In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
from tensorflow.keras.datasets import mnist
from tensorflow.keras.optimizers import SGD
```

In [3]:
```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 9s 1us/step
```

In [4]:
```
x_train.shape
```

Out[4]:
```
(60000, 28, 28)
```

In [5]:
```
x_test.shape
```

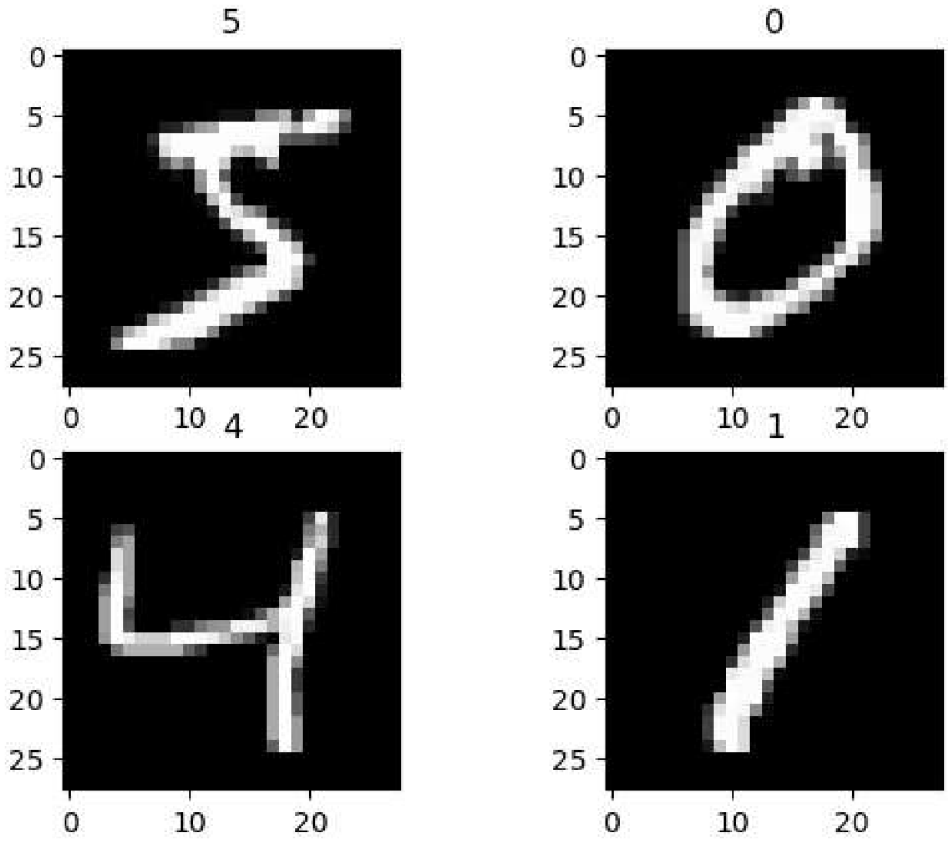
Out[5]:
```
(10000, 28, 28)
```

In [6]:
```
y_train
```

Out[6]:
```
array([5, 0, 4, ..., 5, 6, 8], dtype=uint8)
```

In [7]:
```
plt.subplot(221)
plt.title(y_train[0])
plt.imshow(x_train[0], cmap=plt.get_cmap('gray'))
plt.subplot(222)
plt.title(y_train[1])
plt.imshow(x_train[1], cmap=plt.get_cmap('gray'))
plt.subplot(223)
plt.title(y_train[2])
plt.imshow(x_train[2], cmap=plt.get_cmap('gray'))
plt.subplot(224)
plt.title(y_train[3])
plt.imshow(x_train[3], cmap=plt.get_cmap('gray'))
```

Out[7]:
```
<matplotlib.image.AxesImage at 0x25aed2bcac0>
```



In [8]:
```
num_pixels = x_train.shape[1] * x_train.shape[2]
```

In [9]:
```
num_pixels
```

Out[9]:
```
784
```

In [10]:
```
x_train = x_train.reshape(x_train.shape[0], num_pixels).astype(float)
x_test = x_test.reshape(x_test.shape[0], num_pixels).astype(float)
```

In [11]:
```
x_train.shape
```

Out[11]:
```
(60000, 784)
```

In [12]:
```
x_test.shape
```

Out[12]:
```
(10000, 784)
```

In [13]:
```
# normalize the data
x_train = x_train / 255
x_test = x_test / 255
```

In [14]:
```
x_train
```

Out[14]:
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

In [15]:
```
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
```

In [16]:
```
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

In [17]:
```
y_train.shape
```

Out[17]:
```
(60000, 10)
```

In [18]:
```
def create_model():
    model = Sequential()
    model.add(Dense(num_pixels, input_dim=num_pixels,
                    activation= 'relu' ))
    model.add(Dense(10, activation= 'softmax' ))
    # Compile model
    model.compile(loss= 'categorical_crossentropy' ,
                  optimizer= SGD(), metrics=[ 'accuracy' ])
    return model
```

In [19]:
```
model = create_model()
```

In [20]:
```
model.fit(x_train, y_train, validation_data=(x_test, y_test),
          epochs=10, batch_size = 200)
```

```
Epoch 1/10
300/300 [==============================] - 6s 16ms/step - loss: 1.3190 - accuracy: 0.
7162 - val_loss: 0.7655 - val_accuracy: 0.8485
Epoch 2/10
300/300 [==============================] - 4s 13ms/step - loss: 0.6433 - accuracy: 0.
8571 - val_loss: 0.5220 - val_accuracy: 0.8800
Epoch 3/10
300/300 [==============================] - 4s 13ms/step - loss: 0.4962 - accuracy: 0.
8776 - val_loss: 0.4353 - val_accuracy: 0.8930
Epoch 4/10
300/300 [==============================] - 4s 14ms/step - loss: 0.4319 - accuracy: 0.
8882 - val_loss: 0.3895 - val_accuracy: 0.8999
Epoch 5/10
300/300 [==============================] - 4s 13ms/step - loss: 0.3942 - accuracy: 0.
8947 - val_loss: 0.3607 - val_accuracy: 0.9057
Epoch 6/10
300/300 [==============================] - 4s 14ms/step - loss: 0.3687 - accuracy: 0.
9003 - val_loss: 0.3400 - val_accuracy: 0.9087
Epoch 7/10
300/300 [==============================] - 4s 14ms/step - loss: 0.3498 - accuracy: 0.
9048 - val_loss: 0.3251 - val_accuracy: 0.9124
Epoch 8/10
300/300 [==============================] - 4s 13ms/step - loss: 0.3348 - accuracy: 0.
9078 - val_loss: 0.3124 - val_accuracy: 0.9155
Epoch 9/10
300/300 [==============================] - 4s 13ms/step - loss: 0.3223 - accuracy: 0.
9111 - val_loss: 0.3019 - val_accuracy: 0.9176
Epoch 10/10
300/300 [==============================] - 4s 14ms/step - loss: 0.3118 - accuracy: 0.
9140 - val_loss: 0.2933 - val_accuracy: 0.9196
```

Out[20]:
```
<keras.callbacks.History at 0x25a84074b80>
```

In [21]:
```
model.evaluate(x_test, y_test, batch_size=1)
```

Out[21]:
```
10000/10000 [==============================] - 50s 5ms/step - loss: 0.2933 - accurac
y: 0.9196
[0.2932630777359009, 0.9196000099182129]
```

In [22]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 784)               615440

 dense_1 (Dense)             (None, 10)                7850

=================================================================
Total params: 623,290
Trainable params: 623,290
Non-trainable params: 0
_________________________________________________________________
```

In [23]:
```
new = x_test[45]
```

In [24]:
```
new = new.reshape(1, -1)
```

In [25]:
```
x_train.shape
```

Out[25]:
```
(60000, 784)
```

In [26]:
```
new.shape
```

Out[26]:
```
(1, 784)
```

In [27]:
```
np.argmax(model.predict(new))
```

Out[27]:
```
1/1 [==============================] - 0s 187ms/step
5
```

In [28]:
```
y_test[45]
```

Out[28]:
```
array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.], dtype=float32)
```

In [29]:
```
history = model.fit(x_train, y_train, validation_data=(x_test, y_test),
          epochs=10, batch_size = 200)
```

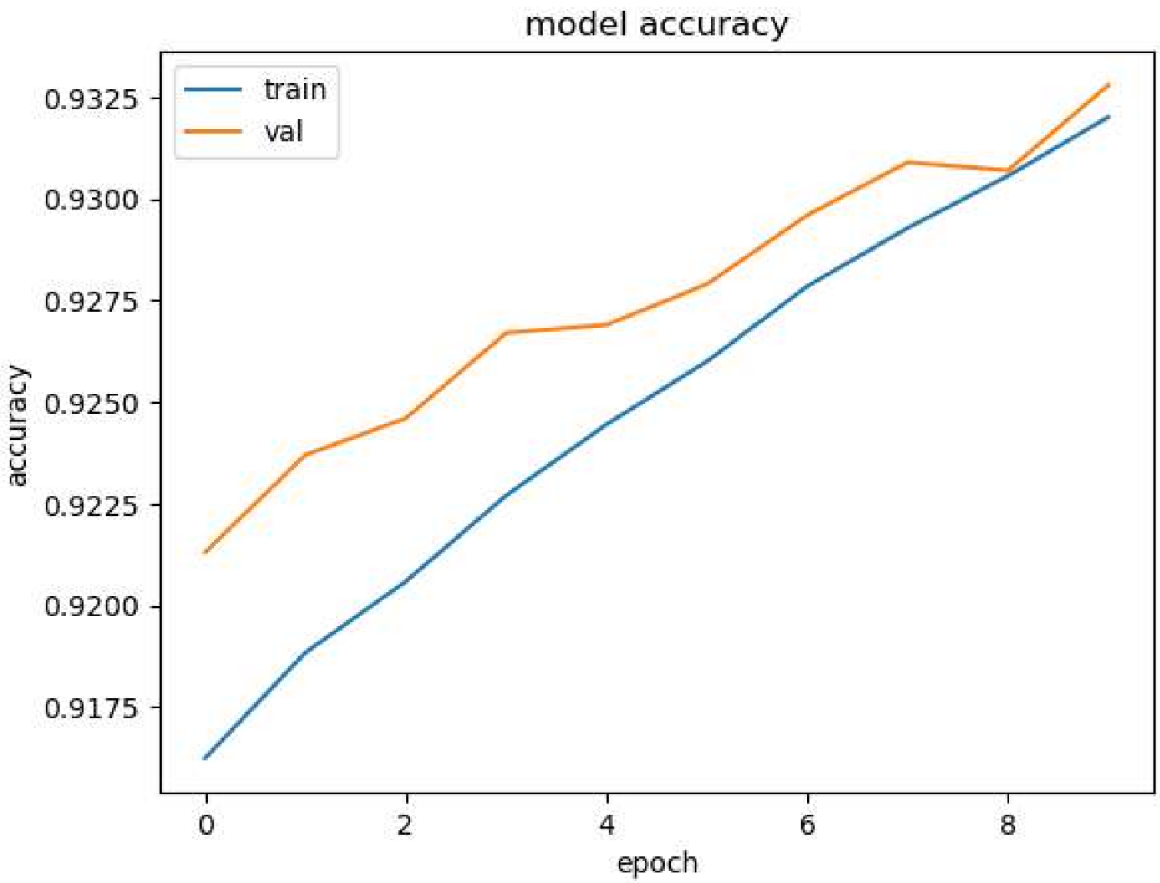
```
Epoch 1/10
300/300 [==============================] - 6s 19ms/step - loss: 0.3026 - accuracy: 0.
9162 - val_loss: 0.2849 - val_accuracy: 0.9213
Epoch 2/10
300/300 [==============================] - 4s 14ms/step - loss: 0.2943 - accuracy: 0.
9188 - val_loss: 0.2781 - val_accuracy: 0.9237
Epoch 3/10
300/300 [==============================] - 4s 14ms/step - loss: 0.2869 - accuracy: 0.
9206 - val_loss: 0.2714 - val_accuracy: 0.9246
Epoch 4/10
300/300 [==============================] - 4s 13ms/step - loss: 0.2801 - accuracy: 0.
9227 - val_loss: 0.2653 - val_accuracy: 0.9267
Epoch 5/10
300/300 [==============================] - 4s 13ms/step - loss: 0.2737 - accuracy: 0.
9244 - val_loss: 0.2601 - val_accuracy: 0.9269
Epoch 6/10
300/300 [==============================] - 4s 13ms/step - loss: 0.2678 - accuracy: 0.
9260 - val_loss: 0.2547 - val_accuracy: 0.9279
Epoch 7/10
300/300 [==============================] - 4s 13ms/step - loss: 0.2623 - accuracy: 0.
9279 - val_loss: 0.2507 - val_accuracy: 0.9296
Epoch 8/10
300/300 [==============================] - 4s 13ms/step - loss: 0.2572 - accuracy: 0.
9293 - val_loss: 0.2455 - val_accuracy: 0.9309
Epoch 9/10
300/300 [==============================] - 4s 13ms/step - loss: 0.2521 - accuracy: 0.
9306 - val_loss: 0.2416 - val_accuracy: 0.9307
Epoch 10/10
300/300 [==============================] - 4s 13ms/step - loss: 0.2475 - accuracy: 0.
9320 - val_loss: 0.2368 - val_accuracy: 0.9328
```

In [30]:

```
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['train', 'val'])
```

Out[30]:

```
<matplotlib.legend.Legend at 0x25a843a80a0>
```


model accuracy
train
val
0.9325
0.9300
0.9275
0.9250
0.9225
0.9200
0.9175
accuracy
0
2
4
6
8
epoch

In [ ]: