THỰC HÀNH 4 : CONVOLUTIONAL NEURAL NETWORK

Sau bài thực hành này, sinh viên có khả năng:

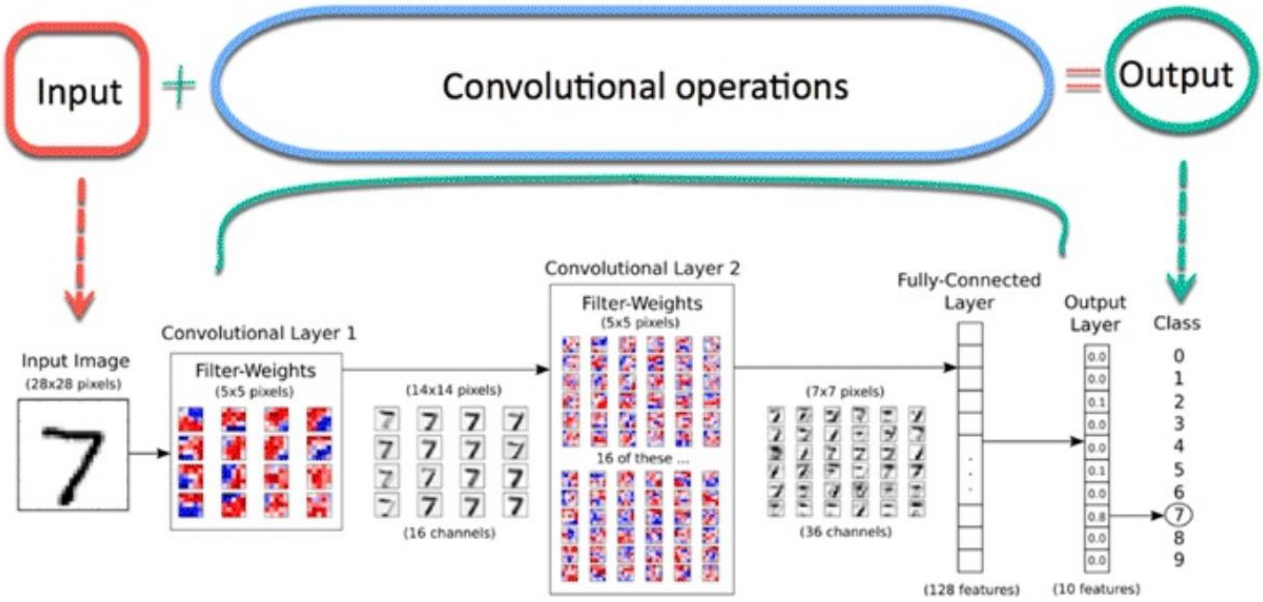
- Xây dựng được kiến trúc Convolutional Neural Network (CNN)
- Cài đặt mô hình Convolutional Neural Network bằng tensorflow
- Cài đặt ứng dụng thực tế sử dụng Convolutional Neural Network

1. GIỚI THIỆU

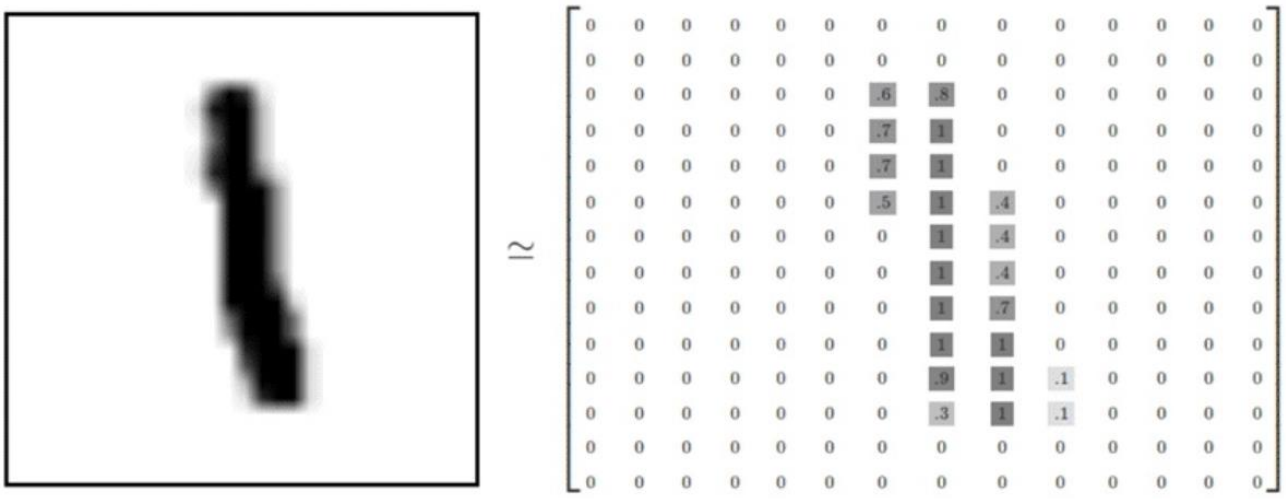
Convolutional Artificial Neural Network (Convnets hoặc CNN) là một mô hình nổi tiếng trong lĩnh vực thị giác máy tính. Kiến trúc này dùng nhận dạng đối tượng trong ảnh hoặc video.

Nguyên lý hoạt động CNN

- Một ảnh được đưa vào CNN gọi là ảnh đầu vào
- Ảnh đầu vào sẽ được các phép toán chập xử lý
- Đầu ra là nhãn được CNN dự báo



Một ảnh là ma trận pixel có chiều cao và chiều rộng. Mỗi pixel có giá trị [0, 255]. Ảnh trắng đen chỉ có 1 kênh màu, ảnh màu có 3 kênh. Số 0 là màu trắng, số 1 là màu đen.



1.1. Convolutional Operation

Thành phần quan trọng nhất là tầng convolutional. Thành phần này nhằm mục đích giảm kích thước của ảnh để tính toán weight nhanh hơn và cải thiện việc dự báo.

Trong quá trình thực hiện chập, CNN giữ đặc trưng của ảnh và loại bỏ tạo âm. Ví dụ, CNN học cách nhận biết một con voi có phía sau là núi đồi bằng cách trích xuất các pixel liên quan đến con voi.

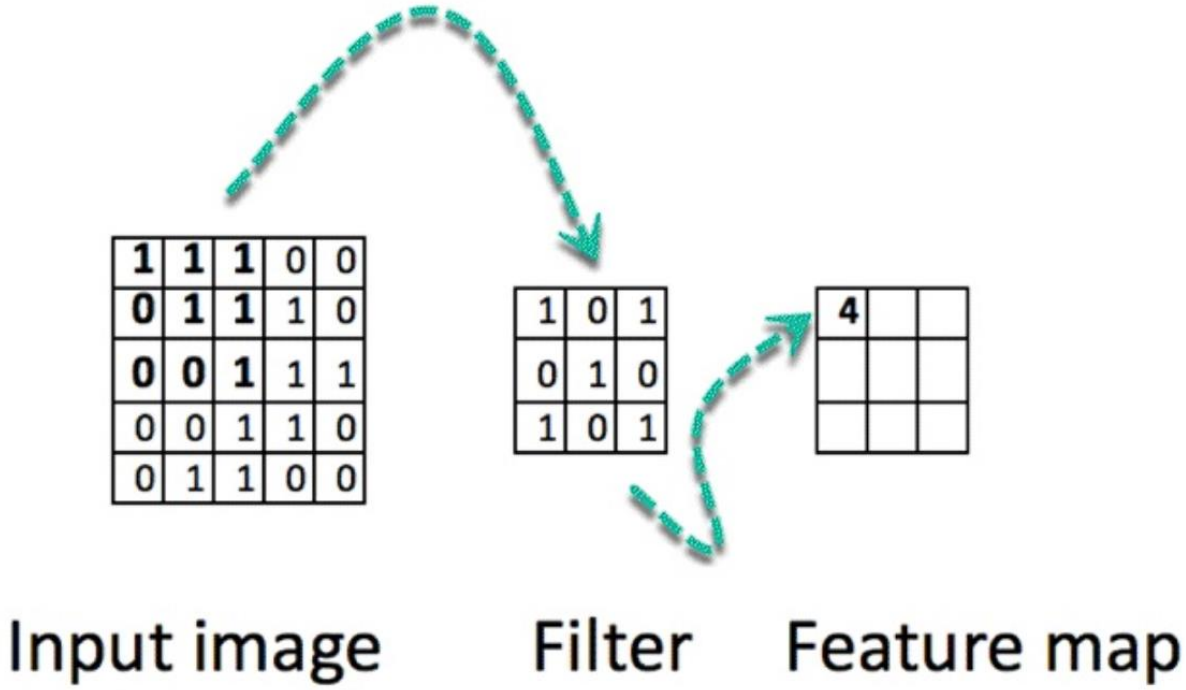
1.2. Các thành phần của Convnet

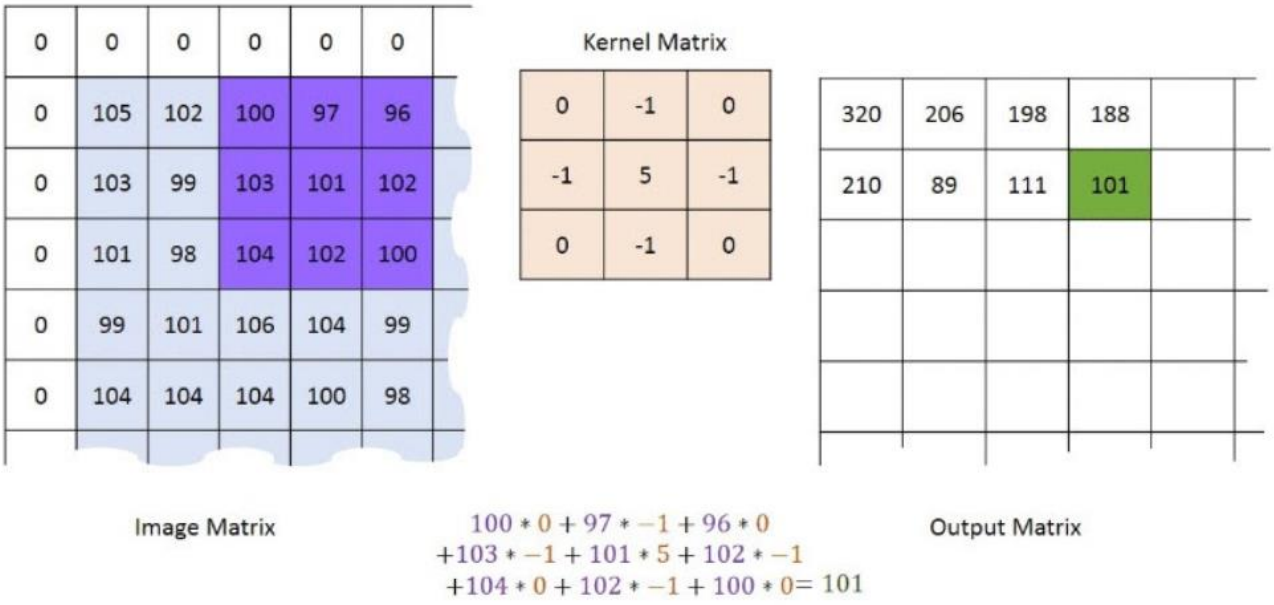
- Convolution
- Non Linearity (ReLU)
- Pooling or Sub sampling
- Classification (Dense / Fully Connected Layer)

1.2.1. Thành phần convolutional (chập)

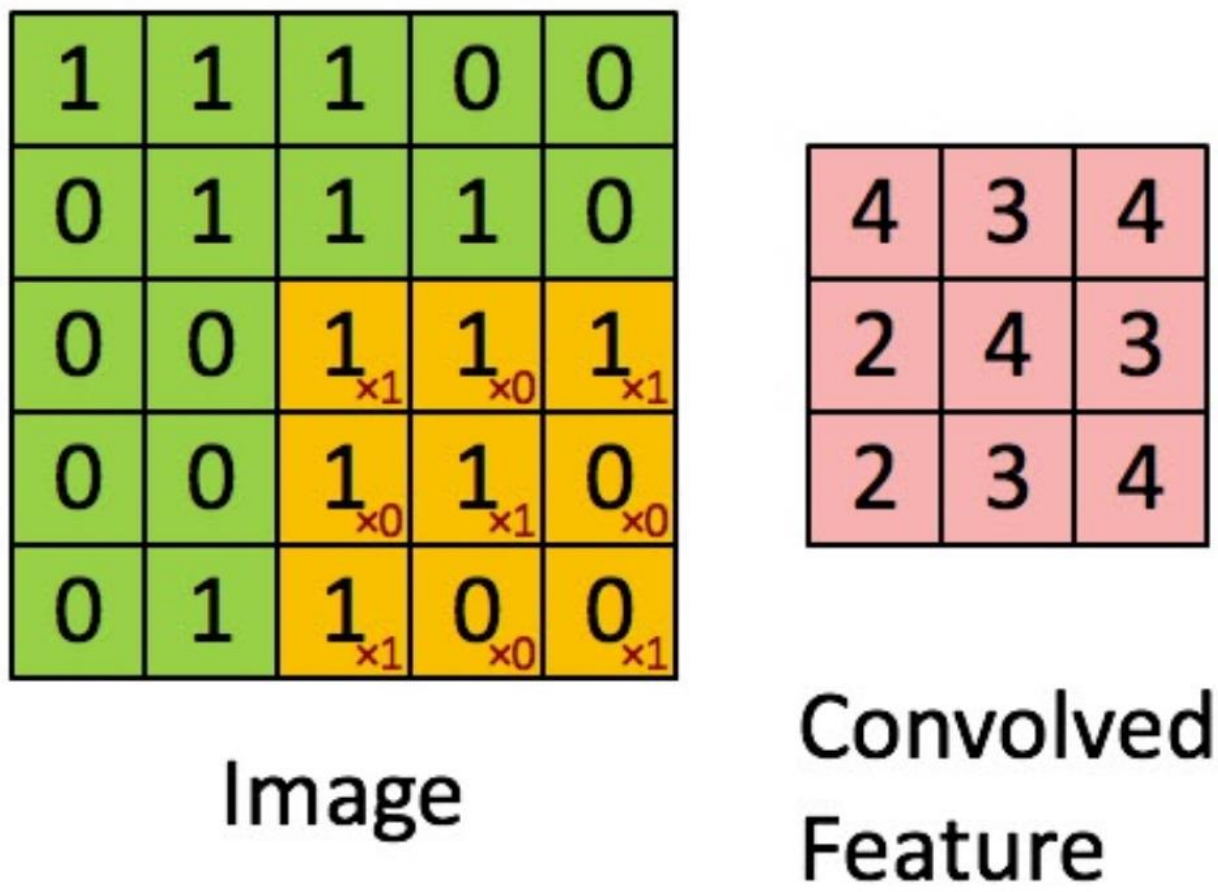
Mục đích của tầng convolution là rút đặc trưng của đối tượng trong ảnh. CNN học các mẫu đặc biệt có thể nhận dạng khắp nơi trong ảnh.

Convolution là phép nhân element-wise hai ma trận là input image và bộ lọc (3 x 3) để cho ra feature map.





Convolution with horizontal and vertical strides = 1

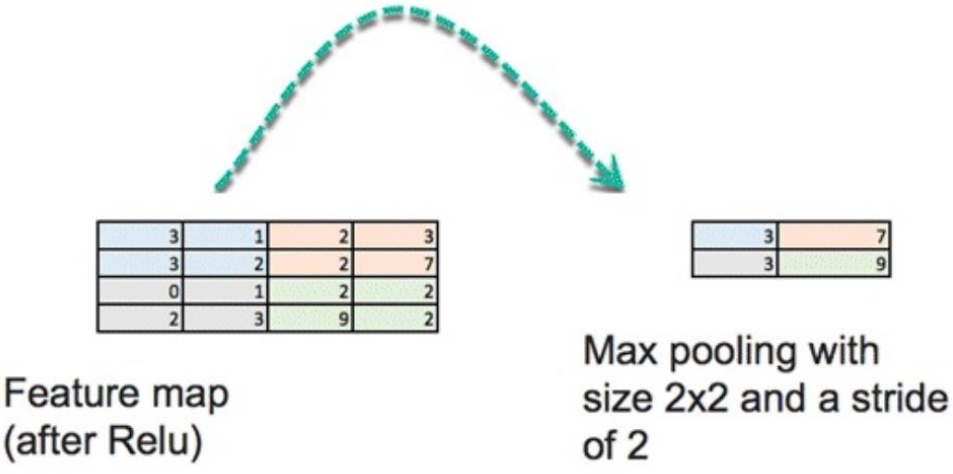


1.2.2. Non Linearity (ReLU)

Sau khi thực hiện tính toán chập, đầu ra là feature map sẽ chuyển đến một hàm activation để tính non linearity. Thông thường là hàm ReLU (Các pixel âm sẽ bị thay thế bởi số 0).

1.2.3. Max pool operation

Bước này nhằm làm giảm chiều của ảnh đầu vào. CNN sẽ giảm việc tính weight, hạn chế overfitting. Bước này thường dùng ma trận 2 x 2 và hàm max để lấy giá trị lớn nhất trong ma trận.



1.2.4. Fully connected layers

Liên kết tất cả các neurons từ tầng trước và tầng kế tiếp. Dùng hàm softmax để phân lớp cho ảnh đầu vào.

2. CÀI ĐẶT CNN

2.1. Nạp thư viện

```
import pandas as pd
import numpy as np
import tensorflow as tf
import keras
from keras.layers import Conv2D, Dense, MaxPool2D, Flatten
from keras.models import Sequential
from keras import Input
#data visualization packages
import matplotlib.pyplot as plt
```

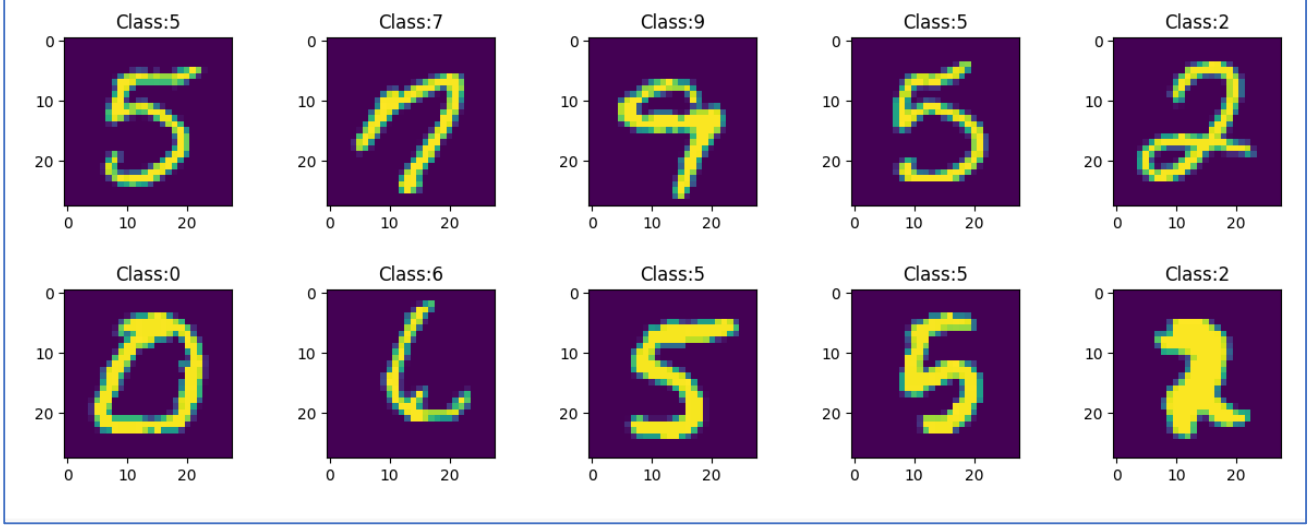
2.2. Đọc dữ liệu MNIST

```
[2] mnist_train = '/content/sample_data/mnist_train_small.csv'
    mnist_test = '/content/sample_data/mnist_test.csv'
    df_train = pd.read_csv(mnist_train)
    df_test = pd.read_csv(mnist_test)

    print(df_train.shape)
    X_train = df_train.iloc[:,1:]
    y_train = df_train.iloc[:,0]
    X_test = df_test.iloc[:,1:]
    y_test = df_test.iloc[:,0]
    print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)

    (19999, 785)
    (19999, 784) (19999,) (9999, 784) (9999,)
```

```
num_classes = 10
input_shape = (28, 28, 1)
X_train = X_train.astype("float32")/255
X_test = X_test.astype("float32")/255
X_train = X_train.to_numpy()
X_train = X_train.reshape(X_train.shape[0], 28, 28, 1)
print(X_train.shape)
X_test = X_test.to_numpy()
X_test = X_test.reshape(X_test.shape[0], 28, 28, 1)
print(X_test.shape)


print('Number of classes:', len(np.unique(y_train)))
print('Classes:', np.unique(y_train))
print(y_train[:5])
```

2.3. Trực quan hóa 10 ảnh đầu tiên

```
#Data visualization
fig, axes = plt.subplots(nrows=2, ncols=5,figsize=(15, 5))
ax = axes.ravel()
for i in range(10):
  ax[i].imshow(X_train[i].reshape(28, 28))
  ax[i].title.set_text('Class:'+str(y_train[i]))
plt.subplots_adjust(hspace=0.5)
plt.show()
```



2.4. Chuyển giá trị nhãn thành one-hot encoding

```
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

2.5. Xây dựng mô hình CNN

```
model = Sequential()
model.add(Input(shape=input_shape))
model.add(Conv2D(32, kernel_size=(3,3), activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Conv2D(64, kernel_size=(3,3), activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(units=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 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 flatten (Flatten)           (None, 1600)              0

 dense (Dense)               (None, 10)                16010

=================================================================
Total params: 34,826
Trainable params: 34,826
Non-trainable params: 0
_________________________________________________________________
```

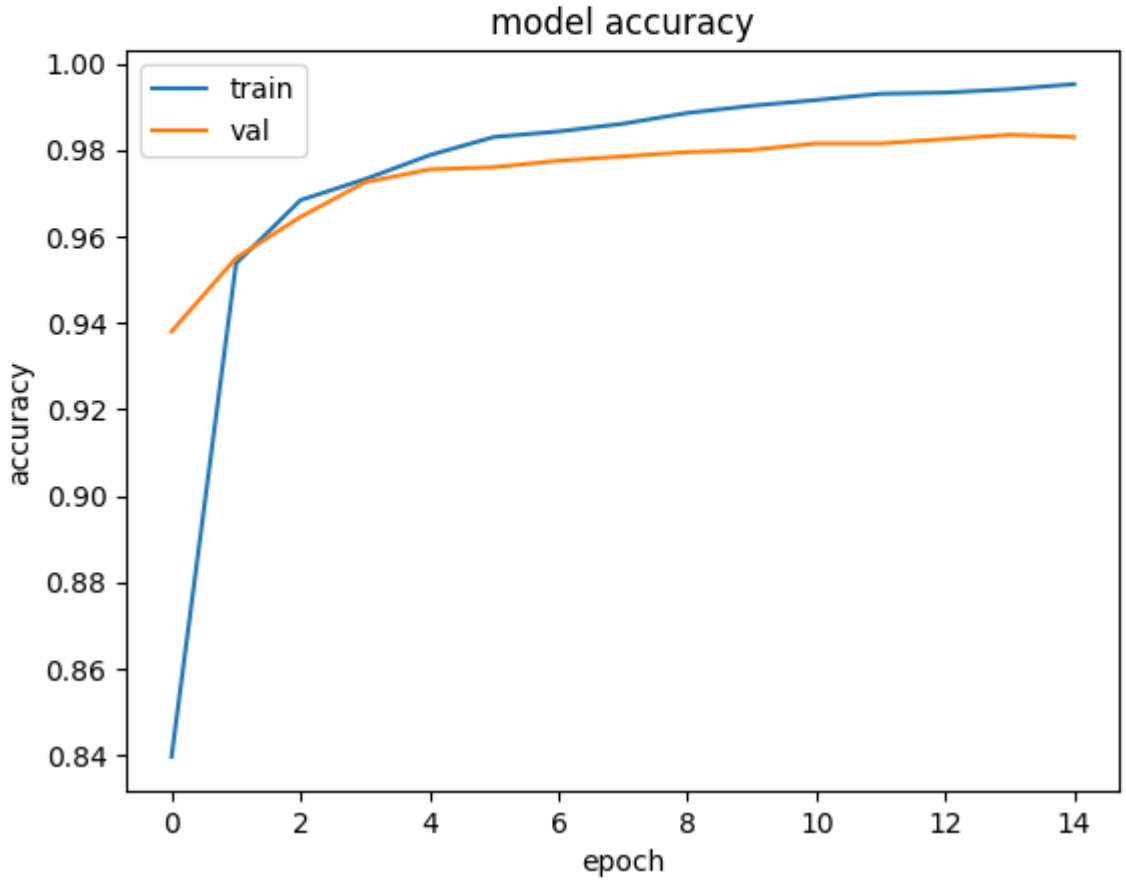
2.6. Huấn luyện mô hình CNN

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model_fit = model.fit(X_train, y_train, batch_size=128, epochs=15, validation_split=0.1, verbose=1)
```

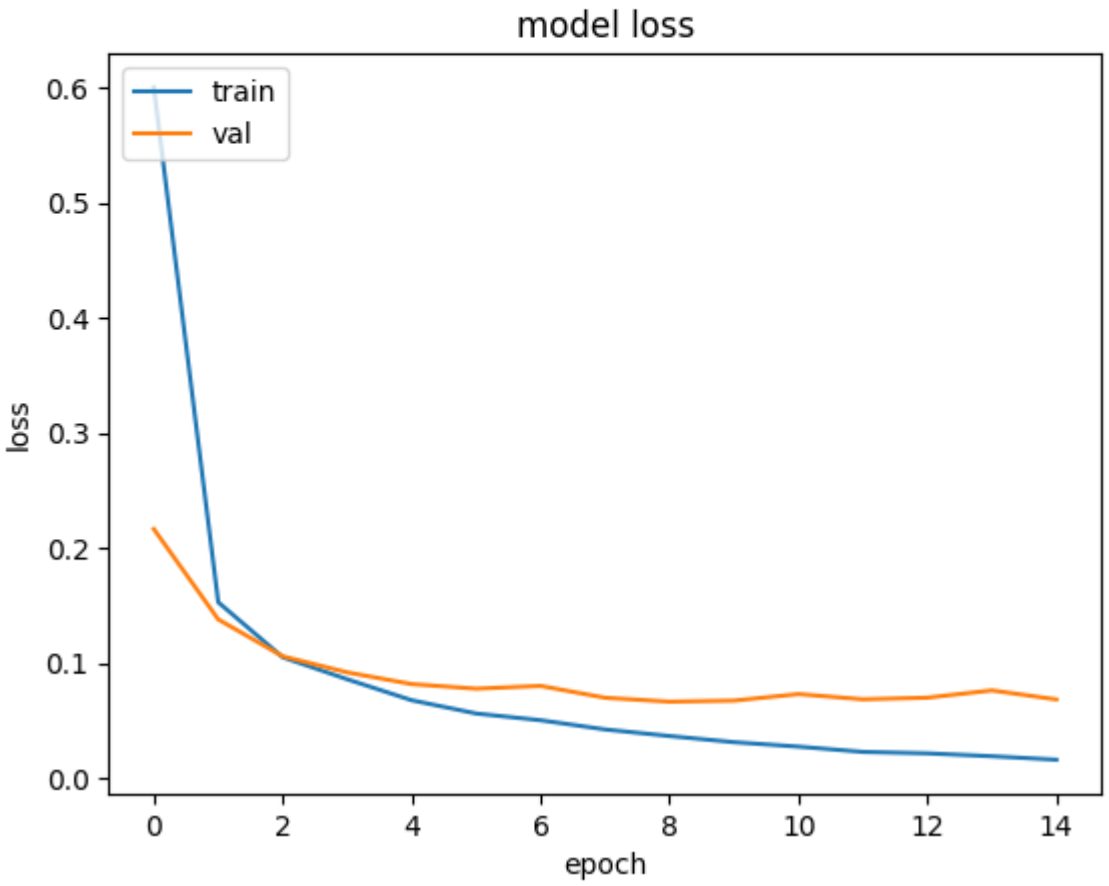
```
Epoch 1/15
141/141 [==============================] - 18s 121ms/step - loss: 0.6006 - accuracy: 0.8396 - val_loss: 0.2166 - val_accuracy: 0.9380
Epoch 2/15
141/141 [==============================] - 16s 112ms/step - loss: 0.1529 - accuracy: 0.9537 - val_loss: 0.1380 - val_accuracy: 0.9550
Epoch 3/15
141/141 [==============================] - 16s 115ms/step - loss: 0.1051 - accuracy: 0.9684 - val_loss: 0.1059 - val_accuracy: 0.9645
Epoch 4/15
141/141 [==============================] - 16s 114ms/step - loss: 0.0860 - accuracy: 0.9732 - val_loss: 0.0919 - val_accuracy: 0.9725
Epoch 5/15
141/141 [==============================] - 17s 119ms/step - loss: 0.0677 - accuracy: 0.9788 - val_loss: 0.0818 - val_accuracy: 0.9755
Epoch 6/15
141/141 [==============================] - 16s 114ms/step - loss: 0.0562 - accuracy: 0.9830 - val_loss: 0.0778 - val_accuracy: 0.9760
Epoch 7/15
141/141 [==============================] - 15s 109ms/step - loss: 0.0503 - accuracy: 0.9843 - val_loss: 0.0802 - val_accuracy: 0.9775
Epoch 8/15
141/141 [==============================] - 16s 111ms/step - loss: 0.0423 - accuracy: 0.9861 - val_loss: 0.0699 - val_accuracy: 0.9785
Epoch 9/15
141/141 [==============================] - 15s 109ms/step - loss: 0.0366 - accuracy: 0.9886 - val_loss: 0.0665 - val_accuracy: 0.9795
Epoch 10/15
141/141 [==============================] - 16s 112ms/step - loss: 0.0313 - accuracy: 0.9902 - val_loss: 0.0674 - val_accuracy: 0.9800
Epoch 11/15
141/141 [==============================] - 17s 121ms/step - loss: 0.0274 - accuracy: 0.9916 - val_loss: 0.0731 - val_accuracy: 0.9815
Epoch 12/15
141/141 [==============================] - 16s 110ms/step - loss: 0.0228 - accuracy: 0.9930 - val_loss: 0.0685 - val_accuracy: 0.9815
Epoch 13/15
141/141 [==============================] - 16s 110ms/step - loss: 0.0216 - accuracy: 0.9933 - val_loss: 0.0700 - val_accuracy: 0.9825
Epoch 14/15
141/141 [==============================] - 15s 110ms/step - loss: 0.0191 - accuracy: 0.9941 - val_loss: 0.0763 - val_accuracy: 0.9835
Epoch 15/15
141/141 [==============================] - 15s 108ms/step - loss: 0.0160 - accuracy: 0.9952 - val_loss: 0.0684 - val_accuracy: 0.9830
```

2.7. Đánh giá mô hình CNN

```
plt.plot(model_fit.history['accuracy'])
plt.plot(model_fit.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```



```
plt.plot(model_fit.history['loss'])
plt.plot(model_fit.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```



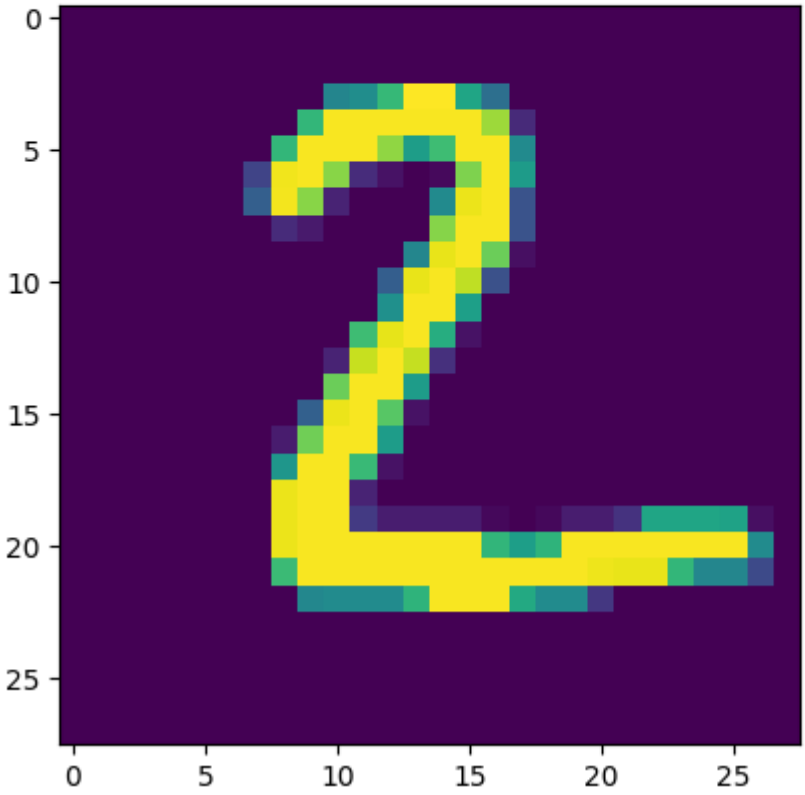
2.8. Đánh giá mô hình CNN thử nghiệm trên tập test

```
[10] score = model.evaluate(X_test, y_test, verbose=2)
     print('Test loss:', score[0])
     print('Test accuracy:', score[1])

     313/313 - 2s - loss: 0.0526 - accuracy: 0.9849 - 2s/epoch - 8ms/step
     Test loss: 0.05255042016506195
     Test accuracy: 0.9848985075950623
```

2.9. Dự báo nhãn cho ảnh sử dụng CNN

```
[16] predict = model.predict(X_test[:1])
     print(predict)
     print(np.argmax(predict), np.argmax(y_test[0]))
     plt.imshow(X_test[:1].reshape(28, 28))
     plt.show()
```

```
1/1 [==============================] - 0s 23ms/step
[[1.6635391e-05 9.1294016e-05 9.9989188e-01 6.6101119e-10 1.1176609e-17
  4.1269346e-14 1.1087320e-07 4.1280837e-18 1.9112152e-09 1.7522592e-19]]
2 2
```



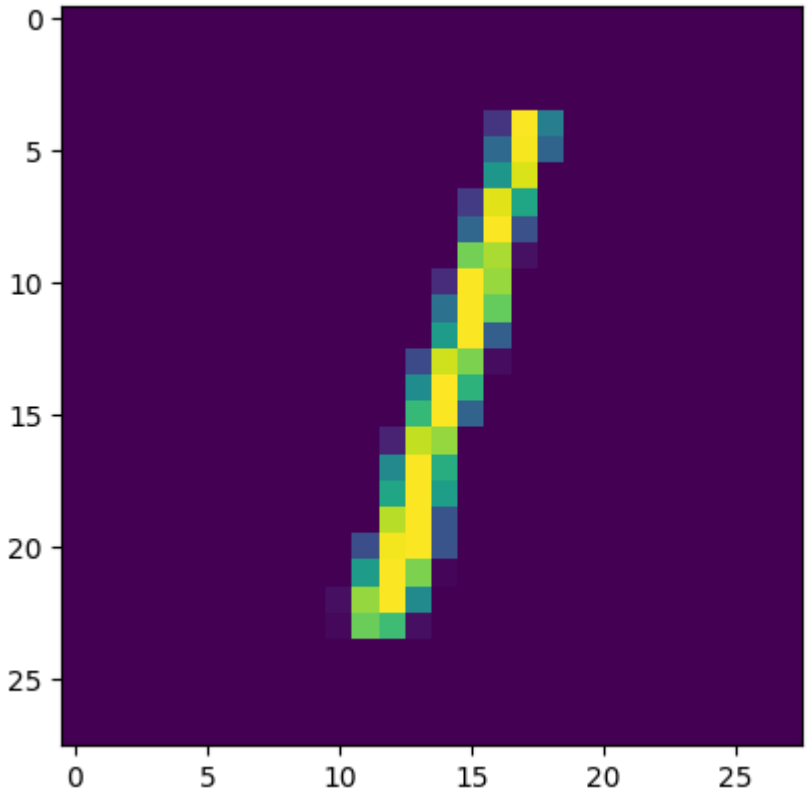
2.10. Lưu tham số mô hình CNN

```
model.save_weights('cnn.h5')
```

2.11. Nạp lại mô hình và tham số

```
model = Sequential()
model.add(Input(shape=input_shape))
model.add(Conv2D(32, kernel_size=(3,3), activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Conv2D(64, kernel_size=(3,3), activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(units=10, activation='softmax'))
model.load_weights('cnn.h5')

predict = model.predict(X_test[1:2])
print(predict)
print(np.argmax(predict), np.argmax(y_test[1]))
plt.imshow(X_test[1:2].reshape(28, 28))
plt.show()
```

```
1/1 [==============================] - 0s 75ms/step
[[4.24157385e-08 9.99622226e-01 1.08659715e-05 1.96134366e-07
  3.32227850e-04 2.13549512e-07 2.66307325e-07 1.79432373e-05
  1.59877309e-05 1.65863394e-08]]
1 1
```



3. BÀI TẬP

1. Viết chương trình cài đặt CNN để nhận dạng ảnh trên bộ dataset CIFAR10 có sẵn trong tensorflow với các nhãn sau

Label	Description
0	airplane
1	automobile
2	bird
3	cat
4	deer
5	dog
6	frog
7	horse
8	ship
9	truck

2. Hãy viết chương trình cài đặt CNN để nhận dạng ảnh Cat hoặc Dog. Dữ liệu do giảng viên cung cấp

3. Hãy viết chương trình cài đặt CNN để nhận dạng ảnh Fashion-MNIST. Dữ liệu do giảng viên cung cấp.

4. Hãy viết chương trình cài đặt CNN để nhận dạng ảnh khuôn mặt {Nam, Nữ}. Dữ liệu do giảng viên cung cấp.

5. Triển khai câu 1 - 4 trên nền tảng WEB sử dụng Flask