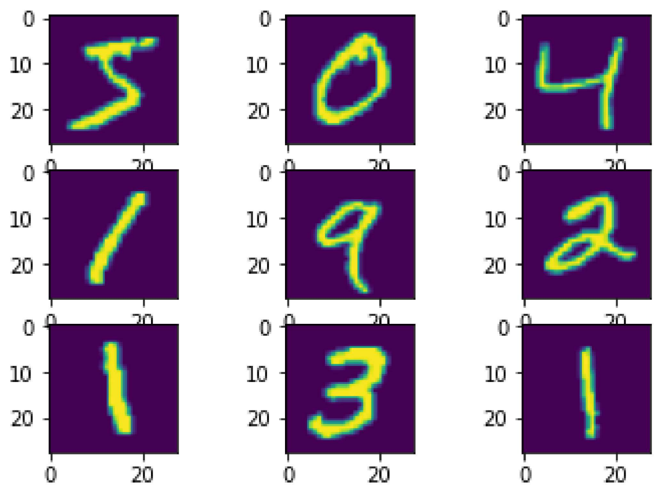
```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, BatchNormalization, Activation
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.utils import np_utils
from keras.datasets import mnist
import PIL
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from tensorflow import keras
```

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
```

```
import matplotlib.pyplot as plt
for i in range(9):
  plt.subplot(330+i+1)
  plt.imshow(x_train[i])
plt.show()
```



```
x = x_test
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train = x_train/255
x_test = x_test/255
```

```
y_train = np_utils.to_categorical(y_train,10)
```

```
y_test = np_utils.to_categorical(y_test,10)
```

```
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, BatchNormalization, Activation
from tensorflow.keras.models import Sequential
model=Sequential()
model.add(Conv2D(64,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'
model.add(Conv2D(64,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'
model.add(MaxPooling2D(2,2))
```

```
model.add(Flatten())
```

```
model.add(Dense(128, activation = 'relu', kernel_initializer ='he_uniform',input_shape=(28
model.add(Dropout(0.2))
model.add(Dense(10,activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(10, activation = 'softmax'))
```

```
from tensorflow.keras.optimizers import SGD
opt = SGD(lr = 0.01, momentum = 0.9)
from keras.backend import categorical_crossentropy
model.compile(optimizer = opt, loss = 'categorical_crossentropy', metrics =['accuracy'])
```

```
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descent.py:102: UserW
  super(SGD, self).__init__(name, **kwargs)
```

```
history = model.fit(x_train, y_train, epochs = 50, batch_size = 128,validation_data = (x_t

```

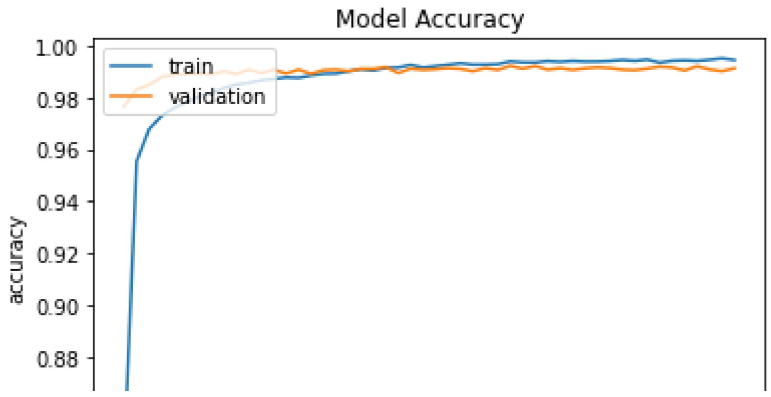
```
Epoch 23/50
469/469 [==============================] - 10s 20ms/step - loss: 0.0260 - accuracy:
Epoch 24/50
469/469 [==============================] - 10s 20ms/step - loss: 0.0232 - accuracy:
Epoch 25/50
469/469 [==============================] - 10s 21ms/step - loss: 0.0263 - accuracy:
Epoch 26/50
469/469 [==============================] - 10s 21ms/step - loss: 0.0245 - accuracy:
Epoch 27/50
469/469 [==============================] - 10s 21ms/step - loss: 0.0227 - accuracy:
Epoch 28/50
469/469 [==============================] - 10s 21ms/step - loss: 0.0222 - accuracy:
Epoch 29/50
469/469 [==============================] - 10s 22ms/step - loss: 0.0229 - accuracy:
Epoch 30/50
469/469 [==============================] - 10s 21ms/step - loss: 0.0223 - accuracy:
Epoch 31/50
469/469 [==============================] - 10s 21ms/step - loss: 0.0217 - accuracy:
Epoch 32/50
```

```
469/469 [==============================] - 10s 21ms/step - loss: 0.0196 - accuracy:
Epoch 33/50
469/469 [==============================] - 10s 21ms/step - loss: 0.0198 - accuracy:
Epoch 34/50
469/469 [==============================] - 10s 21ms/step - loss: 0.0196 - accuracy:
Epoch 35/50
469/469 [==============================] - 10s 21ms/step - loss: 0.0189 - accuracy:
Epoch 36/50
469/469 [==============================] - 10s 21ms/step - loss: 0.0191 - accuracy:
Epoch 37/50
469/469 [==============================] - 10s 21ms/step - loss: 0.0183 - accuracy:
Epoch 38/50
469/469 [==============================] - 10s 21ms/step - loss: 0.0188 - accuracy:
Epoch 39/50
469/469 [==============================] - 10s 21ms/step - loss: 0.0180 - accuracy:
Epoch 40/50
469/469 [==============================] - 10s 21ms/step - loss: 0.0177 - accuracy:
Epoch 41/50
469/469 [==============================] - 10s 21ms/step - loss: 0.0161 - accuracy:
Epoch 42/50
469/469 [==============================] - 10s 21ms/step - loss: 0.0166 - accuracy:
Epoch 43/50
469/469 [==============================] - 10s 21ms/step - loss: 0.0164 - accuracy:
Epoch 44/50
469/469 [==============================] - 10s 21ms/step - loss: 0.0187 - accuracy:
Epoch 45/50
469/469 [==============================] - 10s 21ms/step - loss: 0.0169 - accuracy:
Epoch 46/50
469/469 [==============================] - 10s 21ms/step - loss: 0.0159 - accuracy:
Epoch 47/50
469/469 [==============================] - 10s 21ms/step - loss: 0.0175 - accuracy:
Epoch 48/50
469/469 [==============================] - 10s 21ms/step - loss: 0.0155 - accuracy:
Epoch 49/50
469/469 [==============================] - 10s 21ms/step - loss: 0.0143 - accuracy:
Epoch 50/50
469/469 [==============================] - 10s 21ms/step - loss: 0.0161 - accuracy:
```

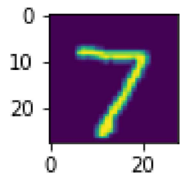
```python
model.save('final_mnist_cnn.h5')
```

```python
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('eposch')
plt.legend(['train','validation'], loc = 'upper left')
```

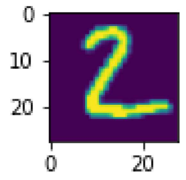
```
<matplotlib.legend.Legend at 0x7fa2c0671350>
```



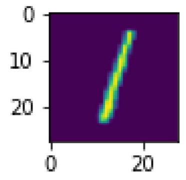
```
y_pred=model.predict(x_test)
for i in range(9):
  plt.subplot(330+i+1)
  plt.imshow(x[i])
  plt.show()
  print(np.round(y_pred[i]))
```



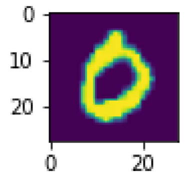
```
[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
```



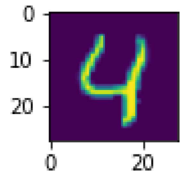
```
[0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
```



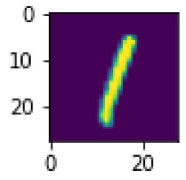
```
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
```



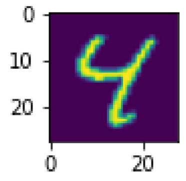
```
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```



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



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



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