

```
print("Student Name: Simon")  
print("Student ID: 1930026144")
```

```
Student Name: Simon  
Student ID: 1930026144
```

```
import keras  
keras.__version__
```

```
'2.8.0'
```

```
from keras import layers  
from keras import models  
  
model = models.Sequential()  
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))  
model.add(layers.MaxPooling2D((2, 2)))  
model.add(layers.Conv2D(64, (3, 3), activation='relu'))  
model.add(layers.MaxPooling2D((2, 2)))  
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_1 (Conv2D)	(None, 11, 11, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 64)	0
conv2d_2 (Conv2D)	(None, 3, 3, 64)	36928
=====		
Total params: 55,744		
Trainable params: 55,744		
Non-trainable params: 0		

```
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
```

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_1 (Conv2D)	(None, 11, 11, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 64)	0
conv2d_2 (Conv2D)	(None, 3, 3, 64)	36928
flatten (Flatten)	(None, 576)	0
dense (Dense)	(None, 64)	36928
dense_1 (Dense)	(None, 10)	650
=====		
Total params: 93,322		
Trainable params: 93,322		
Non-trainable params: 0		

```

import numpy as np
import os
import gzip
def load_data(data_folder):
    files = [
        'train-labels-idx1-ubyte.gz', 'train-images-idx3-ubyte.gz',
        't10k-labels-idx1-ubyte.gz', 't10k-images-idx3-ubyte.gz'
    ]
    paths = []
    for fname in files:
        paths.append(os.path.join(data_folder, fname))
    print("Opening y_train path", paths[0])
    with gzip.open(paths[0], 'rb') as lbpath:
        y_train = np.frombuffer(lbpath.read(), np.uint8, offset=8)
    print("Opening x_train path", paths[1])
    with gzip.open(paths[1], 'rb') as imgpath:
        x_train = np.frombuffer(imgpath.read(), np.uint8,
offset=16).reshape(len(y_train), 28, 28)
    print("Opening y_test path", paths[2])
    with gzip.open(paths[2], 'rb') as lbpath:
        y_test = np.frombuffer(lbpath.read(), np.uint8, offset=8)
    print("Opening x_test path", paths[3])
    with gzip.open(paths[3], 'rb') as imgpath:
        x_test = np.frombuffer(imgpath.read(), np.uint8,
offset=16).reshape(len(y_test), 28, 28)
    return (x_train, y_train), (x_test, y_test)

```

```

local_folder = "mnist_dataset"
(train_images, train_labels), (test_images, test_labels) =
load_data(local_folder)

```

```

Opening y_train path mnist_dataset/train-labels-idx1-ubyte.gz
Opening x_train path mnist_dataset/train-images-idx3-ubyte.gz
Opening y_test path mnist_dataset/t10k-labels-idx1-ubyte.gz
Opening x_test path mnist_dataset/t10k-images-idx3-ubyte.gz

```

```
# from keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
import requests

# (train_images, train_labels), (test_images, test_labels) = mnist.load_data()

train_images = train_images.reshape((60000, 28, 28, 1))
train_images = train_images.astype('float32') / 255

test_images = test_images.reshape((10000, 28, 28, 1))
test_images = test_images.astype('float32') / 255

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
```

```
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=5, batch_size=1)
```

Epoch 1/5

2022-11-05 21:14:22.369766: W tensorflow/core/platform/profile_utils/cpu_utils.cc

```
60000/60000 [=====] - 67s 1ms/step - loss: 0.2044 - accu
Epoch 2/5
60000/60000 [=====] - 64s 1ms/step - loss: 0.2509 - accu
Epoch 3/5
60000/60000 [=====] - 64s 1ms/step - loss: 0.2675 - accu
Epoch 4/5
60000/60000 [=====] - 64s 1ms/step - loss: 0.2844 - accu
Epoch 5/5
60000/60000 [=====] - 64s 1ms/step - loss: 0.3045 - accu
```

<keras.callbacks.History at 0x14894cf10>

Let's evaluate the model on the test data:

```
test_loss, test_acc = model.evaluate(test_images, test_labels)
```

```
313/313 [=====] - 1s 2ms/step - loss: 0.2841 - accuracy:
```

```
test_acc
```

```
0.982200026512146
```