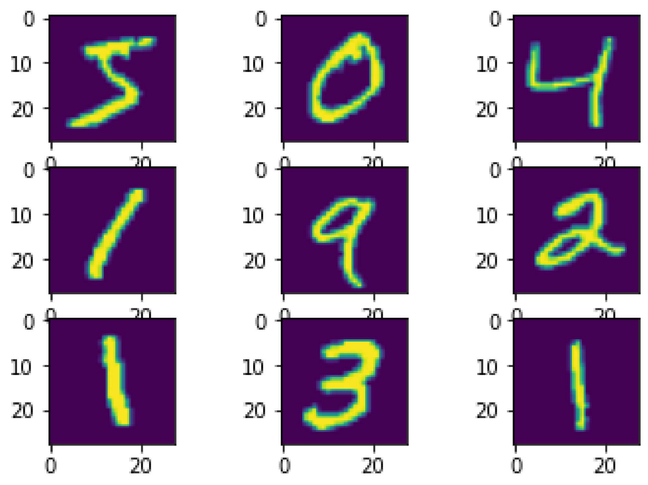
Trương Duy Kha 19146015

https://github.com/TruongDuyKha/Mnist.git

```
from keras.datasets import mnist
import matplotlib.pyplot as plt
from tensorflow.keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Activation, Dropout, Dense, Conv2D, MaxPooling2D
from keras.layers.core.flatten import Flatten
from tensorflow.keras.optimizers import SGD
from keras.models import load_model
import numpy as np
```

```
(x_train, y_train),(x_test,y_test) = mnist.load_data()
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 /= 255
x_test/= 255
```

```
y_train = to_categorical(y_train,10)
y_test = to_categorical(y_test,10)
```

```
Model = Sequential()
Model.add(Conv2D(32,(3,3), activation = 'relu', kernel_initializer= 'he_uniform', padding =
Model.add(Conv2D(32,(3,3),activation = 'relu', kernel_initializer= 'he_uniform',padding = 's
Model.add(MaxPooling2D((2,2)))
Model.add(Conv2D(64,(3,3), activation = 'relu', kernel_initializer= 'he_uniform', padding =
Model.add(Conv2D(64,(3,3),activation = 'relu', kernel_initializer= 'he_uniform',padding = 's
Model.add(MaxPooling2D((2,2)))
Model.add(Conv2D(128,(3,3), activation = 'relu', kernel_initializer= 'he_uniform', padding =
Model.add(Conv2D(128,(3,3),activation = 'relu', kernel_initializer= 'he_uniform',padding = '
Model.add(MaxPooling2D((2,2)))
Model.add(Conv2D(256,(3,3), activation = 'relu', kernel_initializer= 'he_uniform', padding =
Model.add(Conv2D(256,(3,3),activation = 'relu', kernel_initializer= 'he_uniform',padding = '
Model.add(MaxPooling2D((2,2)))
```

```
Model.add(Flatten())
Model.add(Dense(128,activation= 'relu',kernel_initializer='he_uniform'))
Model.add(Dense(10,activation = 'softmax'))
opt = SGD(lr = 0.01,momentum = 0.9)
```

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

```
Model.compile(optimizer= opt,loss = 'categorical_crossentropy', metrics=['accuracy'])
history = Model.fit(x_train,y_train,epochs=20,batch_size = 128,validation_data=(x_test,y_tes
```

```
Epoch 1/20
469/469 [==============================] - 28s 36ms/step - loss: 0.3915 - accuracy: 0.8
Epoch 2/20
469/469 [==============================] - 17s 36ms/step - loss: 0.0513 - accuracy: 0.9
Epoch 3/20
469/469 [==============================] - 16s 34ms/step - loss: 0.0341 - accuracy: 0.9
Epoch 4/20
469/469 [==============================] - 16s 33ms/step - loss: 0.0239 - accuracy: 0.9
Epoch 5/20
469/469 [==============================] - 16s 34ms/step - loss: 0.0173 - accuracy: 0.9
Epoch 6/20
469/469 [==============================] - 16s 34ms/step - loss: 0.0113 - accuracy: 0.9
Epoch 7/20
469/469 [==============================] - 16s 33ms/step - loss: 0.0097 - accuracy: 0.9
Epoch 8/20
469/469 [==============================] - 16s 34ms/step - loss: 0.0087 - accuracy: 0.9
Epoch 9/20
469/469 [==============================] - 16s 34ms/step - loss: 0.0044 - accuracy: 0.9
Epoch 10/20
469/469 [==============================] - 16s 33ms/step - loss: 0.0025 - accuracy: 0.9
Epoch 11/20
469/469 [==============================] - 16s 33ms/step - loss: 0.0021 - accuracy: 0.9
Epoch 12/20
469/469 [==============================] - 16s 33ms/step - loss: 0.0010 - accuracy: 0.9
Epoch 13/20
469/469 [==============================] - 16s 34ms/step - loss: 0.0068 - accuracy: 0.9
Epoch 14/20
469/469 [==============================] - 16s 34ms/step - loss: 0.0023 - accuracy: 0.9
Epoch 15/20
469/469 [==============================] - 16s 34ms/step - loss: 3.7434e-04 - accuracy
Epoch 16/20
469/469 [==============================] - 16s 34ms/step - loss: 1.2379e-04 - accuracy
Epoch 17/20
469/469 [==============================] - 16s 33ms/step - loss: 4.5793e-05 - accuracy
Epoch 18/20
469/469 [==============================] - 16s 34ms/step - loss: 3.1096e-05 - accuracy
Epoch 19/20
469/469 [==============================] - 16s 33ms/step - loss: 2.5818e-05 - accuracy
Epoch 20/20
469/469 [==============================] - 16s 34ms/step - loss: 2.1741e-05 - accuracy
```

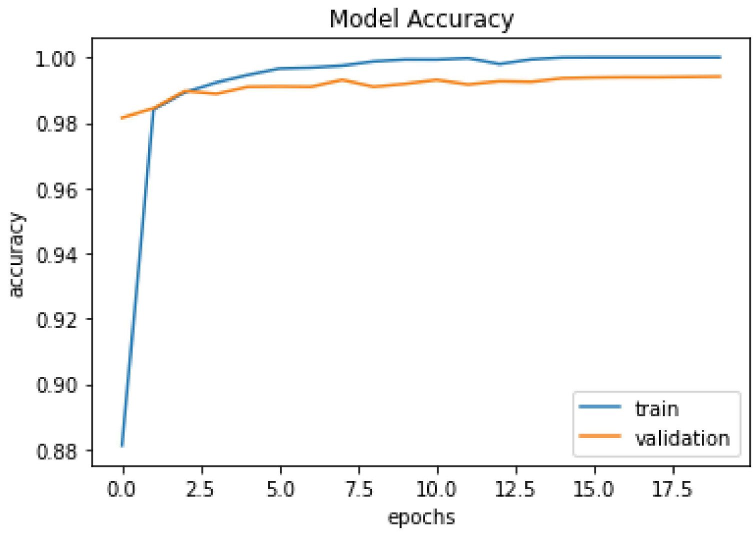
```
Model.save('Mnist_CNN.h5')
model = load_model('Mnist_CNN.h5')
```

```
score = model.evaluate(x_test,y_test,verbose = 1)
y_pred = model.predict(x)
```

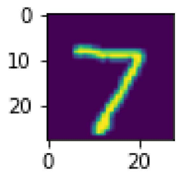
```
313/313 [==============================] - 3s 9ms/step - loss: 0.0286 - accuracy: 0.994
```

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epochs')
plt.legend(['train','validation'],loc='upper_left')
```

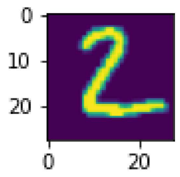
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: MatplotlibDeprecationWa
        best
        upper right
        upper left
        lower left
        lower right
        right
        center left
        center right
        lower center
        upper center
        center
This will raise an exception in 3.3.

<matplotlib.legend.Legend at 0x7fc624b06290>
```



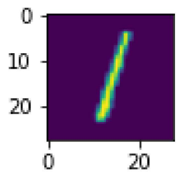
```
import numpy as np
for i in range(9):
  plt.subplot(330 + i + 1)
  plt.imshow(x[i])
  plt.show()
  print(np.argmax(y_pred[0:10],axis = 1)[i])
```


0
10
20
0
20

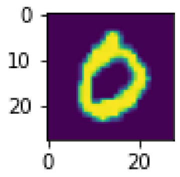
```
7
```


0
10
20
0
20

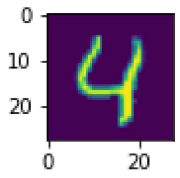
```
2
```


0
10
20
0
20

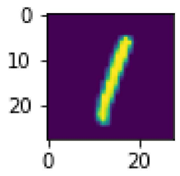
```
1
```


0
10
20
0
20

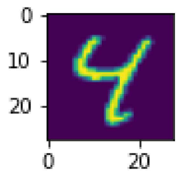
```
0
```


0
10
20
0
20

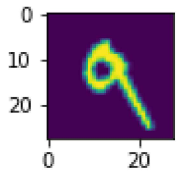
```
4
```


0
10
20
0
20

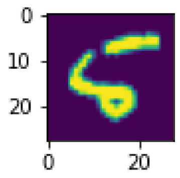
```
1
```


0
10
20
0
20

```
4
```


0
10
20
0
20

```
9
```


0
10
20
0
20

```
5
```