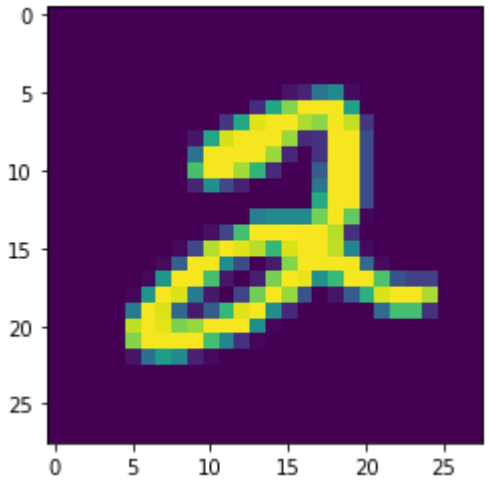
In [1]:
```
# 1. Thêm các thư viện cần thiết
import numpy as np
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
from keras.utils import to_categorical
from keras.datasets import mnist
from keras.utils import np_utils
```

In [2]:
```
# 2. Load dữ liệu MNIST
(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_val, y_val = X_train[50000:60000,:], y_train[50000:60000]
X_train, y_train = X_train[:50000,:], y_train[:50000]
print(X_train.shape)
```

```
(50000, 28, 28)
```

In [41]:
```
plt.imshow(X_train[5])
```

Out[41]:
```
<matplotlib.image.AxesImage at 0x19ff9528940>
```



In [4]:
```
# 3. Reshape lại dữ liệu cho đúng kích thước mà keras yêu cầu
X_train = X_train.reshape(X_train.shape[0], 28, 28, 1)
```

```
X_val = X_val.reshape(X_val.shape[0], 28, 28, 1)
X_test = X_test.reshape(X_test.shape[0], 28, 28, 1)
```

In [5]:

```
# 4. One hot encoding label (Y)
Y_train = np_utils.to_categorical(y_train, 10)
Y_val = np_utils.to_categorical(y_val, 10)
Y_test = np_utils.to_categorical(y_test, 10)
print('Dữ liệu y ban đầu:', y_train[0])
print('Dữ liệu y sau one-hot encoding:',Y_train[0])
```

```
Dữ liệu y ban đầu: 5
Dữ liệu y sau one-hot encoding: [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
```

In [6]:

```
# 5. Định nghĩa model
model = Sequential()

# Thêm Convolutional layer với 32 kernel, kích thước kernel 3*3
# dùng hàm sigmoid làm activation và chỉ rõ input_shape cho layer đầu tiên
model.add(Conv2D(32, (3, 3), activation='sigmoid', input_shape=(28,28,1)))

# Thêm Convolutional layer
model.add(Conv2D(32, (3, 3), activation='sigmoid'))

# Thêm Max pooling layer
model.add(MaxPooling2D(pool_size=(2,2)))

# Flatten layer chuyển từ tensor sang vector
model.add(Flatten())

# Thêm Fully Connected layer với 128 nodes và dùng hàm sigmoid
model.add(Dense(128, activation='sigmoid'))

# Output layer với 10 node và dùng softmax function để chuyển sang xác xuất.
model.add(Dense(10, activation='softmax'))
```

In [7]:

```
# 6. Compile model, chỉ rõ hàm loss_function nào được sử dụng, phương thức dùng để tối ưu hàm loss function.
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

In [8]:

```
# 7. Thực hiện train model với data
H = model.fit(X_train, Y_train, validation_data=(X_val, Y_val),
          batch_size=32, epochs=10, verbose=1)
```

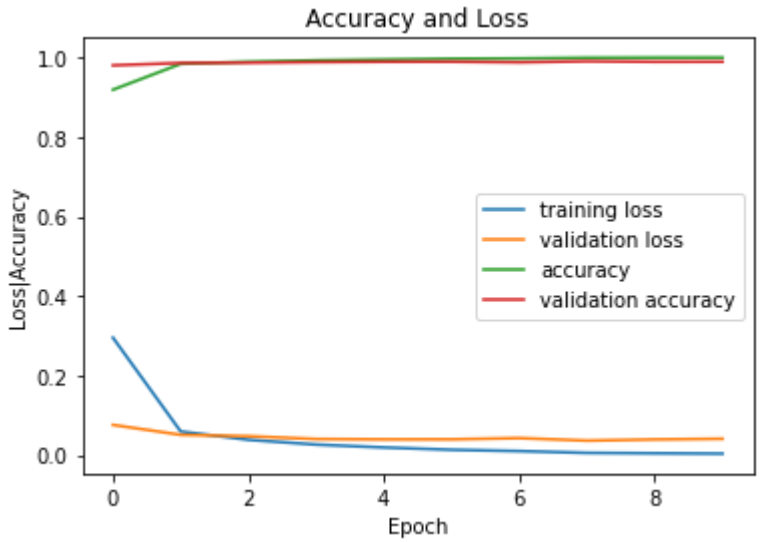
```
Epoch 1/10
1563/1563 [==============================] - 92s 58ms/step - loss: 0.2949 - accuracy: 0.9186 - val_loss: 0.0757 - val_acc
uracy: 0.9799
Epoch 2/10
1563/1563 [==============================] - 99s 63ms/step - loss: 0.0591 - accuracy: 0.9836 - val_loss: 0.0507 - val_acc
uracy: 0.9858
Epoch 3/10
1563/1563 [==============================] - 86s 55ms/step - loss: 0.0382 - accuracy: 0.9890 - val_loss: 0.0473 - val_acc
uracy: 0.9865
Epoch 4/10
1563/1563 [==============================] - 90s 57ms/step - loss: 0.0262 - accuracy: 0.9925 - val_loss: 0.0401 - val_acc
uracy: 0.9881
Epoch 5/10
1563/1563 [==============================] - 97s 62ms/step - loss: 0.0189 - accuracy: 0.9945 - val_loss: 0.0393 - val_acc
uracy: 0.9890
Epoch 6/10
1563/1563 [==============================] - 109s 70ms/step - loss: 0.0131 - accuracy: 0.9966 - val_loss: 0.0394 - val_ac
curacy: 0.9891
Epoch 7/10
1563/1563 [==============================] - 98s 63ms/step - loss: 0.0098 - accuracy: 0.9972 - val_loss: 0.0425 - val_acc
uracy: 0.9876
Epoch 8/10
1563/1563 [==============================] - 94s 60ms/step - loss: 0.0053 - accuracy: 0.9989 - val_loss: 0.0361 - val_acc
uracy: 0.9902
Epoch 9/10
1563/1563 [==============================] - 100s 64ms/step - loss: 0.0043 - accuracy: 0.9992 - val_loss: 0.0390 - val_ac
curacy: 0.9889
Epoch 10/10
1563/1563 [==============================] - 103s 66ms/step - loss: 0.0034 - accuracy: 0.9994 - val_loss: 0.0408 - val_ac
curacy: 0.9890
```

In [9]:

```
# 8. Vẽ đồ thị loss, accuracy của traning set và validation set
fig = plt.figure()
numOfEpoch = 10
plt.plot(np.arange(0, numOfEpoch), H.history['loss'], label='training loss')
plt.plot(np.arange(0, numOfEpoch), H.history['val_loss'], label='validation loss')
plt.plot(np.arange(0, numOfEpoch), H.history['accuracy'], label='accuracy')
plt.plot(np.arange(0, numOfEpoch), H.history['val_accuracy'], label='validation accuracy')
plt.title('Accuracy and Loss')
plt.xlabel('Epoch')
```

```
plt.ylabel('Loss|Accuracy')
plt.legend()
```

Out[9]:
```
<matplotlib.legend.Legend at 0x19ff7bdcd30>
```



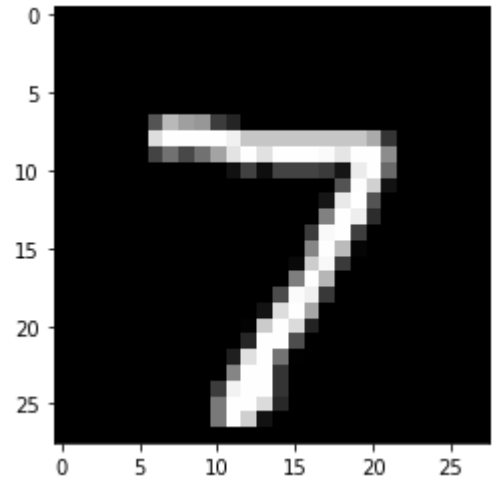
In [10]:
```
# 9. Đánh giá model với dữ liệu test set
score = model.evaluate(X_test, Y_test, verbose=0)
print(score)
```

```
[0.03224751353263855, 0.9901999831199646]
```

In [11]:
```
# 10. Dự đoán ảnh
plt.imshow(X_test[0].reshape(28,28), cmap='gray')

y_predict = model.predict(X_test[0].reshape(1,28,28,1))
print('Giá trị dự đoán: ', np.argmax(y_predict))
```

```
1/1 [==============================] - 0s 101ms/step
Giá trị dự đoán:  7
```



In [ ]: