.90
84
Epoch 3/3
1563/1563 [==============================] - 2s 1ms/step - loss: 0.2665 - accuracy: 0.92
33
[7 2 1 ... 4 5 6]
Validation score for hidden layer  1  is 0.23175029456615448
Validation accuracy: 0.9341999888420105
Epoch 1/3
1563/1563 [==============================] - 3s 2ms/step - loss: 0.8180 - accuracy: 0.76
76
Epoch 2/3
1563/1563 [==============================] - 3s 2ms/step - loss: 0.3034 - accuracy: 0.91
16
Epoch 3/3
1563/1563 [==============================] - 3s 2ms/step - loss: 0.2424 - accuracy: 0.92
99
[7 2 1 ... 4 5 6]
Validation score for hidden layer  2  is 0.1981097012758255
Validation accuracy: 0.944599986076355
Epoch 1/3
1563/1563 [==============================] - 3s 2ms/step - loss: 0.9169 - accuracy: 0.72
00
Epoch 2/3
1563/1563 [==============================] - 3s 2ms/step - loss: 0.2824 - accuracy: 0.91
71
Epoch 3/3
1563/1563 [==============================] - 3s 2ms/step - loss: 0.2079 - accuracy: 0.93
83
[7 2 1 ... 4 5 6]
Validation score for hidden layer  4  is 0.174785315990448
Validation accuracy: 0.9516000151634216
Epoch 1/3
1563/1563 [==============================] - 4s 2ms/step - loss: 1.1287 - accuracy: 0.61
56
Epoch 2/3
1563/1563 [==============================] - 3s 2ms/step - loss: 0.2925 - accuracy: 0.91
30
Epoch 3/3
1563/1563 [==============================] - 3s 2ms/step - loss: 0.1961 - accuracy: 0.94
25
[7 2 1 ... 4 5 6]
Validation score for hidden layer  6  is 0.18730050325393677
Validation accuracy: 0.9463000297546387
Epoch 1/3
1563/1563 [==============================] - 4s 2ms/step - loss: 1.5115 - accuracy: 0.49
83
Epoch 2/3
1563/1563 [==============================] - 4s 2ms/step - loss: 0.3505 - accuracy: 0.89
52
Epoch 3/3
1563/1563 [==============================] - 4s 2ms/step - loss: 0.2187 - accuracy: 0.93
39
```

```
[7 2 1 ... 4 5 6]
Validation score for hidden layer  8  is 0.2140929400920868
Validation accuracy: 0.9365000128746033
```

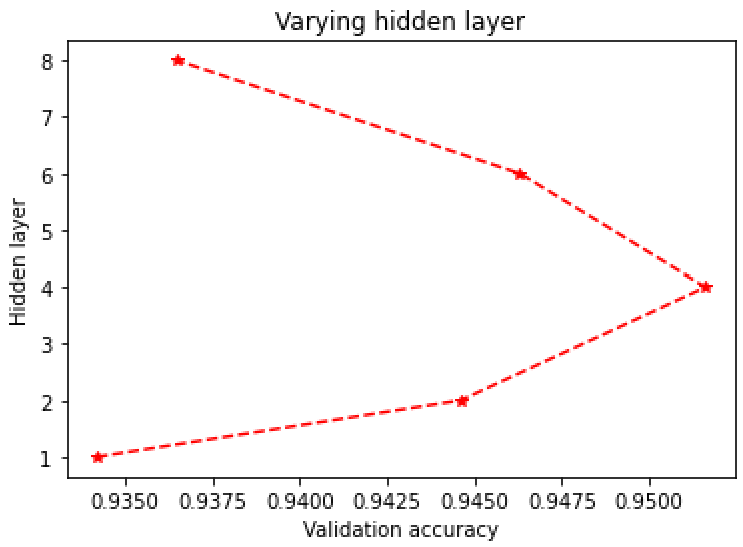
In [19]:
```
plt.plot(batch_val, batch_par, 'r*--')
#no need to write plt.show() as we have imported %matplotlib inline
plt.xlabel('Validation accuracy')
plt.ylabel('batch size')
plt.title('Varying batch size')
plt.savefig('batch-size.png')
```



In [20]:
```
plt.plot(lr_val, lr_par, 'r*--')
#no need to write plt.show() as we have imported %matplotlib inline
plt.xlabel('Validation accuracy')
plt.ylabel('Learning rate')
plt.title('Varying Learning rate')
plt.savefig('learning-rate.png')
```



In [21]:
```
plt.plot(hl_val, hl_par, 'r*--')
#no need to write plt.show() as we have imported %matplotlib inline
plt.xlabel('Validation accuracy')
plt.ylabel('Hidden layer')
plt.title('Varying hidden layer')
plt.savefig('num-hidden-layers.png')
```

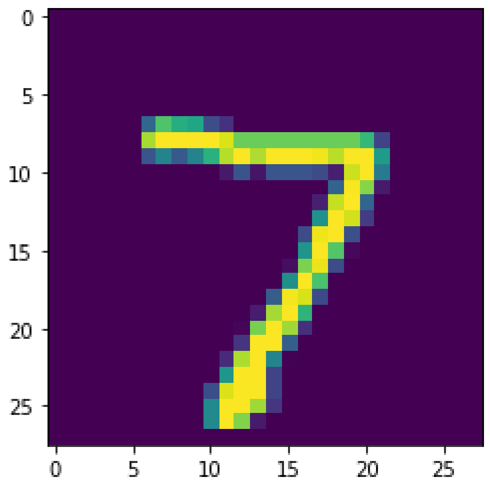


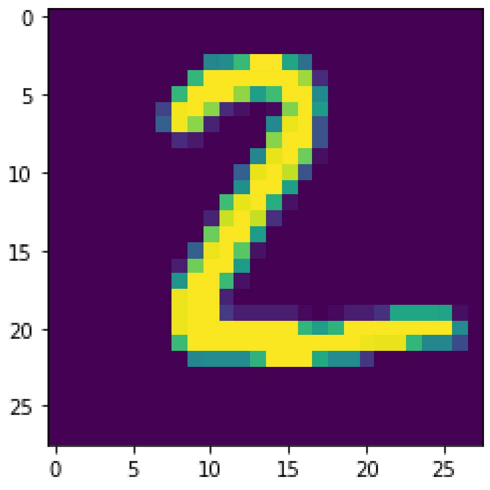
In [48]:
```
#Best Model based on the hyper parameter tuning

# Define model architecture
model = Sequential()

model.add(Dense( 100, input_shape=(784,)))
model.add(Activation('relu'))
for i in range(4):
  model.add(Dense(100))
  model.add(Activation('relu'))

model.add(Dense(10))
model.add(Activation('softmax'))

# Compile model
sgd = SGD(lr=0.05) # Sets learning rate.
model.compile(loss='categorical_crossentropy',
              optimizer=sgd,
              metrics=['accuracy'])

# Fit model on training data
model.fit(X_train, train_labels,batch_size=8, epochs=3, verbose=1)

# Evaluate model on test data.
score = model.evaluate(X_valid, valid_labels, verbose=0)
# This returns only a score, so you will need to use another function for
# extracting predicted labels for your confusion matrix. Use this line for that:
classes = model.predict_classes(X_test, batch_size=8)
print(classes)
print('Validation score:', score[0])
print('Validation accuracy:', score[1])
```

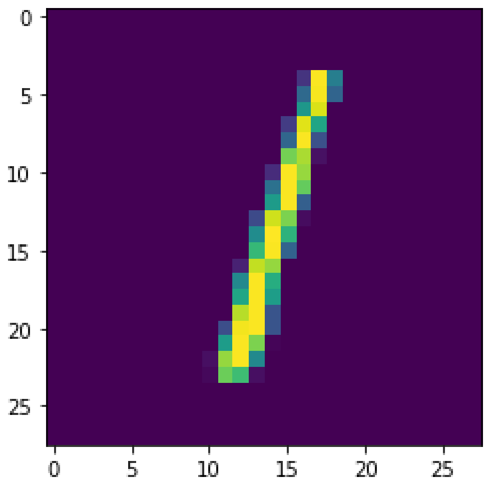
```
Epoch 1/3
6250/6250 [==============================] - 9s 1ms/step - loss: 0.3175 - accuracy: 0.90
34
Epoch 2/3
6250/6250 [==============================] - 8s 1ms/step - loss: 0.1429 - accuracy: 0.95
67
Epoch 3/3
6250/6250 [==============================] - 8s 1ms/step - loss: 0.1010 - accuracy: 0.96
96
[7 2 1 ... 4 5 6]
```

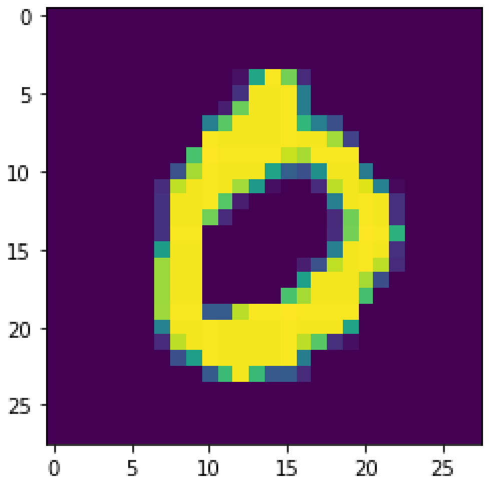
```
Validation score: 0.11261424422264099
Validation accuracy: 0.968500018119812
```

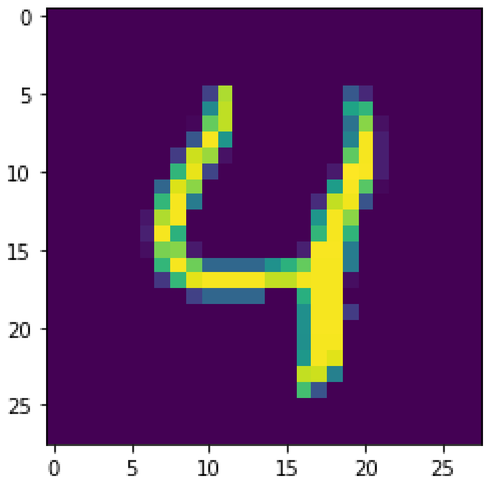
In [35]:
```
samples_to_predict = []
i = 0
# Generate plots for samples
for c in X_test:
  # Generate a plot
  reshaped_image = c.reshape((28, 28))
  plt.imshow(reshaped_image)
  plt.show()
  # Add sample to array for prediction
  samples_to_predict.append(c)
  i += 1
  if i > 10:
    break

# Convert into Numpy array
samples_to_predict = np.array(samples_to_predict)

# Generate predictions for samples
predictions = model.predict(samples_to_predict)
print(predictions)

# Generate arg maxes for predictions
classes = np.argmax(predictions, axis = 1)
print(classes)
```

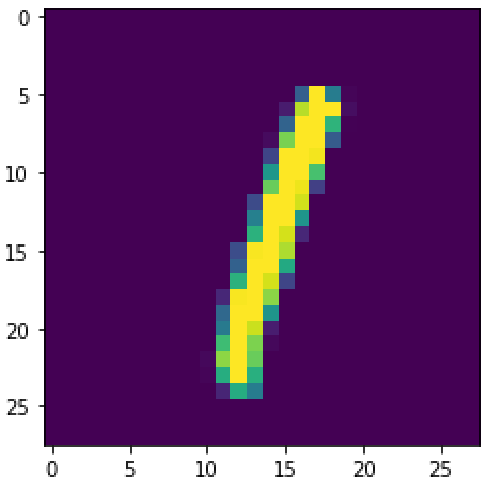


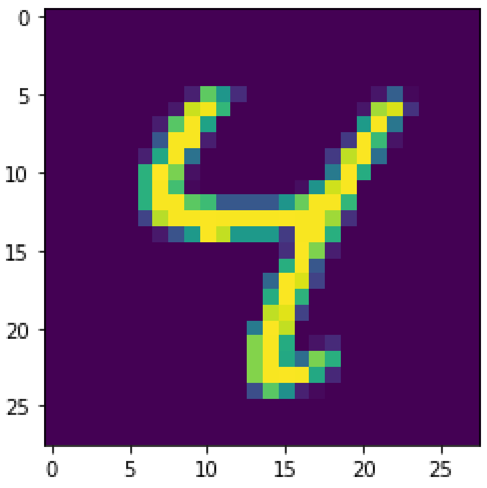


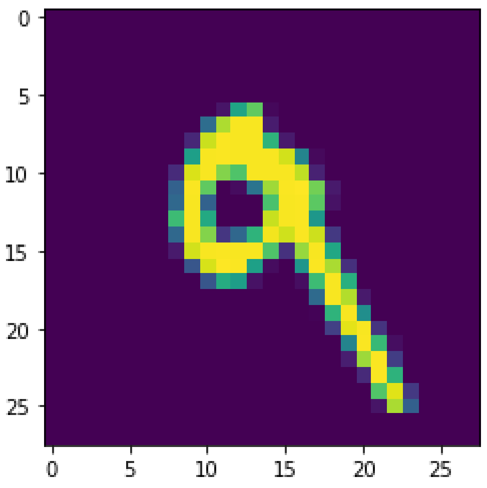

0
5
10
15
20
25
0
5
10
15
20
25

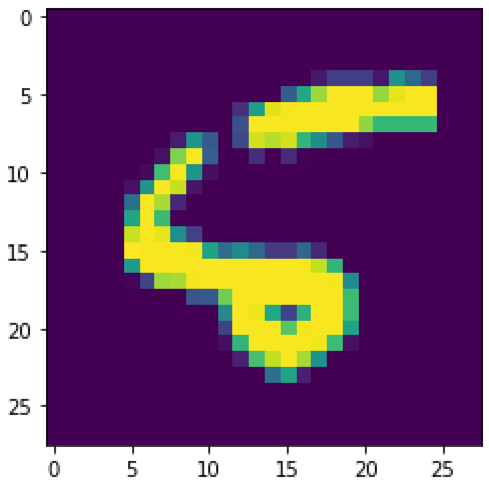

0
5
10
15
20
25
0
5
10
15
20
25

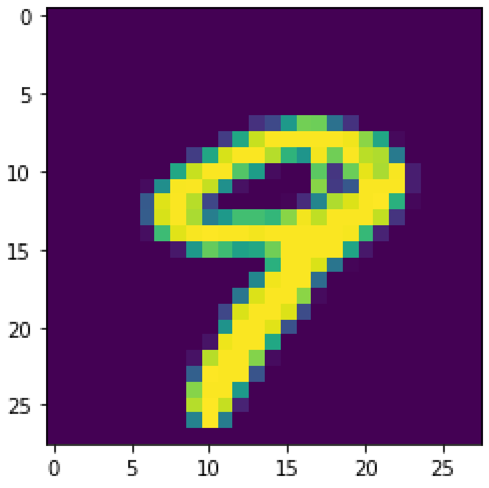

0
5
10
15
20
25
0
5
10
15
20
25

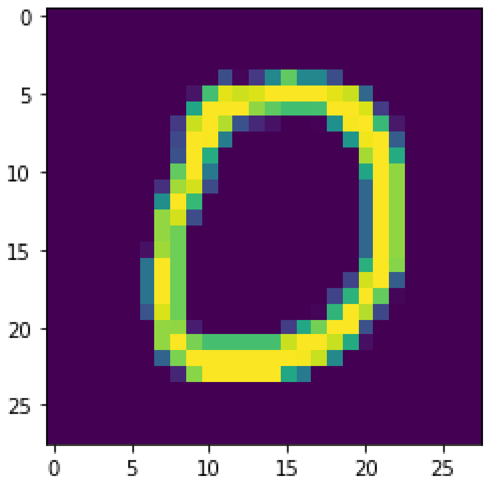

0
5
10
15
20
25
0
5
10
15
20
25


0
5
10
15
20
25
0
5
10
15
20
25


0
5
10
15
20
25
0
5
10
15
20
25







```
[[2.51149430e-07 2.88021556e-06 1.88689392e-05 6.01972715e-06
  4.10287839e-06 4.36071552e-08 1.35157663e-09 9.99934316e-01
  6.06108230e-08 3.33358294e-05]
 [4.80976014e-05 1.01113948e-03 9.78432655e-01 1.75697636e-02
  1.04152015e-04 1.44277597e-04 1.29556007e-04 1.36742671e-03
  1.18343160e-03 9.55329324e-06]
 [4.00893259e-06 9.92076159e-01 1.88560865e-04 7.81863200e-05
  2.64793896e-04 1.21812973e-05 8.38867436e-06 4.27777367e-03
  3.03596561e-03 5.40765832e-05]
 [9.99994278e-01 2.28860972e-10 1.71267618e-06 8.50340314e-08
  3.94639727e-07 2.02443204e-07 2.73165438e-06 6.14863751e-08
  1.85441493e-07 4.56603658e-07]
 [3.74472074e-06 9.53401809e-07 5.62553250e-05 5.67868540e-07
  9.99830604e-01 9.81082167e-07 2.11566330e-05 4.14005226e-05
  2.42110991e-06 4.19976168e-05]
 [8.77405171e-08 9.97668326e-01 1.05785111e-05 2.32107527e-06
  3.28376664e-05 1.77582663e-07 1.09445047e-07 2.03810912e-03
  2.42813432e-04 4.70148871e-06]
 [2.95415180e-06 3.23752356e-05 2.48313503e-04 3.60959211e-06
  9.96362507e-01 4.59834519e-06 2.79956748e-05 2.12428626e-04
  3.01829400e-03 8.69827127e-05]
 [5.98612707e-04 1.12188410e-03 8.12144717e-04 1.47582861e-02
  1.42551335e-02 1.44632934e-02 1.13273272e-04 7.39921164e-03
  5.28196292e-03 9.41196084e-01]
 [2.96892389e-03 7.23024074e-04 3.22803477e-04 2.06703693e-03
  2.38540629e-03 3.50639313e-01 4.11823362e-01 3.27797927e-04
  2.27619410e-01 1.12297945e-03]
 [5.96970040e-06 6.17955629e-06 3.07455484e-05 7.81852941e-05
  5.12411911e-03 2.29606248e-05 5.70456621e-07 2.07056932e-04
  2.95834034e-04 9.94228363e-01]
 [9.99975562e-01 1.56914470e-09 3.84007399e-06 8.44639416e-08
  3.22140363e-06 7.23581252e-07 6.67030872e-06 8.22835830e-07
  1.18235334e-07 8.92977550e-06]]
[7 2 1 0 4 1 4 9 6 9 0]
```

TASK 1 observation in text file

1. Write a brief description of the variation observed in each graph and your hypothesis explaining the variation in your own words.

The number of examples from the training dataset used in the estimate of the error gradient is called the batch size and is an important hyperparameter that influences the dynamics of the learning algorithm. Batch size controls the accuracy of the estimate of the error gradient when training neural networks.There is a tension between batch size and the speed and stability of the learning process.Normally, a smaller batch sizes is preferred because smaller batch sizes are noisy, offering a regularizing effect and lower generalization error. Also, smaller batch sizes make it easier to fit one batch worth of training data in memory (i.e. when using a GPU).In my case, th accuracy of the model increased from 1 to 8, and from 16 to 128 it stayed more or less the same. So, accuracy is best at batch size = 8.

One of the important things is the selection of the learning rate. If the learning rate is too high, the loss may not converge, and if it is too low, the training will be slow. So it is important to select the reasonably fair value of learning rate. One of the good value to start with is 0.01. If it doesn't work, then other higher or lower values can be tried. In my case, from 0.01 to 0.05 the increase in learning rate increases the accuracy. But from 0.1 to 0.8, the learning rate decreased gradually. So, accuracy is best at learning rate = 0.05.

Increasing the number of hidden layers might improve the accuracy or might not, Increasing the number of hidden layers much more than the sufficient number of layers will cause accuracy in the test set to decrease. It will cause your network to overfit to the training set, that is, it will learn the training data, but it won't be able to generalize to new unseen data. In my case, the accuracy increased from 1 to 4, and from 6 to 8, the accuracy reduced. So, accuracy is best at no. of hidden layers = 4.

1. Should accuracy alone be the criterion deciding a parameter setting? What could be other considerations in practice?

Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right. Accuracy alone doesn't tell the full story when you're working with a class-imbalanced data set, where there is a significant disparity between the number of positive and negative labels. To fully evaluate the effectiveness of a model, you must examine both precision and recall. Unfortunately, precision and recall are often in tension. That is, improving precision typically reduces recall and vice versa. TO solve this, F-score is used. The F1 score is the harmonic mean of the precision and recall.

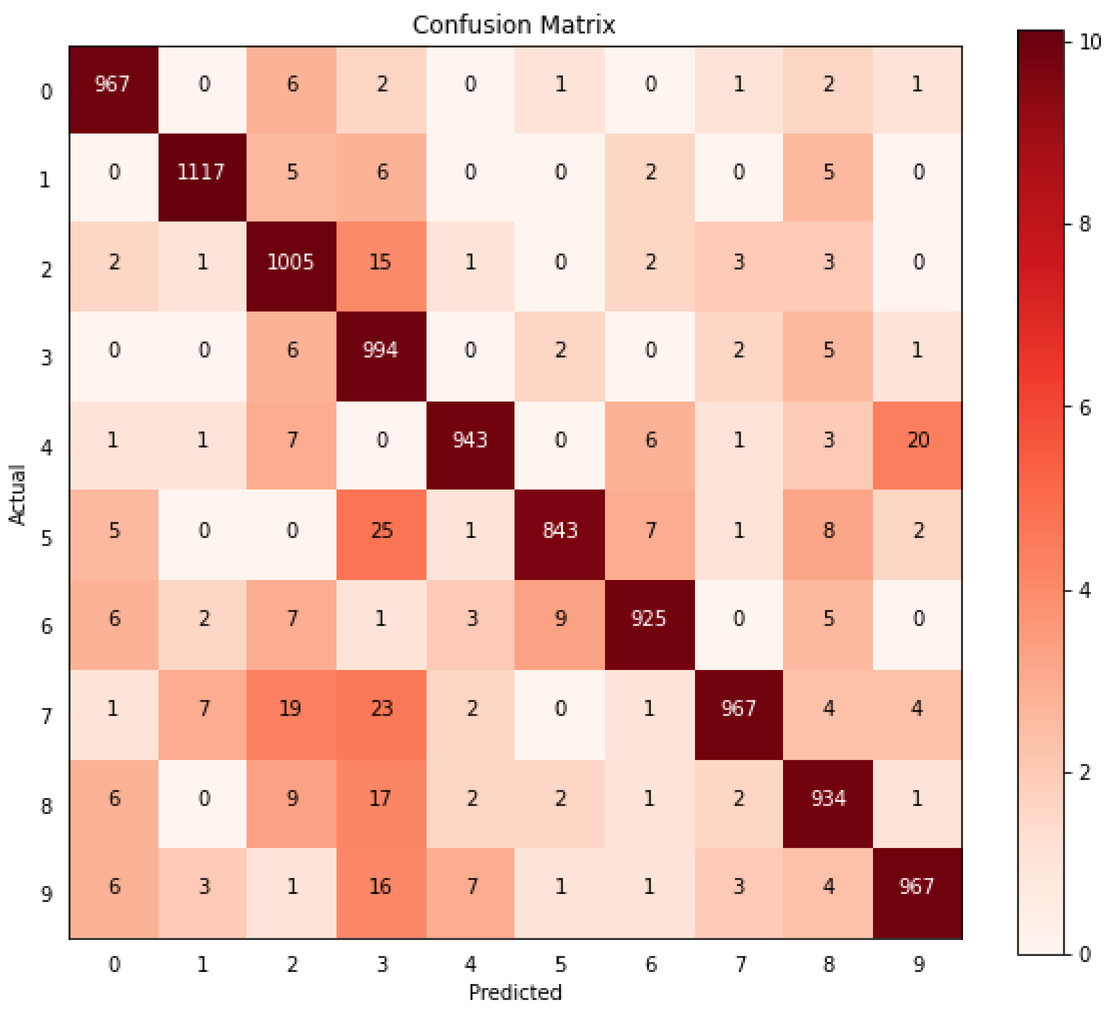
Task 2: Confusion matrix

For this exercise, set the hyperparameters of the network to the values that resulted in the maximum validation accuracy in Task 1. Train the model and evaluate the performance of your model on the validation set. Now use the predictions of your model and the true labels of the validation set to create the 10x10 confusion matrix.

In [61]:
```
val_p = classes
error = 0
confusion_matrix = np.zeros([10,10])
for i in range(X_test.shape[0]):
    confusion_matrix[test_labels_1[i],val_p[i]] += 1
    if test_labels_1[i]!=val_p[i]:
        error +=1
print("Confusion Matrix: \n\n" ,confusion_matrix)
print("\nErrors in validation set: " ,error)
print("\nError Persentage : " ,(error*100)/val_p.shape[0])
print("\nAccuracy : " ,100-(error*100)/val_p.shape[0])
print("\nValidation set Shape :",val_p.shape[0])
```

```
Confusion Matrix:

 [[9.670e+02 0.000e+00 6.000e+00 2.000e+00 0.000e+00 1.000e+00 0.000e+00
  1.000e+00 2.000e+00 1.000e+00]
 [0.000e+00 1.117e+03 5.000e+00 6.000e+00 0.000e+00 0.000e+00 2.000e+00
  0.000e+00 5.000e+00 0.000e+00]
 [2.000e+00 1.000e+00 1.005e+03 1.500e+01 1.000e+00 0.000e+00 2.000e+00
  3.000e+00 3.000e+00 0.000e+00]
 [0.000e+00 0.000e+00 6.000e+00 9.940e+02 0.000e+00 2.000e+00 0.000e+00
  2.000e+00 5.000e+00 1.000e+00]
 [1.000e+00 1.000e+00 7.000e+00 0.000e+00 9.430e+02 0.000e+00 6.000e+00
  1.000e+00 3.000e+00 2.000e+01]
 [5.000e+00 0.000e+00 0.000e+00 2.500e+01 1.000e+00 8.430e+02 7.000e+00
  1.000e+00 8.000e+00 2.000e+00]
 [6.000e+00 2.000e+00 7.000e+00 1.000e+00 3.000e+00 9.000e+00 9.250e+02
  0.000e+00 5.000e+00 0.000e+00]
 [1.000e+00 7.000e+00 1.900e+01 2.300e+01 2.000e+00 0.000e+00 1.000e+00
  9.670e+02 4.000e+00 4.000e+00]
 [6.000e+00 0.000e+00 9.000e+00 1.700e+01 2.000e+00 2.000e+00 1.000e+00
  2.000e+00 9.340e+02 1.000e+00]
 [6.000e+00 3.000e+00 1.000e+00 1.600e+01 7.000e+00 1.000e+00 1.000e+00
  3.000e+00 4.000e+00 9.670e+02]]

Errors in validation set:  338

Error Persentage :  3.38

Accuracy :  96.62

Validation set Shape : 10000
```

In [62]:
```
f = plt.figure(figsize=(10,8.5))
f.add_subplot(111)

plt.imshow(np.log2(confusion_matrix+1),cmap="Reds")
plt.colorbar()
plt.tick_params(size=5,color="white")
plt.xticks(np.arange(0,10),np.arange(0,10))
plt.yticks(np.arange(0,10),np.arange(0,10))

threshold = confusion_matrix.max()/2

for i in range(10):
    for j in range(10):
        plt.text(j,i,int(confusion_matrix[i,j]),horizontalalignment="center",color="whi

plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.savefig("Confusion_matrix1.png")
plt.show()
```



Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	967	0	6	2	0	1	0	1	2	1
1	0	1117	5	6	0	0	2	0	5	0
2	2	1	1005	15	1	0	2	3	3	0
3	0	0	6	994	0	2	0	2	5	1
4	1	1	7	0	943	0	6	1	3	20
5	5	0	0	25	1	843	7	1	8	2
6	6	2	7	1	3	9	925	0	5	0
7	1	7	19	23	2	0	1	967	4	4
8	6	0	9	17	2	2	1	2	934	1
9	6	3	1	16	7	1	1	3	4	967

Between which two classes does your model get the most confused? Which one of those is the true label and which one is the prediction? What would you do if you wanted to make fewer misclassifications of one particular class?

My model got most confused between 5 and 2, showing a error of 25 wrong classifications. 5 is the actual class and 2 is the predicted class. One way to improve the performance of a model that does the classification is to train the model with more data. Because basically, misclassification means the features of this particular sample haven't been learnt by this model. If we don't have more data to re-train your model, think about those helpful techniques, such as "data augmentation" and "Transfer Learning". They work pretty well with small datasets. Another way is to just balance the samples in each class. And if we want to increase the accuracy just take very small value for initial learning rate while defining options parameters.

Task 3: Overfitting

Find out about overfitting by training the neural network on just the first 1000 of the 50,000 training examples, for 500 iterations.

In [65]:
```
#Best Model based on the hyper parameter tuning

# Define model architecture
model = Sequential()

model.add(Dense( 100, input_shape=(784,)))
model.add(Activation('relu'))
for i in range(4):
  model.add(Dense(100))
  model.add(Activation('relu'))

model.add(Dense(10))
model.add(Activation('softmax'))

# Compile model
sgd = SGD(lr=0.05) # Sets learning rate.
model.compile(loss='categorical_crossentropy',
              optimizer=sgd,
              metrics=['accuracy'])

X_train_new = X_train_1[:1000]
train_labels_new = train_labels[:1000]
# Fit model on training data
model.fit(X_train_new, train_labels_new,batch_size=8, epochs=500, verbose=1)
```

```
Epoch 1/500
125/125 [==============================] - 0s 1ms/step - loss: 1.8512 - accuracy: 0.3810
Epoch 2/500
125/125 [==============================] - 0s 1ms/step - loss: 0.8215 - accuracy: 0.7290
Epoch 3/500
125/125 [==============================] - 0s 1ms/step - loss: 0.5068 - accuracy: 0.8470
Epoch 4/500
125/125 [==============================] - 0s 1ms/step - loss: 0.3829 - accuracy: 0.8890
Epoch 5/500
125/125 [==============================] - 0s 1ms/step - loss: 0.2609 - accuracy: 0.9160
Epoch 6/500
125/125 [==============================] - 0s 1ms/step - loss: 0.1997 - accuracy: 0.9390
Epoch 7/500
125/125 [==============================] - 0s 1ms/step - loss: 0.0986 - accuracy: 0.9710
Epoch 8/500
125/125 [==============================] - 0s 1ms/step - loss: 0.0815 - accuracy: 0.9780
Epoch 9/500
125/125 [==============================] - 0s 1ms/step - loss: 0.0365 - accuracy: 0.9900
Epoch 10/500
```

```
125/125 [==============================] - 0s 1ms/step - loss: 0.0700 - accuracy: 0.9760
Epoch 11/500
125/125 [==============================] - 0s 1ms/step - loss: 0.0545 - accuracy: 0.9850
Epoch 12/500
125/125 [==============================] - 0s 1ms/step - loss: 0.0221 - accuracy: 0.9940
Epoch 13/500
125/125 [==============================] - 0s 1ms/step - loss: 0.0053 - accuracy: 0.9990
Epoch 14/500
125/125 [==============================] - 0s 1ms/step - loss: 0.0022 - accuracy: 1.0000
Epoch 15/500
125/125 [==============================] - 0s 1ms/step - loss: 0.0013 - accuracy: 1.0000
Epoch 16/500
125/125 [==============================] - 0s 1ms/step - loss: 9.7562e-04 - accuracy: 1.
0000
Epoch 17/500
125/125 [==============================] - 0s 1ms/step - loss: 8.0377e-04 - accuracy: 1.
0000
Epoch 18/500
125/125 [==============================] - 0s 1ms/step - loss: 6.8727e-04 - accuracy: 1.
0000
Epoch 19/500
125/125 [==============================] - 0s 1ms/step - loss: 5.9649e-04 - accuracy: 1.
0000
Epoch 20/500
125/125 [==============================] - 0s 1ms/step - loss: 5.3134e-04 - accuracy: 1.
0000
Epoch 21/500
125/125 [==============================] - 0s 1ms/step - loss: 4.7762e-04 - accuracy: 1.
0000
Epoch 22/500
125/125 [==============================] - 0s 1ms/step - loss: 4.3343e-04 - accuracy: 1.
0000
Epoch 23/500
125/125 [==============================] - 0s 1ms/step - loss: 3.9723e-04 - accuracy: 1.
0000
Epoch 24/500
125/125 [==============================] - 0s 1ms/step - loss: 3.6617e-04 - accuracy: 1.
0000
Epoch 25/500
125/125 [==============================] - 0s 1ms/step - loss: 3.3983e-04 - accuracy: 1.
0000
Epoch 26/500
125/125 [==============================] - 0s 1ms/step - loss: 3.1651e-04 - accuracy: 1.
0000
Epoch 27/500
125/125 [==============================] - 0s 2ms/step - loss: 2.9643e-04 - accuracy: 1.
0000
Epoch 28/500
125/125 [==============================] - 0s 1ms/step - loss: 2.7902e-04 - accuracy: 1.
0000
Epoch 29/500
125/125 [==============================] - 0s 1ms/step - loss: 2.6353e-04 - accuracy: 1.
0000
Epoch 30/500
125/125 [==============================] - 0s 1ms/step - loss: 2.4914e-04 - accuracy: 1.
0000
Epoch 31/500
125/125 [==============================] - 0s 1ms/step - loss: 2.3664e-04 - accuracy: 1.
0000
Epoch 32/500
125/125 [==============================] - 0s 1ms/step - loss: 2.2511e-04 - accuracy: 1.
0000
Epoch 33/500
125/125 [==============================] - 0s 1ms/step - loss: 2.1479e-04 - accuracy: 1.
0000
```

```
Epoch 34/500
125/125 [==============================] - 0s 1ms/step - loss: 2.0510e-04 - accuracy: 1.
0000
Epoch 35/500
125/125 [==============================] - 0s 1ms/step - loss: 1.9637e-04 - accuracy: 1.
0000
Epoch 36/500
125/125 [==============================] - 0s 1ms/step - loss: 1.8826e-04 - accuracy: 1.
0000
Epoch 37/500
125/125 [==============================] - 0s 1ms/step - loss: 1.8094e-04 - accuracy: 1.
0000
Epoch 38/500
125/125 [==============================] - 0s 1ms/step - loss: 1.7408e-04 - accuracy: 1.
0000
Epoch 39/500
125/125 [==============================] - 0s 1ms/step - loss: 1.6779e-04 - accuracy: 1.
0000
Epoch 40/500
125/125 [==============================] - 0s 1ms/step - loss: 1.6178e-04 - accuracy: 1.
0000
Epoch 41/500
125/125 [==============================] - 0s 1ms/step - loss: 1.5618e-04 - accuracy: 1.
0000
Epoch 42/500
125/125 [==============================] - 0s 1ms/step - loss: 1.5092e-04 - accuracy: 1.
0000
Epoch 43/500
125/125 [==============================] - 0s 2ms/step - loss: 1.4612e-04 - accuracy: 1.
0000
Epoch 44/500
125/125 [==============================] - 0s 1ms/step - loss: 1.4156e-04 - accuracy: 1.
0000
Epoch 45/500
125/125 [==============================] - 0s 1ms/step - loss: 1.3702e-04 - accuracy: 1.
0000
Epoch 46/500
125/125 [==============================] - 0s 1ms/step - loss: 1.3299e-04 - accuracy: 1.
0000
Epoch 47/500
125/125 [==============================] - 0s 1ms/step - loss: 1.2919e-04 - accuracy: 1.
0000
Epoch 48/500
125/125 [==============================] - 0s 1ms/step - loss: 1.2542e-04 - accuracy: 1.
0000
Epoch 49/500
125/125 [==============================] - 0s 1ms/step - loss: 1.2202e-04 - accuracy: 1.
0000
Epoch 50/500
125/125 [==============================] - 0s 1ms/step - loss: 1.1875e-04 - accuracy: 1.
0000
Epoch 51/500
125/125 [==============================] - 0s 1ms/step - loss: 1.1562e-04 - accuracy: 1.
0000
Epoch 52/500
125/125 [==============================] - 0s 1ms/step - loss: 1.1262e-04 - accuracy: 1.
0000
Epoch 53/500
125/125 [==============================] - 0s 1ms/step - loss: 1.0989e-04 - accuracy: 1.
0000
Epoch 54/500
125/125 [==============================] - 0s 1ms/step - loss: 1.0717e-04 - accuracy: 1.
0000
Epoch 55/500
125/125 [==============================] - 0s 1ms/step - loss: 1.0455e-04 - accuracy: 1.
```

```
0000
Epoch 56/500
125/125 [==============================] - 0s 1ms/step - loss: 1.0217e-04 - accuracy: 1.
0000
Epoch 57/500
125/125 [==============================] - 0s 1ms/step - loss: 9.9809e-05 - accuracy: 1.
0000
Epoch 58/500
125/125 [==============================] - 0s 1ms/step - loss: 9.7594e-05 - accuracy: 1.
0000
Epoch 59/500
125/125 [==============================] - 0s 1ms/step - loss: 9.5410e-05 - accuracy: 1.
0000
Epoch 60/500
125/125 [==============================] - 0s 1ms/step - loss: 9.3330e-05 - accuracy: 1.
0000
Epoch 61/500
125/125 [==============================] - 0s 1ms/step - loss: 9.1355e-05 - accuracy: 1.
0000
Epoch 62/500
125/125 [==============================] - 0s 1ms/step - loss: 8.9511e-05 - accuracy: 1.
0000
Epoch 63/500
125/125 [==============================] - 0s 1ms/step - loss: 8.7720e-05 - accuracy: 1.
0000
Epoch 64/500
125/125 [==============================] - 0s 1ms/step - loss: 8.5920e-05 - accuracy: 1.
0000
Epoch 65/500
125/125 [==============================] - 0s 1ms/step - loss: 8.4251e-05 - accuracy: 1.
0000
Epoch 66/500
125/125 [==============================] - 0s 1ms/step - loss: 8.2660e-05 - accuracy: 1.
0000
Epoch 67/500
125/125 [==============================] - 0s 1ms/step - loss: 8.1068e-05 - accuracy: 1.
0000
Epoch 68/500
125/125 [==============================] - 0s 1ms/step - loss: 7.9588e-05 - accuracy: 1.
0000
Epoch 69/500
125/125 [==============================] - 0s 1ms/step - loss: 7.8162e-05 - accuracy: 1.
0000
Epoch 70/500
125/125 [==============================] - 0s 1ms/step - loss: 7.6733e-05 - accuracy: 1.
0000
Epoch 71/500
125/125 [==============================] - 0s 1ms/step - loss: 7.5423e-05 - accuracy: 1.
0000
Epoch 72/500
125/125 [==============================] - 0s 1ms/step - loss: 7.4073e-05 - accuracy: 1.
0000
Epoch 73/500
125/125 [==============================] - 0s 1ms/step - loss: 7.2819e-05 - accuracy: 1.
0000
Epoch 74/500
125/125 [==============================] - 0s 2ms/step - loss: 7.1600e-05 - accuracy: 1.
0000
Epoch 75/500
125/125 [==============================] - 0s 1ms/step - loss: 7.0374e-05 - accuracy: 1.
0000
Epoch 76/500
125/125 [==============================] - 0s 1ms/step - loss: 6.9213e-05 - accuracy: 1.
0000
Epoch 77/500
```

```
125/125 [==============================] - 0s 1ms/step - loss: 6.8120e-05 - accuracy: 1.
0000
Epoch 78/500
125/125 [==============================] - 0s 1ms/step - loss: 6.7029e-05 - accuracy: 1.
0000
Epoch 79/500
125/125 [==============================] - 0s 1ms/step - loss: 6.5977e-05 - accuracy: 1.
0000
Epoch 80/500
125/125 [==============================] - 0s 1ms/step - loss: 6.4939e-05 - accuracy: 1.
0000
Epoch 81/500
125/125 [==============================] - 0s 1ms/step - loss: 6.3962e-05 - accuracy: 1.
0000
Epoch 82/500
125/125 [==============================] - 0s 1ms/step - loss: 6.3002e-05 - accuracy: 1.
0000
Epoch 83/500
125/125 [==============================] - 0s 2ms/step - loss: 6.2031e-05 - accuracy: 1.
0000
Epoch 84/500
125/125 [==============================] - 0s 1ms/step - loss: 6.1149e-05 - accuracy: 1.
0000
Epoch 85/500
125/125 [==============================] - 0s 1ms/step - loss: 6.0241e-05 - accuracy: 1.
0000
Epoch 86/500
125/125 [==============================] - 0s 1ms/step - loss: 5.9387e-05 - accuracy: 1.
0000
Epoch 87/500
125/125 [==============================] - 0s 1ms/step - loss: 5.8562e-05 - accuracy: 1.
0000
Epoch 88/500
125/125 [==============================] - 0s 1ms/step - loss: 5.7745e-05 - accuracy: 1.
0000
Epoch 89/500
125/125 [==============================] - 0s 1ms/step - loss: 5.6952e-05 - accuracy: 1.
0000
Epoch 90/500
125/125 [==============================] - 0s 1ms/step - loss: 5.6174e-05 - accuracy: 1.
0000
Epoch 91/500
125/125 [==============================] - 0s 1ms/step - loss: 5.5432e-05 - accuracy: 1.
0000
Epoch 92/500
125/125 [==============================] - 0s 1ms/step - loss: 5.4707e-05 - accuracy: 1.
0000
Epoch 93/500
125/125 [==============================] - 0s 1ms/step - loss: 5.3974e-05 - accuracy: 1.
0000
Epoch 94/500
125/125 [==============================] - 0s 1ms/step - loss: 5.3291e-05 - accuracy: 1.
0000
Epoch 95/500
125/125 [==============================] - 0s 2ms/step - loss: 5.2601e-05 - accuracy: 1.
0000
Epoch 96/500
125/125 [==============================] - 0s 1ms/step - loss: 5.1943e-05 - accuracy: 1.
0000
Epoch 97/500
125/125 [==============================] - 0s 1ms/step - loss: 5.1299e-05 - accuracy: 1.
0000
Epoch 98/500
125/125 [==============================] - 0s 1ms/step - loss: 5.0666e-05 - accuracy: 1.
0000
```

```
Epoch 99/500
125/125 [==============================] - 0s 1ms/step - loss: 5.0043e-05 - accuracy: 1.
0000
Epoch 100/500
125/125 [==============================] - 0s 1ms/step - loss: 4.9429e-05 - accuracy: 1.
0000
Epoch 101/500
125/125 [==============================] - 0s 1ms/step - loss: 4.8849e-05 - accuracy: 1.
0000
Epoch 102/500
125/125 [==============================] - 0s 1ms/step - loss: 4.8261e-05 - accuracy: 1.
0000
Epoch 103/500
125/125 [==============================] - 0s 1ms/step - loss: 4.7700e-05 - accuracy: 1.
0000
Epoch 104/500
125/125 [==============================] - 0s 1ms/step - loss: 4.7167e-05 - accuracy: 1.
0000
Epoch 105/500
125/125 [==============================] - 0s 2ms/step - loss: 4.6624e-05 - accuracy: 1.
0000
Epoch 106/500
125/125 [==============================] - 0s 2ms/step - loss: 4.6094e-05 - accuracy: 1.
0000
Epoch 107/500
125/125 [==============================] - 0s 1ms/step - loss: 4.5587e-05 - accuracy: 1.
0000
Epoch 108/500
125/125 [==============================] - 0s 1ms/step - loss: 4.5065e-05 - accuracy: 1.
0000
Epoch 109/500
125/125 [==============================] - 0s 1ms/step - loss: 4.4599e-05 - accuracy: 1.
0000
Epoch 110/500
125/125 [==============================] - 0s 1ms/step - loss: 4.4110e-05 - accuracy: 1.
0000
Epoch 111/500
125/125 [==============================] - 0s 1ms/step - loss: 4.3642e-05 - accuracy: 1.
0000
Epoch 112/500
125/125 [==============================] - 0s 1ms/step - loss: 4.3173e-05 - accuracy: 1.
0000
Epoch 113/500
125/125 [==============================] - 0s 1ms/step - loss: 4.2716e-05 - accuracy: 1.
0000
Epoch 114/500
125/125 [==============================] - 0s 1ms/step - loss: 4.2280e-05 - accuracy: 1.
0000
Epoch 115/500
125/125 [==============================] - 0s 1ms/step - loss: 4.1831e-05 - accuracy: 1.
0000
Epoch 116/500
125/125 [==============================] - 0s 1ms/step - loss: 4.1414e-05 - accuracy: 1.
0000
Epoch 117/500
125/125 [==============================] - 0s 1ms/step - loss: 4.1003e-05 - accuracy: 1.
0000
Epoch 118/500
125/125 [==============================] - 0s 2ms/step - loss: 4.0582e-05 - accuracy: 1.
0000
Epoch 119/500
125/125 [==============================] - 0s 1ms/step - loss: 4.0184e-05 - accuracy: 1.
0000
Epoch 120/500
125/125 [==============================] - 0s 1ms/step - loss: 3.9794e-05 - accuracy: 1.
```

```
0000
Epoch 121/500
125/125 [==============================] - 0s 1ms/step - loss: 3.9391e-05 - accuracy: 1.
0000
Epoch 122/500
125/125 [==============================] - 0s 1ms/step - loss: 3.9026e-05 - accuracy: 1.
0000
Epoch 123/500
125/125 [==============================] - 0s 1ms/step - loss: 3.8646e-05 - accuracy: 1.
0000
Epoch 124/500
125/125 [==============================] - 0s 1ms/step - loss: 3.8278e-05 - accuracy: 1.
0000
Epoch 125/500
125/125 [==============================] - 0s 1ms/step - loss: 3.7929e-05 - accuracy: 1.
0000
Epoch 126/500
125/125 [==============================] - 0s 1ms/step - loss: 3.7564e-05 - accuracy: 1.
0000
Epoch 127/500
125/125 [==============================] - 0s 1ms/step - loss: 3.7223e-05 - accuracy: 1.
0000
Epoch 128/500
125/125 [==============================] - 0s 1ms/step - loss: 3.6881e-05 - accuracy: 1.
0000
Epoch 129/500
125/125 [==============================] - 0s 1ms/step - loss: 3.6548e-05 - accuracy: 1.
0000
Epoch 130/500
125/125 [==============================] - 0s 1ms/step - loss: 3.6220e-05 - accuracy: 1.
0000
Epoch 131/500
125/125 [==============================] - 0s 1ms/step - loss: 3.5888e-05 - accuracy: 1.
0000
Epoch 132/500
125/125 [==============================] - 0s 1ms/step - loss: 3.5575e-05 - accuracy: 1.
0000
Epoch 133/500
125/125 [==============================] - 0s 1ms/step - loss: 3.5262e-05 - accuracy: 1.
0000
Epoch 134/500
125/125 [==============================] - 0s 1ms/step - loss: 3.4955e-05 - accuracy: 1.
0000
Epoch 135/500
125/125 [==============================] - 0s 2ms/step - loss: 3.4651e-05 - accuracy: 1.
0000
Epoch 136/500
125/125 [==============================] - 0s 1ms/step - loss: 3.4360e-05 - accuracy: 1.
0000
Epoch 137/500
125/125 [==============================] - 0s 1ms/step - loss: 3.4060e-05 - accuracy: 1.
0000
Epoch 138/500
125/125 [==============================] - 0s 1ms/step - loss: 3.3776e-05 - accuracy: 1.
0000
Epoch 139/500
125/125 [==============================] - 0s 2ms/step - loss: 3.3489e-05 - accuracy: 1.
0000
Epoch 140/500
125/125 [==============================] - 0s 1ms/step - loss: 3.3214e-05 - accuracy: 1.
0000
Epoch 141/500
125/125 [==============================] - 0s 2ms/step - loss: 3.2940e-05 - accuracy: 1.
0000
Epoch 142/500
```

```
125/125 [==============================] - 0s 1ms/step - loss: 3.2667e-05 - accuracy: 1.
0000
Epoch 143/500
125/125 [==============================] - 0s 1ms/step - loss: 3.2402e-05 - accuracy: 1.
0000
Epoch 144/500
125/125 [==============================] - 0s 1ms/step - loss: 3.2147e-05 - accuracy: 1.
0000
Epoch 145/500
125/125 [==============================] - 0s 2ms/step - loss: 3.1888e-05 - accuracy: 1.
0000
Epoch 146/500
125/125 [==============================] - 0s 1ms/step - loss: 3.1625e-05 - accuracy: 1.
0000
Epoch 147/500
125/125 [==============================] - 0s 1ms/step - loss: 3.1380e-05 - accuracy: 1.
0000
Epoch 148/500
125/125 [==============================] - 0s 1ms/step - loss: 3.1129e-05 - accuracy: 1.
0000
Epoch 149/500
125/125 [==============================] - 0s 2ms/step - loss: 3.0893e-05 - accuracy: 1.
0000
Epoch 150/500
125/125 [==============================] - 0s 2ms/step - loss: 3.0654e-05 - accuracy: 1.
0000
Epoch 151/500
125/125 [==============================] - 0s 1ms/step - loss: 3.0412e-05 - accuracy: 1.
0000
Epoch 152/500
125/125 [==============================] - 0s 1ms/step - loss: 3.0182e-05 - accuracy: 1.
0000
Epoch 153/500
125/125 [==============================] - 0s 1ms/step - loss: 2.9952e-05 - accuracy: 1.
0000
Epoch 154/500
125/125 [==============================] - 0s 1ms/step - loss: 2.9734e-05 - accuracy: 1.
0000
Epoch 155/500
125/125 [==============================] - 0s 1ms/step - loss: 2.9507e-05 - accuracy: 1.
0000
Epoch 156/500
125/125 [==============================] - 0s 1ms/step - loss: 2.9285e-05 - accuracy: 1.
0000
Epoch 157/500
125/125 [==============================] - 0s 1ms/step - loss: 2.9065e-05 - accuracy: 1.
0000
Epoch 158/500
125/125 [==============================] - 0s 1ms/step - loss: 2.8853e-05 - accuracy: 1.
0000
Epoch 159/500
125/125 [==============================] - 0s 1ms/step - loss: 2.8646e-05 - accuracy: 1.
0000
Epoch 160/500
125/125 [==============================] - 0s 1ms/step - loss: 2.8438e-05 - accuracy: 1.
0000
Epoch 161/500
125/125 [==============================] - 0s 1ms/step - loss: 2.8231e-05 - accuracy: 1.
0000
Epoch 162/500
125/125 [==============================] - 0s 1ms/step - loss: 2.8031e-05 - accuracy: 1.
0000
Epoch 163/500
125/125 [==============================] - 0s 1ms/step - loss: 2.7823e-05 - accuracy: 1.
0000
```

```
Epoch 164/500
125/125 [==============================] - 0s 2ms/step - loss: 2.7632e-05 - accuracy: 1.
0000
Epoch 165/500
125/125 [==============================] - 0s 1ms/step - loss: 2.7441e-05 - accuracy: 1.
0000
Epoch 166/500
125/125 [==============================] - 0s 1ms/step - loss: 2.7246e-05 - accuracy: 1.
0000
Epoch 167/500
125/125 [==============================] - 0s 1ms/step - loss: 2.7061e-05 - accuracy: 1.
0000
Epoch 168/500
125/125 [==============================] - 0s 1ms/step - loss: 2.6872e-05 - accuracy: 1.
0000
Epoch 169/500
125/125 [==============================] - 0s 1ms/step - loss: 2.6686e-05 - accuracy: 1.
0000
Epoch 170/500
125/125 [==============================] - 0s 1ms/step - loss: 2.6502e-05 - accuracy: 1.
0000
Epoch 171/500
125/125 [==============================] - 0s 1ms/step - loss: 2.6329e-05 - accuracy: 1.
0000
Epoch 172/500
125/125 [==============================] - 0s 1ms/step - loss: 2.6150e-05 - accuracy: 1.
0000
Epoch 173/500
125/125 [==============================] - 0s 1ms/step - loss: 2.5975e-05 - accuracy: 1.
0000
Epoch 174/500
125/125 [==============================] - 0s 1ms/step - loss: 2.5800e-05 - accuracy: 1.
0000
Epoch 175/500
125/125 [==============================] - 0s 1ms/step - loss: 2.5629e-05 - accuracy: 1.
0000
Epoch 176/500
125/125 [==============================] - 0s 1ms/step - loss: 2.5462e-05 - accuracy: 1.
0000
Epoch 177/500
125/125 [==============================] - 0s 2ms/step - loss: 2.5292e-05 - accuracy: 1.
0000
Epoch 178/500
125/125 [==============================] - 0s 1ms/step - loss: 2.5131e-05 - accuracy: 1.
0000
Epoch 179/500
125/125 [==============================] - 0s 1ms/step - loss: 2.4971e-05 - accuracy: 1.
0000
Epoch 180/500
125/125 [==============================] - 0s 1ms/step - loss: 2.4809e-05 - accuracy: 1.
0000
Epoch 181/500
125/125 [==============================] - 0s 1ms/step - loss: 2.4651e-05 - accuracy: 1.
0000
Epoch 182/500
125/125 [==============================] - 0s 1ms/step - loss: 2.4495e-05 - accuracy: 1.
0000
Epoch 183/500
125/125 [==============================] - 0s 1ms/step - loss: 2.4340e-05 - accuracy: 1.
0000
Epoch 184/500
125/125 [==============================] - 0s 1ms/step - loss: 2.4189e-05 - accuracy: 1.
0000
Epoch 185/500
125/125 [==============================] - 0s 1ms/step - loss: 2.4037e-05 - accuracy: 1.
```

```
0000
Epoch 186/500
125/125 [==============================] - 0s 1ms/step - loss: 2.3887e-05 - accuracy: 1.
0000
Epoch 187/500
125/125 [==============================] - 0s 1ms/step - loss: 2.3742e-05 - accuracy: 1.
0000
Epoch 188/500
125/125 [==============================] - 0s 1ms/step - loss: 2.3595e-05 - accuracy: 1.
0000
Epoch 189/500
125/125 [==============================] - 0s 1ms/step - loss: 2.3452e-05 - accuracy: 1.
0000
Epoch 190/500
125/125 [==============================] - 0s 1ms/step - loss: 2.3304e-05 - accuracy: 1.
0000
Epoch 191/500
125/125 [==============================] - 0s 2ms/step - loss: 2.3163e-05 - accuracy: 1.
0000
Epoch 192/500
125/125 [==============================] - 0s 1ms/step - loss: 2.3027e-05 - accuracy: 1.
0000
Epoch 193/500
125/125 [==============================] - 0s 2ms/step - loss: 2.2890e-05 - accuracy: 1.
0000
Epoch 194/500
125/125 [==============================] - 0s 1ms/step - loss: 2.2754e-05 - accuracy: 1.
0000
Epoch 195/500
125/125 [==============================] - 0s 1ms/step - loss: 2.2620e-05 - accuracy: 1.
0000
Epoch 196/500
125/125 [==============================] - 0s 2ms/step - loss: 2.2485e-05 - accuracy: 1.
0000
Epoch 197/500
125/125 [==============================] - 0s 1ms/step - loss: 2.2355e-05 - accuracy: 1.
0000
Epoch 198/500
125/125 [==============================] - 0s 1ms/step - loss: 2.2222e-05 - accuracy: 1.
0000
Epoch 199/500
125/125 [==============================] - 0s 1ms/step - loss: 2.2095e-05 - accuracy: 1.
0000
Epoch 200/500
125/125 [==============================] - 0s 2ms/step - loss: 2.1967e-05 - accuracy: 1.
0000
Epoch 201/500
125/125 [==============================] - 0s 1ms/step - loss: 2.1842e-05 - accuracy: 1.
0000
Epoch 202/500
125/125 [==============================] - 0s 1ms/step - loss: 2.1717e-05 - accuracy: 1.
0000
Epoch 203/500
125/125 [==============================] - 0s 1ms/step - loss: 2.1596e-05 - accuracy: 1.
0000
Epoch 204/500
125/125 [==============================] - 0s 1ms/step - loss: 2.1471e-05 - accuracy: 1.
0000
Epoch 205/500
125/125 [==============================] - 0s 1ms/step - loss: 2.1354e-05 - accuracy: 1.
0000
Epoch 206/500
125/125 [==============================] - 0s 1ms/step - loss: 2.1237e-05 - accuracy: 1.
0000
Epoch 207/500
```

```
125/125 [==============================] - 0s 1ms/step - loss: 2.1115e-05 - accuracy: 1.
0000
Epoch 208/500
125/125 [==============================] - 0s 1ms/step - loss: 2.0999e-05 - accuracy: 1.
0000
Epoch 209/500
125/125 [==============================] - 0s 2ms/step - loss: 2.0884e-05 - accuracy: 1.
0000
Epoch 210/500
125/125 [==============================] - 0s 1ms/step - loss: 2.0771e-05 - accuracy: 1.
0000
Epoch 211/500
125/125 [==============================] - 0s 1ms/step - loss: 2.0655e-05 - accuracy: 1.
0000
Epoch 212/500
125/125 [==============================] - 0s 1ms/step - loss: 2.0544e-05 - accuracy: 1.
0000
Epoch 213/500
125/125 [==============================] - 0s 1ms/step - loss: 2.0430e-05 - accuracy: 1.
0000
Epoch 214/500
125/125 [==============================] - 0s 1ms/step - loss: 2.0321e-05 - accuracy: 1.
0000
Epoch 215/500
125/125 [==============================] - 0s 1ms/step - loss: 2.0212e-05 - accuracy: 1.
0000
Epoch 216/500
125/125 [==============================] - 0s 1ms/step - loss: 2.0107e-05 - accuracy: 1.
0000
Epoch 217/500
125/125 [==============================] - 0s 1ms/step - loss: 1.9999e-05 - accuracy: 1.
0000
Epoch 218/500
125/125 [==============================] - 0s 1ms/step - loss: 1.9893e-05 - accuracy: 1.
0000
Epoch 219/500
125/125 [==============================] - 0s 1ms/step - loss: 1.9791e-05 - accuracy: 1.
0000
Epoch 220/500
125/125 [==============================] - 0s 1ms/step - loss: 1.9686e-05 - accuracy: 1.
0000
Epoch 221/500
125/125 [==============================] - 0s 2ms/step - loss: 1.9586e-05 - accuracy: 1.
0000
Epoch 222/500
125/125 [==============================] - 0s 2ms/step - loss: 1.9482e-05 - accuracy: 1.
0000
Epoch 223/500
125/125 [==============================] - 0s 1ms/step - loss: 1.9381e-05 - accuracy: 1.
0000
Epoch 224/500
125/125 [==============================] - 0s 1ms/step - loss: 1.9284e-05 - accuracy: 1.
0000
Epoch 225/500
125/125 [==============================] - 0s 1ms/step - loss: 1.9187e-05 - accuracy: 1.
0000
Epoch 226/500
125/125 [==============================] - 0s 1ms/step - loss: 1.9086e-05 - accuracy: 1.
0000
Epoch 227/500
125/125 [==============================] - 0s 1ms/step - loss: 1.8989e-05 - accuracy: 1.
0000
Epoch 228/500
125/125 [==============================] - 0s 1ms/step - loss: 1.8896e-05 - accuracy: 1.
0000
```

```
Epoch 229/500
125/125 [==============================] - 0s 1ms/step - loss: 1.8799e-05 - accuracy: 1.
0000
Epoch 230/500
125/125 [==============================] - 0s 2ms/step - loss: 1.8703e-05 - accuracy: 1.
0000
Epoch 231/500
125/125 [==============================] - 0s 1ms/step - loss: 1.8613e-05 - accuracy: 1.
0000
Epoch 232/500
125/125 [==============================] - 0s 1ms/step - loss: 1.8522e-05 - accuracy: 1.
0000
Epoch 233/500
125/125 [==============================] - 0s 1ms/step - loss: 1.8431e-05 - accuracy: 1.
0000
Epoch 234/500
125/125 [==============================] - 0s 1ms/step - loss: 1.8339e-05 - accuracy: 1.
0000
Epoch 235/500
125/125 [==============================] - 0s 1ms/step - loss: 1.8249e-05 - accuracy: 1.
0000
Epoch 236/500
125/125 [==============================] - 0s 1ms/step - loss: 1.8162e-05 - accuracy: 1.
0000
Epoch 237/500
125/125 [==============================] - 0s 1ms/step - loss: 1.8074e-05 - accuracy: 1.
0000
Epoch 238/500
125/125 [==============================] - 0s 2ms/step - loss: 1.7987e-05 - accuracy: 1.
0000
Epoch 239/500
125/125 [==============================] - 0s 1ms/step - loss: 1.7899e-05 - accuracy: 1.
0000
Epoch 240/500
125/125 [==============================] - 0s 1ms/step - loss: 1.7816e-05 - accuracy: 1.
0000
Epoch 241/500
125/125 [==============================] - 0s 1ms/step - loss: 1.7730e-05 - accuracy: 1.
0000
Epoch 242/500
125/125 [==============================] - 0s 2ms/step - loss: 1.7645e-05 - accuracy: 1.
0000
Epoch 243/500
125/125 [==============================] - 0s 1ms/step - loss: 1.7563e-05 - accuracy: 1.
0000
Epoch 244/500
125/125 [==============================] - 0s 1ms/step - loss: 1.7480e-05 - accuracy: 1.
0000
Epoch 245/500
125/125 [==============================] - 0s 1ms/step - loss: 1.7400e-05 - accuracy: 1.
0000
Epoch 246/500
125/125 [==============================] - 0s 1ms/step - loss: 1.7320e-05 - accuracy: 1.
0000
Epoch 247/500
125/125 [==============================] - 0s 1ms/step - loss: 1.7240e-05 - accuracy: 1.
0000
Epoch 248/500
125/125 [==============================] - 0s 1ms/step - loss: 1.7158e-05 - accuracy: 1.
0000
Epoch 249/500
125/125 [==============================] - 0s 1ms/step - loss: 1.7081e-05 - accuracy: 1.
0000
Epoch 250/500
125/125 [==============================] - 0s 1ms/step - loss: 1.7002e-05 - accuracy: 1.
```

```
0000
Epoch 251/500
125/125 [==============================] - 0s 1ms/step - loss: 1.6925e-05 - accuracy: 1.
0000
Epoch 252/500
125/125 [==============================] - 0s 1ms/step - loss: 1.6849e-05 - accuracy: 1.
0000
Epoch 253/500
125/125 [==============================] - 0s 1ms/step - loss: 1.6772e-05 - accuracy: 1.
0000
Epoch 254/500
125/125 [==============================] - 0s 1ms/step - loss: 1.6695e-05 - accuracy: 1.
0000
Epoch 255/500
125/125 [==============================] - 0s 1ms/step - loss: 1.6623e-05 - accuracy: 1.
0000
Epoch 256/500
125/125 [==============================] - 0s 1ms/step - loss: 1.6550e-05 - accuracy: 1.
0000
Epoch 257/500
125/125 [==============================] - 0s 2ms/step - loss: 1.6478e-05 - accuracy: 1.
0000
Epoch 258/500
125/125 [==============================] - 0s 1ms/step - loss: 1.6402e-05 - accuracy: 1.
0000
Epoch 259/500
125/125 [==============================] - 0s 1ms/step - loss: 1.6331e-05 - accuracy: 1.
0000
Epoch 260/500
125/125 [==============================] - 0s 1ms/step - loss: 1.6257e-05 - accuracy: 1.
0000
Epoch 261/500
125/125 [==============================] - 0s 2ms/step - loss: 1.6187e-05 - accuracy: 1.
0000
Epoch 262/500
125/125 [==============================] - 0s 1ms/step - loss: 1.6117e-05 - accuracy: 1.
0000
Epoch 263/500
125/125 [==============================] - 0s 1ms/step - loss: 1.6047e-05 - accuracy: 1.
0000
Epoch 264/500
125/125 [==============================] - 0s 1ms/step - loss: 1.5977e-05 - accuracy: 1.
0000
Epoch 265/500
125/125 [==============================] - 0s 1ms/step - loss: 1.5907e-05 - accuracy: 1.
0000
Epoch 266/500
125/125 [==============================] - 0s 2ms/step - loss: 1.5841e-05 - accuracy: 1.
0000
Epoch 267/500
125/125 [==============================] - 0s 1ms/step - loss: 1.5776e-05 - accuracy: 1.
0000
Epoch 268/500
125/125 [==============================] - 0s 1ms/step - loss: 1.5706e-05 - accuracy: 1.
0000
Epoch 269/500
125/125 [==============================] - 0s 1ms/step - loss: 1.5641e-05 - accuracy: 1.
0000
Epoch 270/500
125/125 [==============================] - 0s 1ms/step - loss: 1.5574e-05 - accuracy: 1.
0000
Epoch 271/500
125/125 [==============================] - 0s 1ms/step - loss: 1.5505e-05 - accuracy: 1.
0000
Epoch 272/500
```

```
125/125 [==============================] - 0s 1ms/step - loss: 1.5442e-05 - accuracy: 1.
0000
Epoch 273/500
125/125 [==============================] - 0s 1ms/step - loss: 1.5378e-05 - accuracy: 1.
0000
Epoch 274/500
125/125 [==============================] - 0s 1ms/step - loss: 1.5314e-05 - accuracy: 1.
0000
Epoch 275/500
125/125 [==============================] - 0s 1ms/step - loss: 1.5251e-05 - accuracy: 1.
0000
Epoch 276/500
125/125 [==============================] - 0s 1ms/step - loss: 1.5189e-05 - accuracy: 1.
0000
Epoch 277/500
125/125 [==============================] - 0s 1ms/step - loss: 1.5124e-05 - accuracy: 1.
0000
Epoch 278/500
125/125 [==============================] - 0s 1ms/step - loss: 1.5062e-05 - accuracy: 1.
0000
Epoch 279/500
125/125 [==============================] - 0s 1ms/step - loss: 1.5001e-05 - accuracy: 1.
0000
Epoch 280/500
125/125 [==============================] - 0s 1ms/step - loss: 1.4941e-05 - accuracy: 1.
0000
Epoch 281/500
125/125 [==============================] - 0s 1ms/step - loss: 1.4881e-05 - accuracy: 1.
0000
Epoch 282/500
125/125 [==============================] - 0s 1ms/step - loss: 1.4822e-05 - accuracy: 1.
0000
Epoch 283/500
125/125 [==============================] - 0s 2ms/step - loss: 1.4762e-05 - accuracy: 1.
0000
Epoch 284/500
125/125 [==============================] - 0s 1ms/step - loss: 1.4703e-05 - accuracy: 1.
0000
Epoch 285/500
125/125 [==============================] - 0s 1ms/step - loss: 1.4644e-05 - accuracy: 1.
0000
Epoch 286/500
125/125 [==============================] - 0s 1ms/step - loss: 1.4586e-05 - accuracy: 1.
0000
Epoch 287/500
125/125 [==============================] - 0s 1ms/step - loss: 1.4526e-05 - accuracy: 1.
0000
Epoch 288/500
125/125 [==============================] - 0s 1ms/step - loss: 1.4471e-05 - accuracy: 1.
0000
Epoch 289/500
125/125 [==============================] - 0s 1ms/step - loss: 1.4412e-05 - accuracy: 1.
0000
Epoch 290/500
125/125 [==============================] - 0s 1ms/step - loss: 1.4355e-05 - accuracy: 1.
0000
Epoch 291/500
125/125 [==============================] - 0s 1ms/step - loss: 1.4300e-05 - accuracy: 1.
0000
Epoch 292/500
125/125 [==============================] - 0s 1ms/step - loss: 1.4243e-05 - accuracy: 1.
0000
Epoch 293/500
125/125 [==============================] - 0s 1ms/step - loss: 1.4190e-05 - accuracy: 1.
0000
```

```
Epoch 294/500
125/125 [==============================] - 0s 1ms/step - loss: 1.4133e-05 - accuracy: 1.
0000
Epoch 295/500
125/125 [==============================] - 0s 1ms/step - loss: 1.4079e-05 - accuracy: 1.
0000
Epoch 296/500
125/125 [==============================] - 0s 2ms/step - loss: 1.4025e-05 - accuracy: 1.
0000
Epoch 297/500
125/125 [==============================] - 0s 2ms/step - loss: 1.3973e-05 - accuracy: 1.
0000
Epoch 298/500
125/125 [==============================] - 0s 2ms/step - loss: 1.3918e-05 - accuracy: 1.
0000
Epoch 299/500
125/125 [==============================] - 0s 2ms/step - loss: 1.3866e-05 - accuracy: 1.
0000
Epoch 300/500
125/125 [==============================] - 0s 1ms/step - loss: 1.3816e-05 - accuracy: 1.
0000
Epoch 301/500
125/125 [==============================] - 0s 1ms/step - loss: 1.3760e-05 - accuracy: 1.
0000
Epoch 302/500
125/125 [==============================] - 0s 1ms/step - loss: 1.3708e-05 - accuracy: 1.
0000
Epoch 303/500
125/125 [==============================] - 0s 1ms/step - loss: 1.3657e-05 - accuracy: 1.
0000
Epoch 304/500
125/125 [==============================] - 0s 1ms/step - loss: 1.3606e-05 - accuracy: 1.
0000
Epoch 305/500
125/125 [==============================] - 0s 1ms/step - loss: 1.3555e-05 - accuracy: 1.
0000
Epoch 306/500
125/125 [==============================] - 0s 1ms/step - loss: 1.3504e-05 - accuracy: 1.
0000
Epoch 307/500
125/125 [==============================] - 0s 1ms/step - loss: 1.3455e-05 - accuracy: 1.
0000
Epoch 308/500
125/125 [==============================] - 0s 1ms/step - loss: 1.3404e-05 - accuracy: 1.
0000
Epoch 309/500
125/125 [==============================] - 0s 1ms/step - loss: 1.3356e-05 - accuracy: 1.
0000
Epoch 310/500
125/125 [==============================] - 0s 2ms/step - loss: 1.3307e-05 - accuracy: 1.
0000
Epoch 311/500
125/125 [==============================] - 0s 1ms/step - loss: 1.3258e-05 - accuracy: 1.
0000
Epoch 312/500
125/125 [==============================] - 0s 1ms/step - loss: 1.3209e-05 - accuracy: 1.
0000
Epoch 313/500
125/125 [==============================] - 0s 1ms/step - loss: 1.3163e-05 - accuracy: 1.
0000
Epoch 314/500
125/125 [==============================] - 0s 1ms/step - loss: 1.3115e-05 - accuracy: 1.
0000
Epoch 315/500
125/125 [==============================] - 0s 1ms/step - loss: 1.3065e-05 - accuracy: 1.
```

```
0000
Epoch 316/500
125/125 [==============================] - 0s 2ms/step - loss: 1.3019e-05 - accuracy: 1.
0000
Epoch 317/500
125/125 [==============================] - 0s 1ms/step - loss: 1.2973e-05 - accuracy: 1.
0000
Epoch 318/500
125/125 [==============================] - 0s 1ms/step - loss: 1.2927e-05 - accuracy: 1.
0000
Epoch 319/500
125/125 [==============================] - 0s 2ms/step - loss: 1.2879e-05 - accuracy: 1.
0000
Epoch 320/500
125/125 [==============================] - 0s 1ms/step - loss: 1.2834e-05 - accuracy: 1.
0000
Epoch 321/500
125/125 [==============================] - 0s 1ms/step - loss: 1.2790e-05 - accuracy: 1.
0000
Epoch 322/500
125/125 [==============================] - 0s 1ms/step - loss: 1.2742e-05 - accuracy: 1.
0000
Epoch 323/500
125/125 [==============================] - 0s 1ms/step - loss: 1.2700e-05 - accuracy: 1.
0000
Epoch 324/500
125/125 [==============================] - 0s 1ms/step - loss: 1.2653e-05 - accuracy: 1.
0000
Epoch 325/500
125/125 [==============================] - 0s 1ms/step - loss: 1.2610e-05 - accuracy: 1.
0000
Epoch 326/500
125/125 [==============================] - 0s 1ms/step - loss: 1.2565e-05 - accuracy: 1.
0000
Epoch 327/500
125/125 [==============================] - 0s 1ms/step - loss: 1.2523e-05 - accuracy: 1.
0000
Epoch 328/500
125/125 [==============================] - 0s 1ms/step - loss: 1.2479e-05 - accuracy: 1.
0000
Epoch 329/500
125/125 [==============================] - 0s 2ms/step - loss: 1.2436e-05 - accuracy: 1.
0000
Epoch 330/500
125/125 [==============================] - 0s 1ms/step - loss: 1.2393e-05 - accuracy: 1.
0000
Epoch 331/500
125/125 [==============================] - 0s 1ms/step - loss: 1.2352e-05 - accuracy: 1.
0000
Epoch 332/500
125/125 [==============================] - 0s 1ms/step - loss: 1.2310e-05 - accuracy: 1.
0000
Epoch 333/500
125/125 [==============================] - 0s 2ms/step - loss: 1.2268e-05 - accuracy: 1.
0000
Epoch 334/500
125/125 [==============================] - 0s 1ms/step - loss: 1.2226e-05 - accuracy: 1.
0000
Epoch 335/500
125/125 [==============================] - 0s 1ms/step - loss: 1.2185e-05 - accuracy: 1.
0000
Epoch 336/500
125/125 [==============================] - 0s 1ms/step - loss: 1.2144e-05 - accuracy: 1.
0000
Epoch 337/500
```

```
125/125 [==============================] - 0s 2ms/step - loss: 1.2102e-05 - accuracy: 1.
0000
Epoch 338/500
125/125 [==============================] - 0s 2ms/step - loss: 1.2061e-05 - accuracy: 1.
0000
Epoch 339/500
125/125 [==============================] - 0s 2ms/step - loss: 1.2022e-05 - accuracy: 1.
0000
Epoch 340/500
125/125 [==============================] - 0s 2ms/step - loss: 1.1980e-05 - accuracy: 1.
0000
Epoch 341/500
125/125 [==============================] - 0s 2ms/step - loss: 1.1940e-05 - accuracy: 1.
0000
Epoch 342/500
125/125 [==============================] - 0s 1ms/step - loss: 1.1900e-05 - accuracy: 1.
0000
Epoch 343/500
125/125 [==============================] - 0s 2ms/step - loss: 1.1861e-05 - accuracy: 1.
0000
Epoch 344/500
125/125 [==============================] - 0s 2ms/step - loss: 1.1824e-05 - accuracy: 1.
0000
Epoch 345/500
125/125 [==============================] - 0s 2ms/step - loss: 1.1781e-05 - accuracy: 1.
0000
Epoch 346/500
125/125 [==============================] - 0s 1ms/step - loss: 1.1743e-05 - accuracy: 1.
0000
Epoch 347/500
125/125 [==============================] - 0s 1ms/step - loss: 1.1706e-05 - accuracy: 1.
0000
Epoch 348/500
125/125 [==============================] - 0s 1ms/step - loss: 1.1667e-05 - accuracy: 1.
0000
Epoch 349/500
125/125 [==============================] - 0s 2ms/step - loss: 1.1630e-05 - accuracy: 1.
0000
Epoch 350/500
125/125 [==============================] - 0s 2ms/step - loss: 1.1592e-05 - accuracy: 1.
0000
Epoch 351/500
125/125 [==============================] - 0s 2ms/step - loss: 1.1554e-05 - accuracy: 1.
0000
Epoch 352/500
125/125 [==============================] - 0s 1ms/step - loss: 1.1519e-05 - accuracy: 1.
0000
Epoch 353/500
125/125 [==============================] - 0s 1ms/step - loss: 1.1482e-05 - accuracy: 1.
0000
Epoch 354/500
125/125 [==============================] - 0s 1ms/step - loss: 1.1444e-05 - accuracy: 1.
0000
Epoch 355/500
125/125 [==============================] - 0s 1ms/step - loss: 1.1406e-05 - accuracy: 1.
0000
Epoch 356/500
125/125 [==============================] - 0s 1ms/step - loss: 1.1370e-05 - accuracy: 1.
0000
Epoch 357/500
125/125 [==============================] - 0s 1ms/step - loss: 1.1336e-05 - accuracy: 1.
0000
Epoch 358/500
125/125 [==============================] - 0s 2ms/step - loss: 1.1299e-05 - accuracy: 1.
0000
```

```
Epoch 359/500
125/125 [==============================] - 0s 2ms/step - loss: 1.1265e-05 - accuracy: 1.
0000
Epoch 360/500
125/125 [==============================] - 0s 2ms/step - loss: 1.1229e-05 - accuracy: 1.
0000
Epoch 361/500
125/125 [==============================] - 0s 2ms/step - loss: 1.1195e-05 - accuracy: 1.
0000
Epoch 362/500
125/125 [==============================] - 0s 1ms/step - loss: 1.1159e-05 - accuracy: 1.
0000
Epoch 363/500
125/125 [==============================] - 0s 1ms/step - loss: 1.1122e-05 - accuracy: 1.
0000
Epoch 364/500
125/125 [==============================] - 0s 1ms/step - loss: 1.1089e-05 - accuracy: 1.
0000
Epoch 365/500
125/125 [==============================] - 0s 2ms/step - loss: 1.1055e-05 - accuracy: 1.
0000
Epoch 366/500
125/125 [==============================] - 0s 2ms/step - loss: 1.1019e-05 - accuracy: 1.
0000
Epoch 367/500
125/125 [==============================] - 0s 1ms/step - loss: 1.0987e-05 - accuracy: 1.
0000
Epoch 368/500
125/125 [==============================] - 0s 2ms/step - loss: 1.0952e-05 - accuracy: 1.
0000
Epoch 369/500
125/125 [==============================] - 0s 2ms/step - loss: 1.0919e-05 - accuracy: 1.
0000
Epoch 370/500
125/125 [==============================] - 0s 1ms/step - loss: 1.0885e-05 - accuracy: 1.
0000
Epoch 371/500
125/125 [==============================] - 0s 2ms/step - loss: 1.0852e-05 - accuracy: 1.
0000
Epoch 372/500
125/125 [==============================] - 0s 2ms/step - loss: 1.0819e-05 - accuracy: 1.
0000
Epoch 373/500
125/125 [==============================] - 0s 2ms/step - loss: 1.0784e-05 - accuracy: 1.
0000
Epoch 374/500
125/125 [==============================] - 0s 1ms/step - loss: 1.0752e-05 - accuracy: 1.
0000
Epoch 375/500
125/125 [==============================] - 0s 1ms/step - loss: 1.0721e-05 - accuracy: 1.
0000
Epoch 376/500
125/125 [==============================] - 0s 2ms/step - loss: 1.0689e-05 - accuracy: 1.
0000
Epoch 377/500
125/125 [==============================] - 0s 2ms/step - loss: 1.0655e-05 - accuracy: 1.
0000
Epoch 378/500
125/125 [==============================] - 0s 1ms/step - loss: 1.0625e-05 - accuracy: 1.
0000
Epoch 379/500
125/125 [==============================] - 0s 2ms/step - loss: 1.0593e-05 - accuracy: 1.
0000
Epoch 380/500
125/125 [==============================] - 0s 1ms/step - loss: 1.0562e-05 - accuracy: 1.
```

```
0000
Epoch 381/500
125/125 [==============================] - 0s 1ms/step - loss: 1.0530e-05 - accuracy: 1.
0000
Epoch 382/500
125/125 [==============================] - 0s 2ms/step - loss: 1.0498e-05 - accuracy: 1.
0000
Epoch 383/500
125/125 [==============================] - 0s 2ms/step - loss: 1.0467e-05 - accuracy: 1.
0000
Epoch 384/500
125/125 [==============================] - 0s 2ms/step - loss: 1.0435e-05 - accuracy: 1.
0000
Epoch 385/500
125/125 [==============================] - 0s 1ms/step - loss: 1.0407e-05 - accuracy: 1.
0000
Epoch 386/500
125/125 [==============================] - 0s 1ms/step - loss: 1.0374e-05 - accuracy: 1.
0000
Epoch 387/500
125/125 [==============================] - 0s 1ms/step - loss: 1.0344e-05 - accuracy: 1.
0000
Epoch 388/500
125/125 [==============================] - 0s 2ms/step - loss: 1.0315e-05 - accuracy: 1.
0000
Epoch 389/500
125/125 [==============================] - 0s 2ms/step - loss: 1.0284e-05 - accuracy: 1.
0000
Epoch 390/500
125/125 [==============================] - 0s 2ms/step - loss: 1.0257e-05 - accuracy: 1.
0000
Epoch 391/500
125/125 [==============================] - 0s 2ms/step - loss: 1.0226e-05 - accuracy: 1.
0000
Epoch 392/500
125/125 [==============================] - 0s 1ms/step - loss: 1.0196e-05 - accuracy: 1.
0000
Epoch 393/500
125/125 [==============================] - 0s 2ms/step - loss: 1.0167e-05 - accuracy: 1.
0000
Epoch 394/500
125/125 [==============================] - 0s 1ms/step - loss: 1.0140e-05 - accuracy: 1.
0000
Epoch 395/500
125/125 [==============================] - 0s 2ms/step - loss: 1.0108e-05 - accuracy: 1.
0000
Epoch 396/500
125/125 [==============================] - 0s 1ms/step - loss: 1.0081e-05 - accuracy: 1.
0000
Epoch 397/500
125/125 [==============================] - 0s 2ms/step - loss: 1.0052e-05 - accuracy: 1.
0000
Epoch 398/500
125/125 [==============================] - 0s 2ms/step - loss: 1.0022e-05 - accuracy: 1.
0000
Epoch 399/500
125/125 [==============================] - 0s 1ms/step - loss: 9.9938e-06 - accuracy: 1.
0000
Epoch 400/500
125/125 [==============================] - 0s 1ms/step - loss: 9.9663e-06 - accuracy: 1.
0000
Epoch 401/500
125/125 [==============================] - 0s 1ms/step - loss: 9.9382e-06 - accuracy: 1.
0000
Epoch 402/500
```

```
125/125 [==============================] - 0s 1ms/step - loss: 9.9121e-06 - accuracy: 1.
0000
Epoch 403/500
125/125 [==============================] - 0s 1ms/step - loss: 9.8825e-06 - accuracy: 1.
0000
Epoch 404/500
125/125 [==============================] - 0s 1ms/step - loss: 9.8544e-06 - accuracy: 1.
0000
Epoch 405/500
125/125 [==============================] - 0s 1ms/step - loss: 9.8266e-06 - accuracy: 1.
0000
Epoch 406/500
125/125 [==============================] - 0s 2ms/step - loss: 9.7996e-06 - accuracy: 1.
0000
Epoch 407/500
125/125 [==============================] - 0s 1ms/step - loss: 9.7714e-06 - accuracy: 1.
0000
Epoch 408/500
125/125 [==============================] - 0s 2ms/step - loss: 9.7453e-06 - accuracy: 1.
0000
Epoch 409/500
125/125 [==============================] - 0s 2ms/step - loss: 9.7173e-06 - accuracy: 1.
0000
Epoch 410/500
125/125 [==============================] - 0s 2ms/step - loss: 9.6906e-06 - accuracy: 1.
0000
Epoch 411/500
125/125 [==============================] - 0s 1ms/step - loss: 9.6633e-06 - accuracy: 1.
0000
Epoch 412/500
125/125 [==============================] - 0s 1ms/step - loss: 9.6382e-06 - accuracy: 1.
0000
Epoch 413/500
125/125 [==============================] - 0s 2ms/step - loss: 9.6120e-06 - accuracy: 1.
0000
Epoch 414/500
125/125 [==============================] - 0s 2ms/step - loss: 9.5859e-06 - accuracy: 1.
0000
Epoch 415/500
125/125 [==============================] - 0s 1ms/step - loss: 9.5594e-06 - accuracy: 1.
0000
Epoch 416/500
125/125 [==============================] - 0s 2ms/step - loss: 9.5334e-06 - accuracy: 1.
0000
Epoch 417/500
125/125 [==============================] - 0s 1ms/step - loss: 9.5077e-06 - accuracy: 1.
0000
Epoch 418/500
125/125 [==============================] - 0s 1ms/step - loss: 9.4818e-06 - accuracy: 1.
0000
Epoch 419/500
125/125 [==============================] - 0s 1ms/step - loss: 9.4562e-06 - accuracy: 1.
0000
Epoch 420/500
125/125 [==============================] - 0s 1ms/step - loss: 9.4316e-06 - accuracy: 1.
0000
Epoch 421/500
125/125 [==============================] - 0s 1ms/step - loss: 9.4065e-06 - accuracy: 1.
0000
Epoch 422/500
125/125 [==============================] - 0s 2ms/step - loss: 9.3828e-06 - accuracy: 1.
0000
Epoch 423/500
125/125 [==============================] - 0s 1ms/step - loss: 9.3564e-06 - accuracy: 1.
0000
```

```
Epoch 424/500
125/125 [==============================] - 0s 1ms/step - loss: 9.3297e-06 - accuracy: 1.
0000
Epoch 425/500
125/125 [==============================] - 0s 1ms/step - loss: 9.3055e-06 - accuracy: 1.
0000
Epoch 426/500
125/125 [==============================] - 0s 2ms/step - loss: 9.2824e-06 - accuracy: 1.
0000
Epoch 427/500
125/125 [==============================] - 0s 1ms/step - loss: 9.2557e-06 - accuracy: 1.
0000
Epoch 428/500
125/125 [==============================] - 0s 2ms/step - loss: 9.2317e-06 - accuracy: 1.
0000
Epoch 429/500
125/125 [==============================] - 0s 1ms/step - loss: 9.2070e-06 - accuracy: 1.
0000
Epoch 430/500
125/125 [==============================] - 0s 1ms/step - loss: 9.1834e-06 - accuracy: 1.
0000
Epoch 431/500
125/125 [==============================] - 0s 1ms/step - loss: 9.1589e-06 - accuracy: 1.
0000
Epoch 432/500
125/125 [==============================] - 0s 2ms/step - loss: 9.1351e-06 - accuracy: 1.
0000
Epoch 433/500
125/125 [==============================] - 0s 1ms/step - loss: 9.1118e-06 - accuracy: 1.
0000
Epoch 434/500
125/125 [==============================] - 0s 2ms/step - loss: 9.0879e-06 - accuracy: 1.
0000
Epoch 435/500
125/125 [==============================] - 0s 2ms/step - loss: 9.0643e-06 - accuracy: 1.
0000
Epoch 436/500
125/125 [==============================] - 0s 2ms/step - loss: 9.0398e-06 - accuracy: 1.
0000
Epoch 437/500
125/125 [==============================] - 0s 2ms/step - loss: 9.0168e-06 - accuracy: 1.
0000
Epoch 438/500
125/125 [==============================] - 0s 1ms/step - loss: 8.9934e-06 - accuracy: 1.
0000
Epoch 439/500
125/125 [==============================] - 0s 1ms/step - loss: 8.9696e-06 - accuracy: 1.
0000
Epoch 440/500
125/125 [==============================] - 0s 2ms/step - loss: 8.9465e-06 - accuracy: 1.
0000
Epoch 441/500
125/125 [==============================] - 0s 2ms/step - loss: 8.9247e-06 - accuracy: 1.
0000
Epoch 442/500
125/125 [==============================] - 0s 2ms/step - loss: 8.9006e-06 - accuracy: 1.
0000
Epoch 443/500
125/125 [==============================] - 0s 2ms/step - loss: 8.8797e-06 - accuracy: 1.
0000
Epoch 444/500
125/125 [==============================] - 0s 2ms/step - loss: 8.8562e-06 - accuracy: 1.
0000
Epoch 445/500
125/125 [==============================] - 0s 2ms/step - loss: 8.8355e-06 - accuracy: 1.
```

```
0000
Epoch 446/500
125/125 [==============================] - 0s 2ms/step - loss: 8.8114e-06 - accuracy: 1.
0000
Epoch 447/500
125/125 [==============================] - 0s 2ms/step - loss: 8.7908e-06 - accuracy: 1.
0000
Epoch 448/500
125/125 [==============================] - 0s 2ms/step - loss: 8.7672e-06 - accuracy: 1.
0000
Epoch 449/500
125/125 [==============================] - 0s 2ms/step - loss: 8.7440e-06 - accuracy: 1.
0000
Epoch 450/500
125/125 [==============================] - 0s 2ms/step - loss: 8.7235e-06 - accuracy: 1.
0000
Epoch 451/500
125/125 [==============================] - 0s 2ms/step - loss: 8.7011e-06 - accuracy: 1.
0000
Epoch 452/500
125/125 [==============================] - 0s 1ms/step - loss: 8.6797e-06 - accuracy: 1.
0000
Epoch 453/500
125/125 [==============================] - 0s 2ms/step - loss: 8.6590e-06 - accuracy: 1.
0000
Epoch 454/500
125/125 [==============================] - 0s 1ms/step - loss: 8.6372e-06 - accuracy: 1.
0000
Epoch 455/500
125/125 [==============================] - 0s 2ms/step - loss: 8.6157e-06 - accuracy: 1.
0000
Epoch 456/500
125/125 [==============================] - 0s 1ms/step - loss: 8.5946e-06 - accuracy: 1.
0000
Epoch 457/500
125/125 [==============================] - 0s 1ms/step - loss: 8.5740e-06 - accuracy: 1.
0000
Epoch 458/500
125/125 [==============================] - 0s 2ms/step - loss: 8.5527e-06 - accuracy: 1.
0000
Epoch 459/500
125/125 [==============================] - 0s 2ms/step - loss: 8.5316e-06 - accuracy: 1.
0000
Epoch 460/500
125/125 [==============================] - 0s 2ms/step - loss: 8.5114e-06 - accuracy: 1.
0000
Epoch 461/500
125/125 [==============================] - 0s 1ms/step - loss: 8.4895e-06 - accuracy: 1.
0000
Epoch 462/500
125/125 [==============================] - 0s 2ms/step - loss: 8.4672e-06 - accuracy: 1.
0000
Epoch 463/500
125/125 [==============================] - 0s 1ms/step - loss: 8.4470e-06 - accuracy: 1.
0000
Epoch 464/500
125/125 [==============================] - 0s 1ms/step - loss: 8.4269e-06 - accuracy: 1.
0000
Epoch 465/500
125/125 [==============================] - 0s 2ms/step - loss: 8.4072e-06 - accuracy: 1.
0000
Epoch 466/500
125/125 [==============================] - 0s 2ms/step - loss: 8.3866e-06 - accuracy: 1.
0000
Epoch 467/500
```

```
125/125 [==============================] - 0s 2ms/step - loss: 8.3647e-06 - accuracy: 1.
0000
Epoch 468/500
125/125 [==============================] - 0s 2ms/step - loss: 8.3450e-06 - accuracy: 1.
0000
Epoch 469/500
125/125 [==============================] - 0s 1ms/step - loss: 8.3244e-06 - accuracy: 1.
0000
Epoch 470/500
125/125 [==============================] - 0s 1ms/step - loss: 8.3053e-06 - accuracy: 1.
0000
Epoch 471/500
125/125 [==============================] - 0s 1ms/step - loss: 8.2858e-06 - accuracy: 1.
0000
Epoch 472/500
125/125 [==============================] - 0s 1ms/step - loss: 8.2662e-06 - accuracy: 1.
0000
Epoch 473/500
125/125 [==============================] - 0s 2ms/step - loss: 8.2470e-06 - accuracy: 1.
0000
Epoch 474/500
125/125 [==============================] - 0s 2ms/step - loss: 8.2270e-06 - accuracy: 1.
0000
Epoch 475/500
125/125 [==============================] - 0s 1ms/step - loss: 8.2066e-06 - accuracy: 1.
0000
Epoch 476/500
125/125 [==============================] - 0s 1ms/step - loss: 8.1865e-06 - accuracy: 1.
0000
Epoch 477/500
125/125 [==============================] - 0s 2ms/step - loss: 8.1673e-06 - accuracy: 1.
0000
Epoch 478/500
125/125 [==============================] - 0s 1ms/step - loss: 8.1481e-06 - accuracy: 1.
0000
Epoch 479/500
125/125 [==============================] - 0s 2ms/step - loss: 8.1288e-06 - accuracy: 1.
0000
Epoch 480/500
125/125 [==============================] - 0s 1ms/step - loss: 8.1101e-06 - accuracy: 1.
0000
Epoch 481/500
125/125 [==============================] - 0s 1ms/step - loss: 8.0910e-06 - accuracy: 1.
0000
Epoch 482/500
125/125 [==============================] - 0s 1ms/step - loss: 8.0724e-06 - accuracy: 1.
0000
Epoch 483/500
125/125 [==============================] - 0s 2ms/step - loss: 8.0546e-06 - accuracy: 1.
0000
Epoch 484/500
125/125 [==============================] - 0s 1ms/step - loss: 8.0356e-06 - accuracy: 1.
0000
Epoch 485/500
125/125 [==============================] - 0s 2ms/step - loss: 8.0158e-06 - accuracy: 1.
0000
Epoch 486/500
125/125 [==============================] - 0s 2ms/step - loss: 7.9984e-06 - accuracy: 1.
0000
Epoch 487/500
125/125 [==============================] - 0s 1ms/step - loss: 7.9788e-06 - accuracy: 1.
0000
Epoch 488/500
125/125 [==============================] - 0s 2ms/step - loss: 7.9605e-06 - accuracy: 1.
0000
```

```
Epoch 489/500
125/125 [==============================] - 0s 2ms/step - loss: 7.9408e-06 - accuracy: 1.
0000
Epoch 490/500
125/125 [==============================] - 0s 1ms/step - loss: 7.9242e-06 - accuracy: 1.
0000
Epoch 491/500
125/125 [==============================] - 0s 1ms/step - loss: 7.9062e-06 - accuracy: 1.
0000
Epoch 492/500
125/125 [==============================] - 0s 1ms/step - loss: 7.8880e-06 - accuracy: 1.
0000
Epoch 493/500
125/125 [==============================] - 0s 2ms/step - loss: 7.8690e-06 - accuracy: 1.
0000
Epoch 494/500
125/125 [==============================] - 0s 1ms/step - loss: 7.8515e-06 - accuracy: 1.
0000
Epoch 495/500
125/125 [==============================] - 0s 2ms/step - loss: 7.8345e-06 - accuracy: 1.
0000
Epoch 496/500
125/125 [==============================] - 0s 2ms/step - loss: 7.8172e-06 - accuracy: 1.
0000
Epoch 497/500
125/125 [==============================] - 0s 2ms/step - loss: 7.7981e-06 - accuracy: 1.
0000
Epoch 498/500
125/125 [==============================] - 0s 2ms/step - loss: 7.7794e-06 - accuracy: 1.
0000
Epoch 499/500
125/125 [==============================] - 0s 2ms/step - loss: 7.7620e-06 - accuracy: 1.
0000
Epoch 500/500
125/125 [==============================] - 0s 2ms/step - loss: 7.7447e-06 - accuracy: 1.
0000
```

Out[65]:
```
<tensorflow.python.keras.callbacks.History at 0x7fba0c7bf908>
```

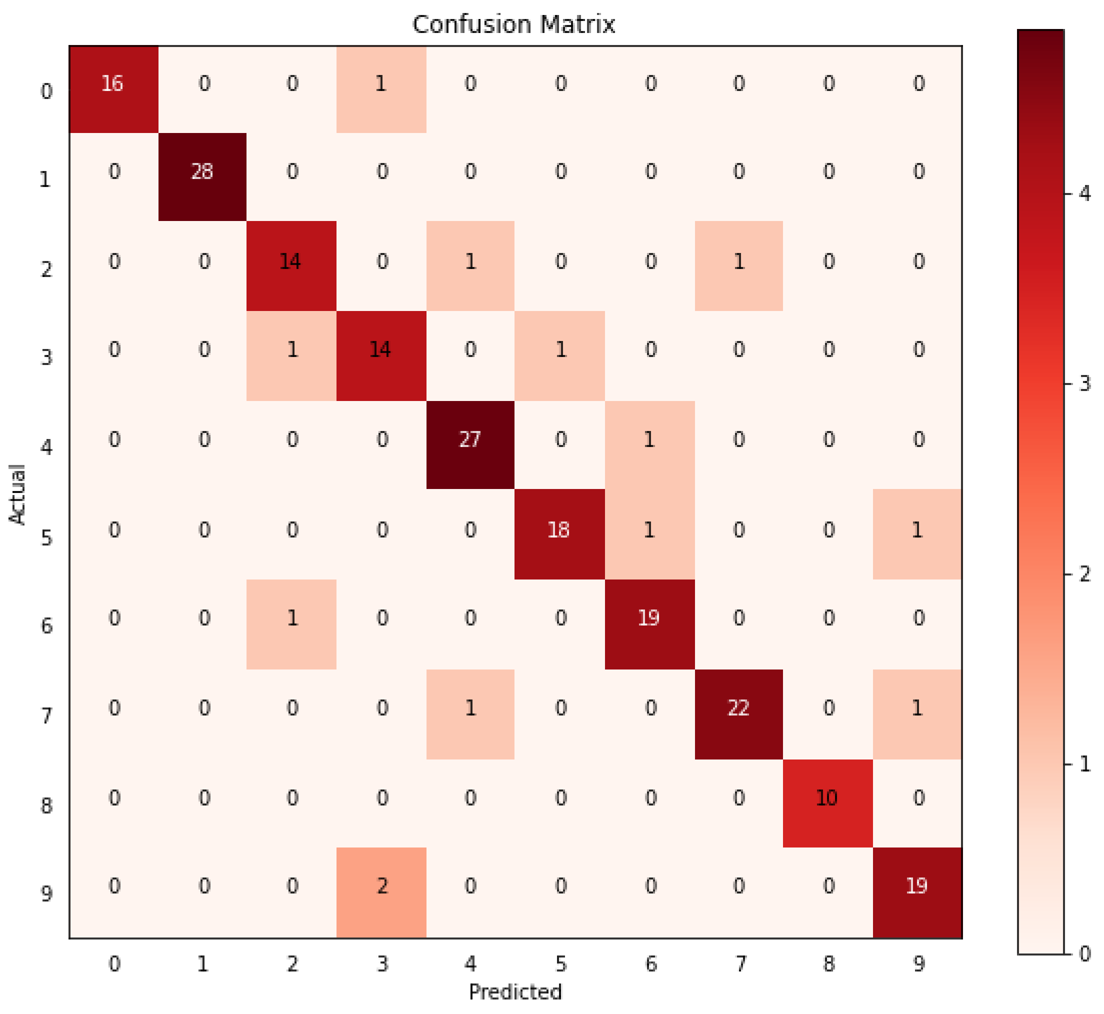
In [72]:
```python
X_valid_new = X_valid[:200]
valid_labels_new = valid_labels[:200]
# Evaluate model on test data.
score = model.evaluate(X_valid_new, valid_labels_new, verbose=0)
# This returns only a score, so you will need to use another function for
# extracting predicted labels for your confusion matrix. Use this line for that:
X_test_new = X_test[:200]
classes = model.predict_classes(X_test_new, batch_size=8)
print('Validation score:', score[0])
print('Validation accuracy:', score[1])
```

```
Validation score: 0.8063956499099731
Validation accuracy: 0.875
```

In [74]:
```python
val_p = classes
error = 0
test_labels_1_new = test_labels_1[:200]
confusion_matrix = np.zeros([10,10])
for i in range(X_test_new.shape[0]):
    confusion_matrix[test_labels_1_new[i],val_p[i]] += 1
    if test_labels_1_new[i]!=val_p[i]:
        error +=1
print("Confusion Matrix: \n\n" ,confusion_matrix)
print("\nErrors in validation set: " ,error)
print("\nError Persentage : " ,(error*100)/val_p.shape[0])
print("\nAccuracy : " ,100-(error*100)/val_p.shape[0])
print("\nValidation set Shape :",val_p.shape[0])
```

```
Confusion Matrix:

 [[16.  0.  0.  1.  0.  0.  0.  0.  0.  0.]
 [ 0. 28.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0. 14.  0.  1.  0.  0.  1.  0.  0.]
 [ 0.  0.  1. 14.  0.  1.  0.  0.  0.  0.]
 [ 0.  0.  0.  0. 27.  0.  1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0. 18.  1.  0.  0.  1.]
 [ 0.  0.  1.  0.  0.  0. 19.  0.  0.  0.]
 [ 0.  0.  0.  0.  1.  0.  0. 22.  0.  1.]
 [ 0.  0.  0.  0.  0.  0.  0.  0. 10.  0.]
 [ 0.  0.  0.  2.  0.  0.  0.  0.  0. 19.]]

Errors in validation set:  13

Error Persentage :  6.5

Accuracy :  93.5

Validation set Shape : 200
```

In [75]:
```
f = plt.figure(figsize=(10,8.5))
f.add_subplot(111)

plt.imshow(np.log2(confusion_matrix+1),cmap="Reds")
plt.colorbar()
plt.tick_params(size=5,color="white")
plt.xticks(np.arange(0,10),np.arange(0,10))
plt.yticks(np.arange(0,10),np.arange(0,10))

threshold = confusion_matrix.max()/2

for i in range(10):
    for j in range(10):
        plt.text(j,i,int(confusion_matrix[i,j]),horizontalalignment="center",color="whi

plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.savefig("Confusion_matrix1.png")
plt.show()
```



	0	1	2	3	4	5	6	7	8	9
0	16	0	0	1	0	0	0	0	0	0
1	0	28	0	0	0	0	0	0	0	0
2	0	0	14	0	1	0	0	1	0	0
3	0	0	1	14	0	1	0	0	0	0
4	0	0	0	0	27	0	1	0	0	0
5	0	0	0	0	0	18	1	0	0	1
6	0	0	1	0	0	0	19	0	0	0
7	0	0	0	0	1	0	0	22	0	1
8	0	0	0	0	0	0	0	0	10	0
9	0	0	0	2	0	0	0	0	0	19

1. How many parameters does your neural network created in the Task 2 have?

Hyperparamters:

learning rate = 0.05

batchsize = 8

number of epochs = 3

number of hidden layers = 4

number of nodes in hidden layer = 100

number of nodes in input layer = 784

number of nodes in output layer = 10

Accuaracy obtained = 96.62

1. Find out by training the neural network on just the first 1000 of the 50,000 training examples, for 500 iterations. Write down the loss and the accuracy on the training set as well as the test set for the trained network; leave your answer in the file task3-answers.txt.

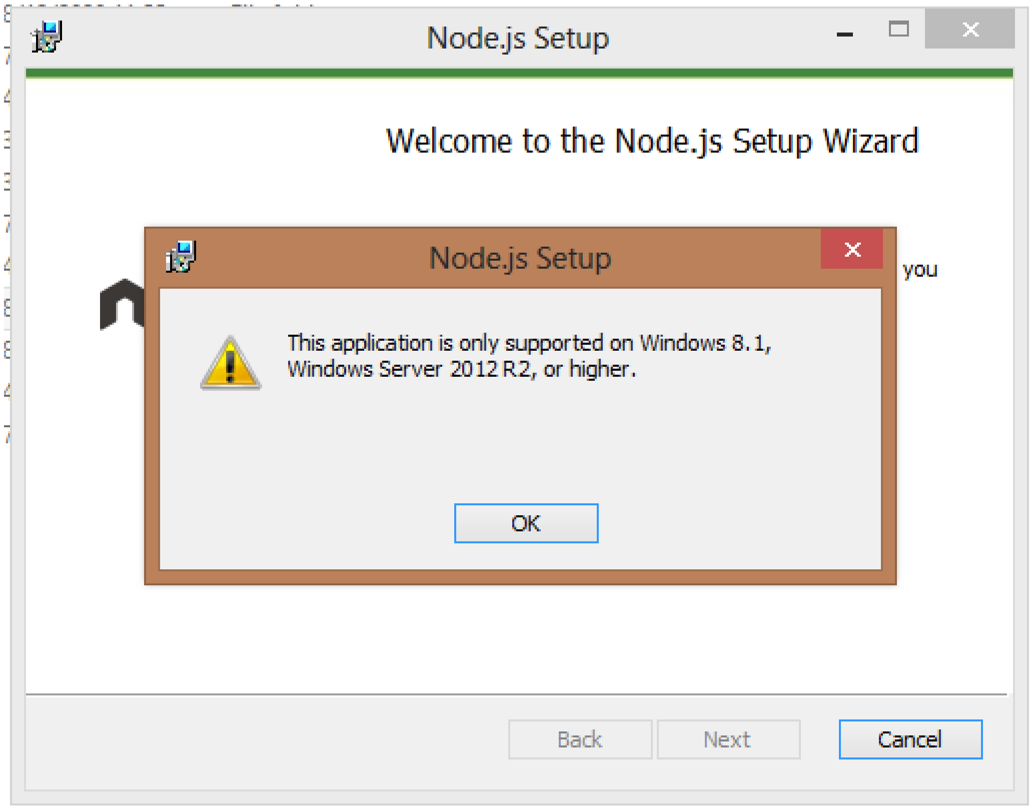
For training set, loss = 0.80, accuracy = 0.87.

For test set, loss = 0.06, accuracy = 0.93.

The accuracy dropped from 0.96(no overfitting) to 0.93 because of overfitting.

Task 4: Visualizing a neural network

In this task, we want you to play around with neural networks in an edited version of the Tensorflow Playground, and to classify the `spiral' and the 'flower' data sets using as few layers (and number of neurons in those layers) as possible. Under the "Data" panel on the left, 'spiral' is the data set in the second row and second column, while 'flower' is the data set in the third row and first column. Use only the features X1 and X2, without regularisation. Feel free to choose your activation function, learning rate and other hyperparameters.

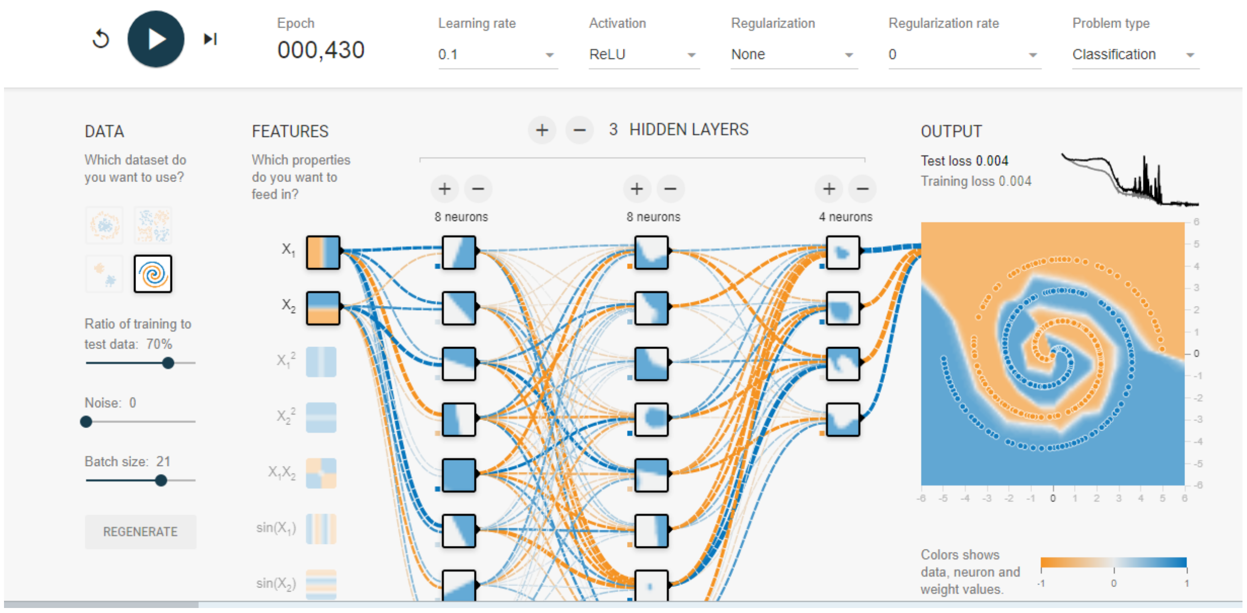

Node.js Setup

Welcome to the Node.js Setup Wizard

Node.js Setup

This application is only supported on Windows 8.1, Windows Server 2012 R2, or higher.

Since, I have windows 8, npm module was not getting installed showing the above mentioned error. No other compatible version was available in their official site. For that reason, I have done in task in the official tensorflow playground website. Although I hhave used spiral dataset set but for the second one, I have used the xor dataset instead of the flower dataset as the flower dataset was not available in the website.


Epoch 000,430 | Learning rate 0.1 | Activation ReLU | Regularization None | Regularization rate 0 | Problem type Classification

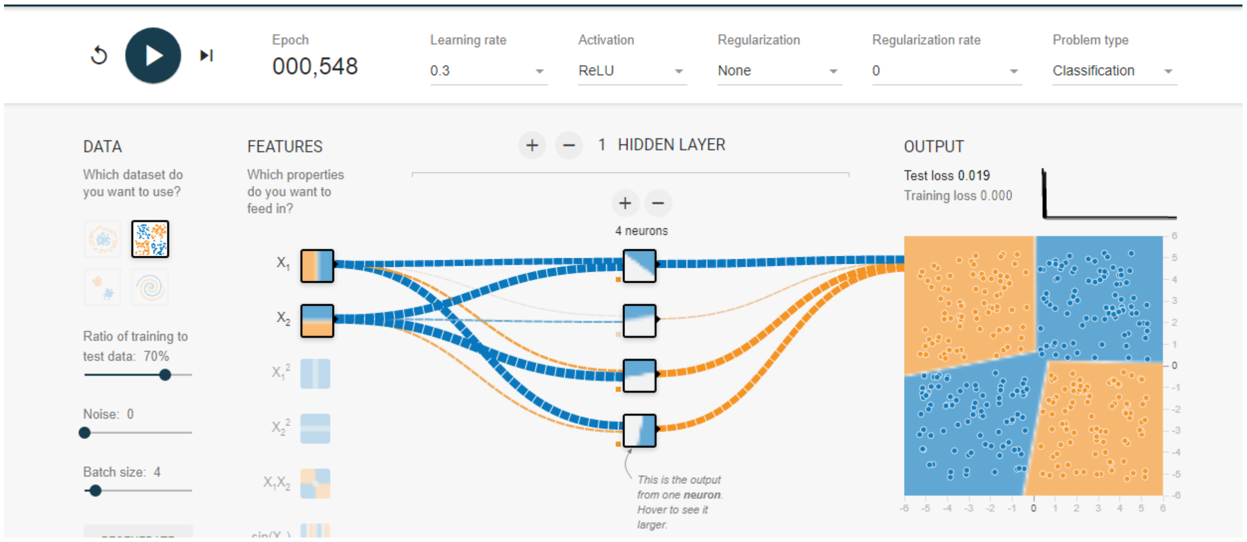
DATA Which dataset do you want to use? Ratio of training to test data: 70% Noise: 0 Batch size: 21 REGENERATE

FEATURES Which properties do you want to feed in?

3 HIDDEN LAYERS: 8 neurons, 8 neurons, 4 neurons

OUTPUT Test loss 0.004 Training loss 0.004

Colors shows data, neuron and weight values.


Epoch 000,548 | Learning rate 0.3 | Activation ReLU | Regularization None | Regularization rate 0 | Problem type Classification

DATA Which dataset do you want to use? Ratio of training to test data: 70% Noise: 0 Batch size: 4

FEATURES Which properties do you want to feed in?

1 HIDDEN LAYER: 4 neurons

OUTPUT Test loss 0.019 Training loss 0.000

Observations:

The output layer should be greater than or equal to the input layer. At least that's what I noticed in the case of this spiral problem. Keep the initial learning rate high, like 0.1 in this case, then as you approach a low test error like 3-5% or less, decrease the learning rate by a notch(0.03) or two. This helps in converging faster and avoids jumping around the global minima. You can see the effects of keeping the learning rate high by checking the error graph at the top right. For smaller batch sizes like 1, 0.1 is too high a learning rate as the model fails to converge as it jumps around the global minima. So, if you would like to keep a high learning rate(0.1), keep the batch size high(10) as well. This usually gives a slow yet smoother convergence.