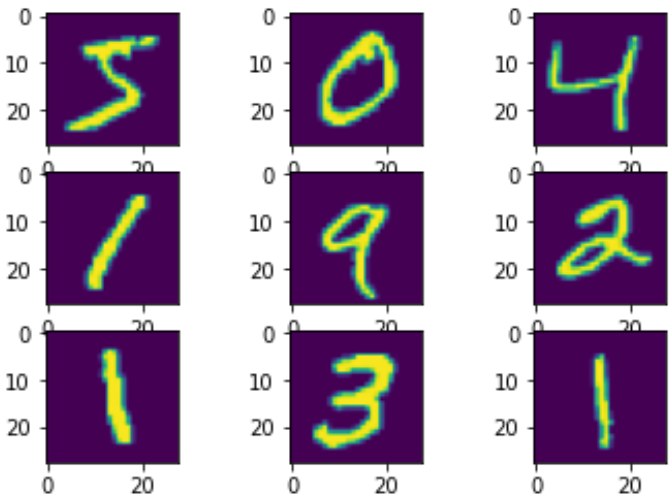
ANN_mnist

```
from keras.datasets import mnist
import matplotlib.pyplot as plt


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

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

```
for i in range(9):
  plt.subplot(330+i+1)    # 330 mean: 3 hang 3 cot
  plt.imshow(x_train[i])
plt.show()
```



```
x = x_test
x_train = x_train.reshape(60000,784)
x_test = x_test.reshape(10000,784)
x_train = x_train.astype('float32')/255
x_test = x_test.astype('float32')/255


from tensorflow.keras.utils import to_categorical
y_train = to_categorical(y_train,10)
y_test = to_categorical(y_test,10)


from keras.models import Sequential
from keras.layers import Activation, Dropout, Dense
from tensorflow.keras.optimizers import RMSprop


model = Sequential()
model.add(Dense(512,activation='relu',input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512,activation='relu'))
model.add(Dropout(0.2))
```

```
model.add(Dense(10,activation='softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 512)               401920

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 512)               262656

 dropout_1 (Dropout)         (None, 512)               0

 dense_2 (Dense)             (None, 10)                5130

=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(loss='categorical_crossentropy', optimizer=RMSprop(), metrics=['accuracy'])
```

```
history = model.fit(x_train, y_train, batch_size=128, epochs=100, verbose=1, validation_data=(x_t
```

```
Epoch 1/100
469/469 [==============================] - 4s 8ms/step - loss: 0.1028 - accuracy: 0.9687
Epoch 2/100
469/469 [==============================] - 3s 6ms/step - loss: 0.0760 - accuracy: 0.9768
Epoch 3/100
469/469 [==============================] - 3s 6ms/step - loss: 0.0596 - accuracy: 0.9819
Epoch 4/100
469/469 [==============================] - 3s 6ms/step - loss: 0.0506 - accuracy: 0.9845
Epoch 5/100
469/469 [==============================] - 3s 6ms/step - loss: 0.0431 - accuracy: 0.9875
Epoch 6/100
469/469 [==============================] - 3s 6ms/step - loss: 0.0377 - accuracy: 0.9886
Epoch 7/100
469/469 [==============================] - 4s 9ms/step - loss: 0.0355 - accuracy: 0.9897
Epoch 8/100
469/469 [==============================] - 4s 8ms/step - loss: 0.0322 - accuracy: 0.9904
Epoch 9/100
469/469 [==============================] - 3s 6ms/step - loss: 0.0290 - accuracy: 0.9916
Epoch 10/100
469/469 [==============================] - 3s 6ms/step - loss: 0.0264 - accuracy: 0.9924
Epoch 11/100
469/469 [==============================] - 3s 6ms/step - loss: 0.0261 - accuracy: 0.9922
Epoch 12/100
469/469 [==============================] - 3s 6ms/step - loss: 0.0222 - accuracy: 0.9933
Epoch 13/100
469/469 [==============================] - 3s 6ms/step - loss: 0.0221 - accuracy: 0.9939
Epoch 14/100
469/469 [==============================] - 3s 6ms/step - loss: 0.0209 - accuracy: 0.9946
Epoch 15/100
```

```
469/469 [==============================] - 3s 6ms/step - loss: 0.0206 - accuracy: 0.9946
Epoch 16/100
469/469 [==============================] - 3s 6ms/step - loss: 0.0196 - accuracy: 0.9950
Epoch 17/100
469/469 [==============================] - 3s 6ms/step - loss: 0.0163 - accuracy: 0.9955
Epoch 18/100
469/469 [==============================] - 3s 6ms/step - loss: 0.0162 - accuracy: 0.9957
Epoch 19/100
469/469 [==============================] - 3s 6ms/step - loss: 0.0171 - accuracy: 0.9954
Epoch 20/100
469/469 [==============================] - 3s 6ms/step - loss: 0.0175 - accuracy: 0.9953
Epoch 21/100
469/469 [==============================] - 3s 6ms/step - loss: 0.0185 - accuracy: 0.9952
Epoch 22/100
469/469 [==============================] - 3s 6ms/step - loss: 0.0161 - accuracy: 0.9957
Epoch 23/100
469/469 [==============================] - 3s 6ms/step - loss: 0.0148 - accuracy: 0.9962
Epoch 24/100
469/469 [==============================] - 3s 6ms/step - loss: 0.0147 - accuracy: 0.9963
Epoch 25/100
469/469 [==============================] - 3s 6ms/step - loss: 0.0141 - accuracy: 0.9963
Epoch 26/100
469/469 [==============================] - 3s 6ms/step - loss: 0.0141 - accuracy: 0.9965
Epoch 27/100
469/469 [==============================] - 3s 6ms/step - loss: 0.0155 - accuracy: 0.9966
Epoch 28/100
469/469 [==============================] - 3s 6ms/step - loss: 0.0129 - accuracy: 0.9966
```

```
model.save("ANN_MNIST.h5")
from tensorflow.keras.models import load_model
model=load_model('ANN_MNIST.h5')
```

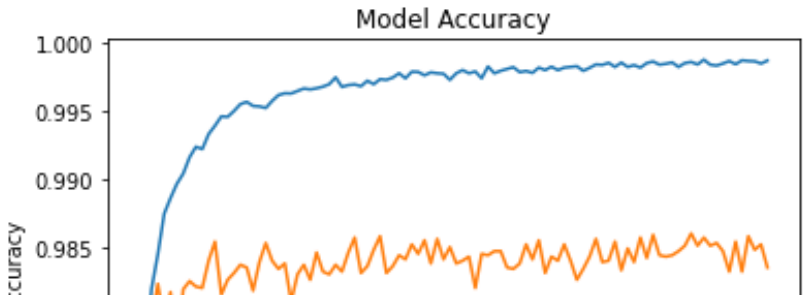
```
score=model.evaluate(x_test, y_test, verbose=1)
print('Test loss =', score[0])
print('Test accuracy =', score[1])
```

```
313/313 [==============================] - 2s 6ms/step - loss: 0.2874 - accuracy: 0.9835
Test loss = 0.28740328550338745
Test accuracy = 0.9835000038146973
```

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

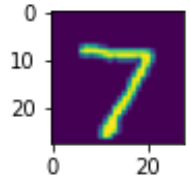
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: MatplotlibDeprecationWarning
        best
        upper right
        upper left
        lower left
        lower right
        right
        center left
        center right
        lower center
        upper center
        center
This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

<matplotlib.legend.Legend at 0x7fbb77788190>
```

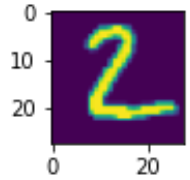



```
import numpy as np
y_pred = model.predict(x_test)
for i in range (9):
  plt.subplot(330+i+1)    # 330 mean: 3 hang 3 cot
  plt.imshow(x[i])
  print(np.round(y_pred[i]))
  plt.show()
```

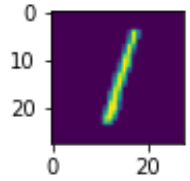
```
[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
```



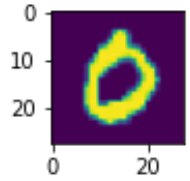
```
[0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
```



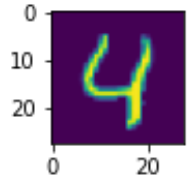
```
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
```

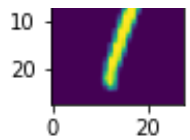


```
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

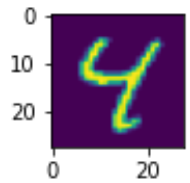


```
[0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
```

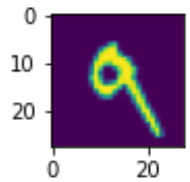




```
[0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
```



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



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