



CSE: Faculty of Computer Science and Engineering

Thuyloi University

Mạng nơ ron tích chập

Convolutional Neural Network - CNN

TS. Nguyễn Thị Kim Ngân



Nội dung

- Tầng tích chập (convolutional layer)
 - Phép tích chập cho ảnh
 - Độm và bước sải
 - Đa kênh đầu vào/ra
- Tầng gộp (pooling layer)
 - Phép gộp
 - Độm và bước sải
 - Đa kênh đầu vào
- Tầng kết nối đầy đủ (Fully connected layer)
- Mạng Nơ-ron tích chập LeNet



Nội dung

- **Tầng tích chập (convolutional layer)**
 - Phép tích chập cho ảnh
 - Độm và bước sải
 - Đa kênh đầu vào/ra
- Tầng gộp (pooling layer)
 - Phép gộp
 - Độm và bước sải
 - Đa kênh đầu vào
- Tầng kết nối đầy đủ (Fully connected layer)
- Mạng Nơ-ron tích chập LeNet

Toán tử Tương quan Chéo

Đầu vào

0	1	2
3	4	5
6	7	8

*

Bộ lọc

0	1
2	3

=

Đầu ra

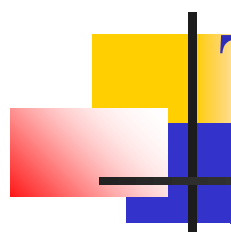
19	25
37	43

$$0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19$$

$$1 \times 0 + 2 \times 1 + 4 \times 2 + 5 \times 3 = 25$$

$$3 \times 0 + 4 \times 1 + 6 \times 2 + 7 \times 3 = 37$$

$$4 \times 0 + 5 \times 1 + 7 \times 2 + 8 \times 3 = 43$$



Tầng tích chập (convolutional layer)

Trong một tầng tích chập,

- Input: mảng đầu vào (X), mảng hạt nhân tương quan (K) được kết hợp
- Output: $X \otimes K$ (\otimes : phép toán tương quan chéo)

Toán tử tương quan chéo

Đầu vào		Bộ lọc		Đầu ra																	
<table><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>3</td><td>4</td><td>5</td></tr><tr><td>6</td><td>7</td><td>8</td></tr></table>	0	1	2	3	4	5	6	7	8	*	<table><tr><td>0</td><td>1</td></tr><tr><td>2</td><td>3</td></tr></table>	0	1	2	3	=	<table><tr><td>19</td><td>25</td></tr><tr><td>37</td><td>43</td></tr></table>	19	25	37	43
0	1	2																			
3	4	5																			
6	7	8																			
0	1																				
2	3																				
19	25																				
37	43																				
3 x 3		2 x 2		$(3-2+1) \times (3-2+1) = 2 \times 2$																	

Kích thước đầu vào: $H \times W$

Kích thước của bộ lọc tích chập: $h \times w$

Kích thước đầu ra: $(H - h + 1) \times (W - w + 1)$

Toán tử tương quan chéo

```
from mxnet import autograd, np, npx
from mxnet.gluon import nn
npx.set_np()

# Saved in the d2l package for later use
def corr2d(X, K):
    """Compute 2D cross-correlation."""
    h, w = K.shape
    Y = np.zeros((X.shape[0] - h + 1, X.shape[1] - w + 1))
    for i in range(Y.shape[0]):
        for j in range(Y.shape[1]):
            Y[i, j] = (X[i: i + h, j: j + w] * K).sum()
    return Y
```

Đầu vào

0	1	2
3	4	5
6	7	8

*

Bộ lọc

0	1
2	3

=

Đầu ra

19	25
37	43

```
X = np.array([[0, 1, 2], [3, 4, 5], [6, 7, 8]])
K = np.array([[0, 1], [2, 3]])
corr2d(X, K)
```



Ví dụ

Lập trình tầng tích chập 2 chiều

```
class Conv2D(nn.Block):  
    def __init__(self, kernel_size, **kwargs):  
        super(Conv2D, self).__init__(**kwargs)  
        self.weight = self.params.get('weight', shape=kernel_size)  
        self.bias = self.params.get('bias', shape=(1,))  
  
    def forward(self, x):  
        return corr2d(x, self.weight.data()) + self.bias.data()
```


Ví dụ

Phát hiện Biên của Vật thể trong Ảnh

Bức ảnh có kích thước là 6×8 điểm ảnh. Bốn cột ở giữa có màu đen (giá trị 0) và các cột còn lại có màu trắng (giá trị 1)

1	1	0	0	0	0	1	1
1	1	0	0	0	0	1	1
1	1	0	0	0	0	1	1
1	1	0	0	0	0	1	1
1	1	0	0	0	0	1	1
1	1	0	0	0	0	1	1

```
X = np.ones((6, 8))  
X[:, 2:6] = 0  
X
```

Hạt nhân K có kích thước 1×2

1	-1
---	----

```
K = np.array([[1, -1]])
```

Thực hiện phép tương quan chéo giữa X (đầu vào) và K (hạt nhân):

- Hai phần tử cạnh nhau theo chiều ngang có giá trị giống nhau thì đầu ra sẽ bằng 0
- Ngược lại, đầu ra khác không

0	1	0	0	0	-1	0
0	1	0	0	0	-1	0
0	1	0	0	0	-1	0
0	1	0	0	0	-1	0
0	1	0	0	0	-1	0
0	1	0	0	0	-1	0

```
Y = corr2d(X, K)  
Y
```



Học một bộ lọc

```
# Construct a convolutional layer with 1 output channel
# (channels will be introduced in the following section)
# and a kernel array shape of (1, 2)
conv2d = nn.Conv2D(1, kernel_size=(1, 2))
conv2d.initialize()

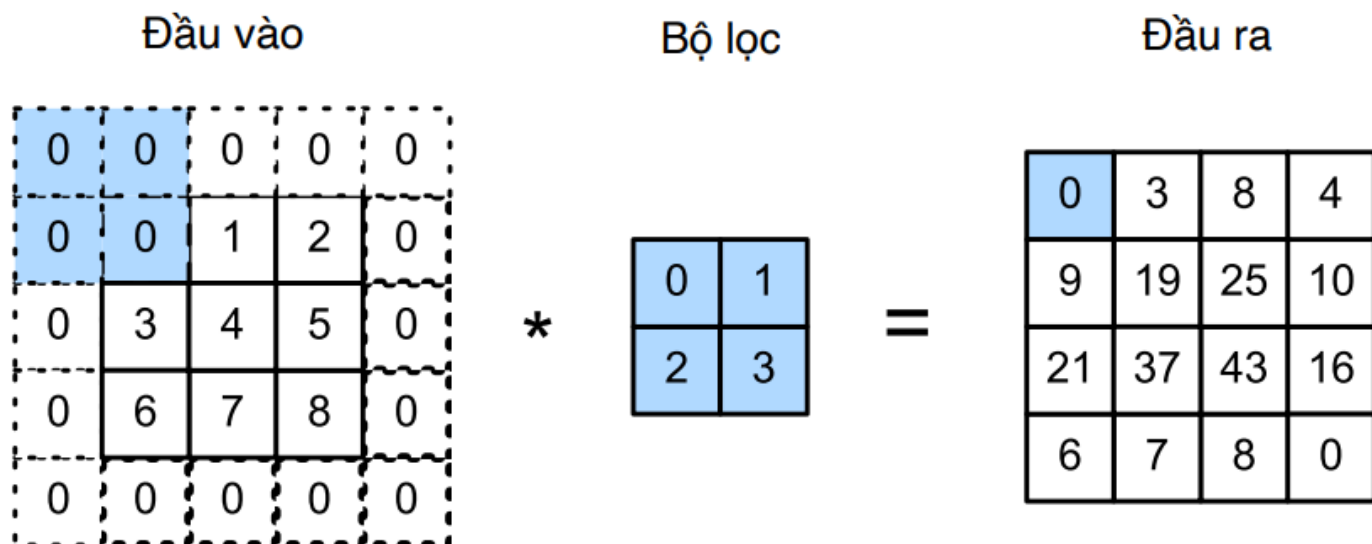
# The two-dimensional convolutional layer uses four-dimensional input and
# output in the format of (example, channel, height, width), where the batch
# size (number of examples in the batch) and the number of channels are both 1
X = X.reshape(1, 1, 6, 8)
Y = Y.reshape(1, 1, 6, 7)

for i in range(10):
    with autograd.record():
        Y_hat = conv2d(X)
        l = (Y_hat - Y) ** 2
    l.backward()
    # For the sake of simplicity, we ignore the bias here
    conv2d.weight.data()[:] -= 3e-2 * conv2d.weight.grad()
    if (i + 1) % 2 == 0:
        print('batch %d, loss %.3f' % (i + 1, l.sum()))

conv2d.weight.data().reshape(1, 2)
```

Đệm (padding)

- Chèn thêm các điểm ảnh xung quanh đường biên trên bức ảnh đầu vào, để tăng kích thước sử dụng của bức ảnh
- Thông thường, các giá trị của các điểm ảnh thêm vào là 0





Đệm (padding)

- Nếu chèn thêm tổng cộng p_h hàng đệm, p_w cột đệm, kích thước đầu ra sẽ là:

$$(n_h - k_h + p_h + 1) \times (n_w - k_w + p_w + 1)$$

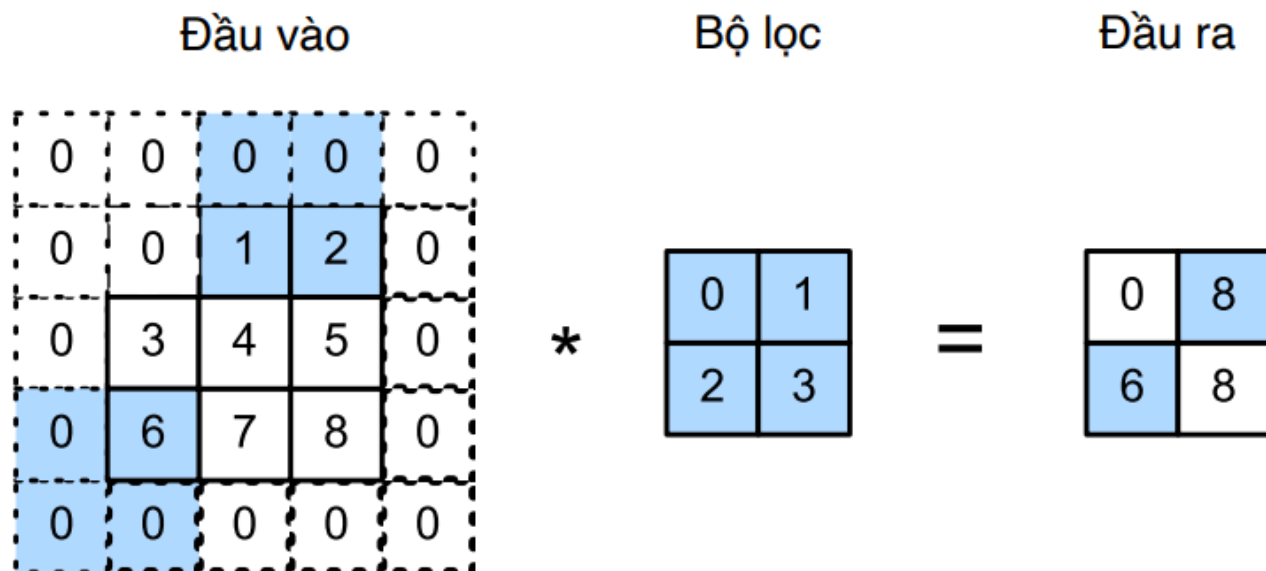
- Để đầu vào và đầu ra có cùng chiều dài và chiều rộng:

$$p_h = k_h - 1 \text{ và } p_w = k_w - 1$$

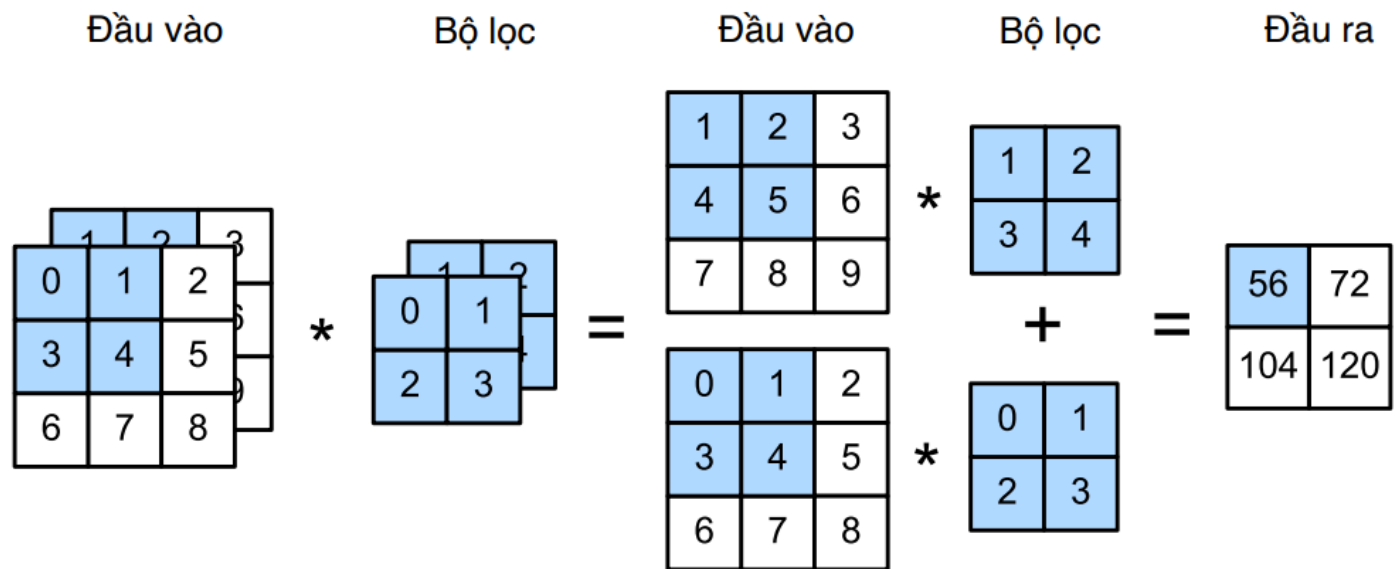
Sải bước (stride)

- Sải bước: số hàng và cột di chuyển qua mỗi lần
- Ví dụ, sải bước 3 theo chiều dọc và 2 theo chiều ngang
- Sải bước theo chiều cao là s_h và sải bước theo chiều rộng là s_w , kích thước đầu ra là:

$$\lfloor (n_h - k_h + p_h + s_h) / s_h \rfloor \times \lfloor (n_w - k_w + p_w + s_w) / s_w \rfloor$$

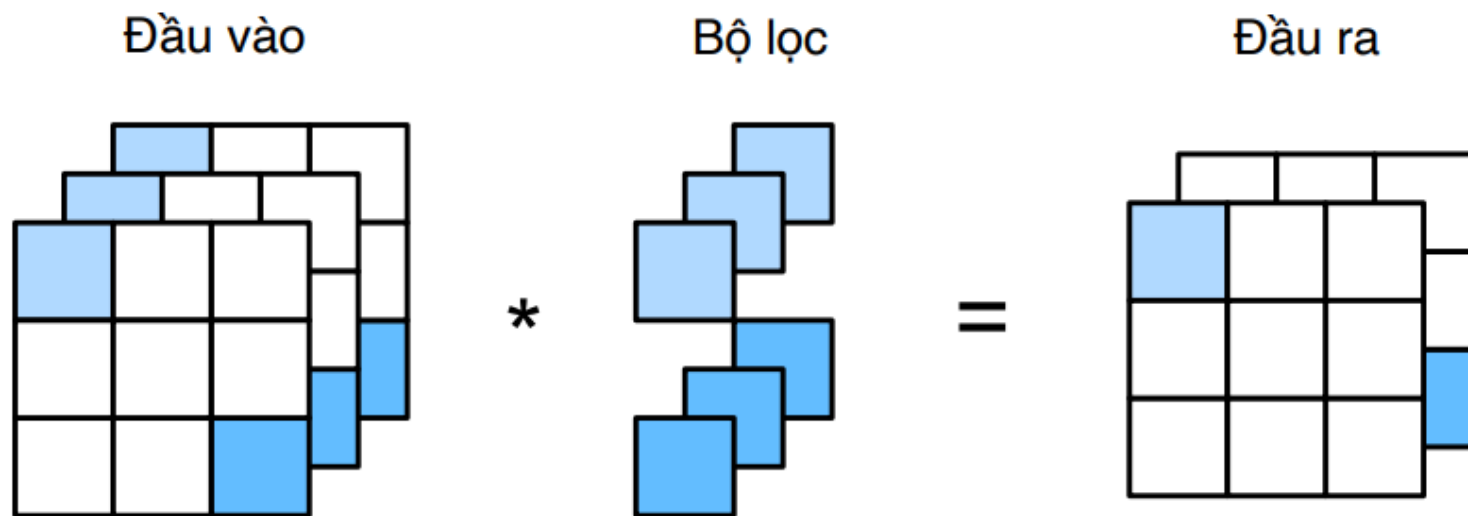


Đa kênh đầu vào



$$(1 \times 1 + 2 \times 2 + 4 \times 3 + 5 \times 4) + (0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3) = 56$$

Đa kênh đầu ra





Nội dung

- Tầng tích chập (convolutional layer)
 - Phép tích chập cho ảnh
 - Đệm và bước sải
 - Đa kênh đầu vào/ra
- **Tầng gộp (pooling layer)**
 - Phép gộp
 - Đệm và bước sải
 - Đa kênh đầu vào
- Tầng kết nối đầy đủ (Fully connected layer)
- Mạng Nơ-ron tích chập LeNet



Tầng gộp (pooling layer)

Pooling layer thường được dùng giữa các convolutional layer. Chức năng của các tầng gộp:

- Giảm độ nhạy cảm của các tầng tích chập đối với vị trí
- Giảm kích thước của các biểu diễn



Gộp (Pooling)

- Input: ma trận đầu vào (X), ma trận gộp K
- Output: Max pooling (X, K) hoặc Average pooling(X, K)

Max Pooling

- Tầng gộp với cửa sổ gộp $p \times q$ được gọi là một tầng gộp $p \times q$. Phép gộp được gọi là phép gộp $p \times q$
- Tính giá trị cực đại của các phần tử trong cửa sổ gộp

$$\max(0, 1, 3, 4) = 4,$$

$$\max(1, 2, 4, 5) = 5,$$

$$\max(3, 4, 6, 7) = 7,$$

$$\max(4, 5, 7, 8) = 8$$

Đầu vào

0	1	2
3	4	5
6	7	8

2 x 2
Gộp cực đại

Đầu ra

4	5
7	8



Average Pooling

- Tính giá trị trung bình của các phần tử trong cửa sổ gộp

$$\text{avg}(0, 1, 3, 4) = 2,$$

$$\text{avg}(1, 2, 4, 5) = 5,$$


$$\text{avg}(3, 4, 6, 7) = 7,$$

$$\text{avg}(4, 5, 7, 8) = 8$$

0	1	2
3	4	5
6	7	8

2 x 2
Gộp cực đại

2	3
5	6



Đệm (padding) và sải bước (stride)

- Cách thực hiện padding và stride của tầng gộp tương tự trong tầng tích chập



Đa kênh đầu vào

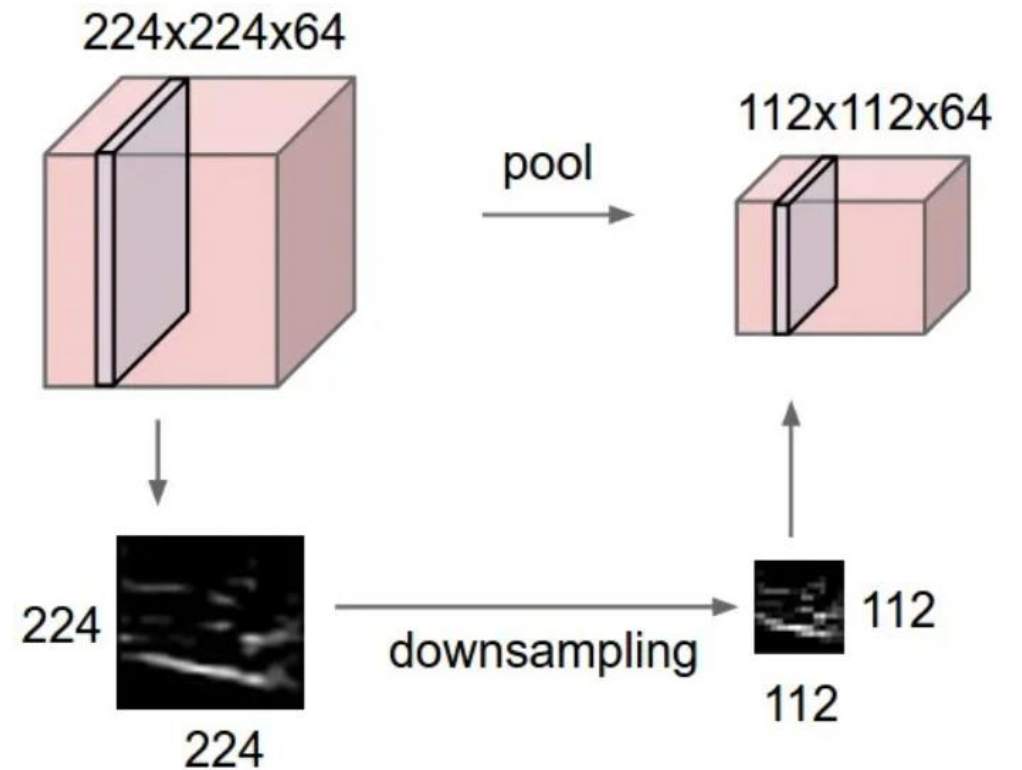
- Khi xử lý dữ liệu đầu vào đa kênh, tầng gộp sẽ áp dụng lên từng kênh một cách riêng biệt thay vì cộng từng phần tử tương ứng của các kênh lại với nhau như tầng tích chập

=> Số lượng kênh đầu ra của tầng gộp sẽ giống số lượng kênh đầu vào

Đa kênh đầu vào

Ví dụ

- pooling layer có: $\text{size}=(2,2)$, $\text{stride}=2$, $\text{padding}=0$
- output width và height của dữ liệu giảm đi một nửa, depth thì được giữ nguyên





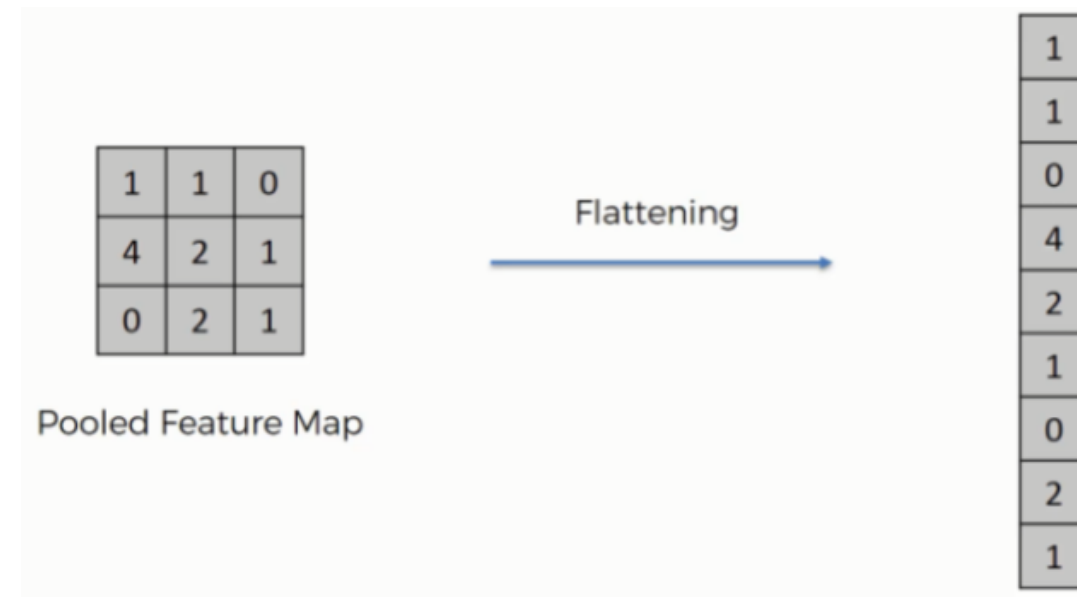
Nội dung

- Tầng tích chập (convolutional layer)
 - Phép tích chập cho ảnh
 - Độm và bước sải
 - Đa kênh đầu vào/ra
- Tầng gộp (pooling layer)
 - Phép gộp
 - Độm và bước sải
 - Đa kênh đầu vào
- **Tầng kết nối đầy đủ (Fully connected layer)**
- Mạng Nơ-ron tích chập LeNet

Tầng kết nối đầy đủ (Fully connected layer)

Ví dụ

- Sau khi ảnh được truyền qua nhiều convolutional layer và pooling layer thì tensor của output của layer cuối cùng (kích thước $H*W*D$) được chuyển về 1 vector (kích thước $(H*W*D)$)
- Dùng các fully connected layer để kết hợp các đặc điểm của ảnh để ra được output của model



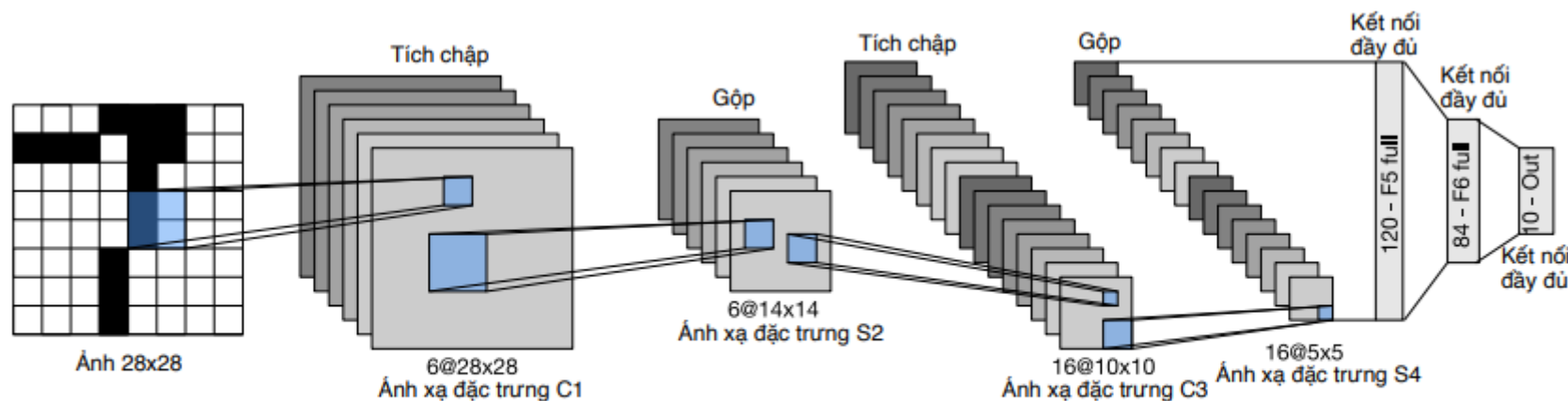


Nội dung

- Tầng tích chập (convolutional layer)
 - Phép tích chập cho ảnh
 - Độm và bước sải
 - Đa kênh đầu vào/ra
- Tầng gộp (pooling layer)
 - Phép gộp
 - Độm và bước sải
 - Đa kênh đầu vào
- Tầng kết nối đầy đủ (Fully connected layer)
- **Mạng Nơ-ron tích chập LeNet**

Mạng Nơ-ron Tích chập (LeNet)

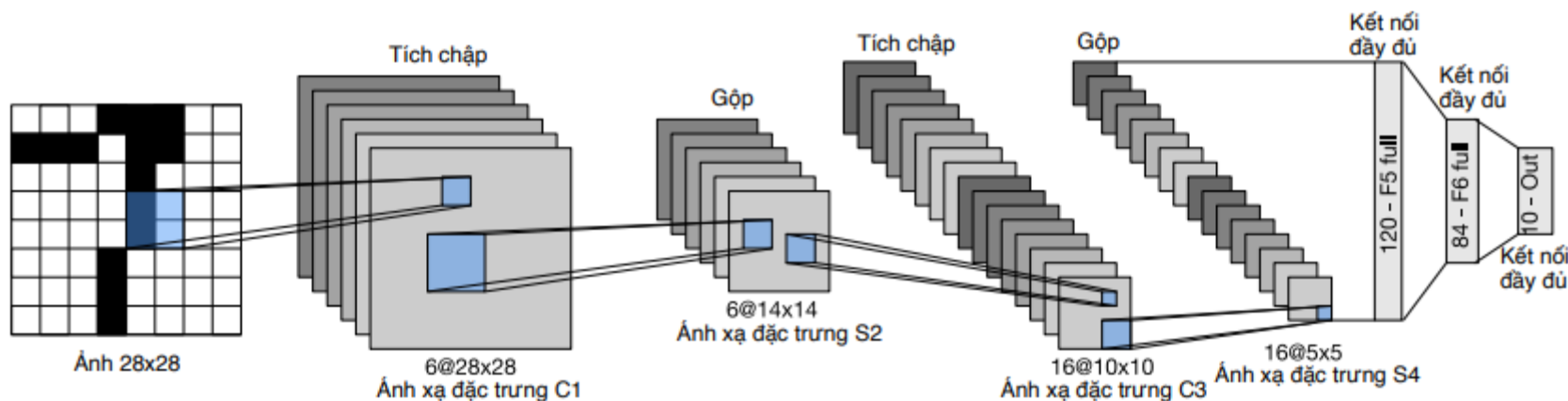
- Mạng tích chập được minh họa lần đầu bởi Yann Lecun với ứng dụng nhận dạng các số viết tay trong ảnh (LeNet5) vào những năm 1990



Dòng dữ liệu trong LeNet 5. Đầu vào là một chữ số viết tay, đầu ra là một xác suất đối với 10 kết quả khả thi.

Mạng Nơ-ron Tích chập (LeNet)

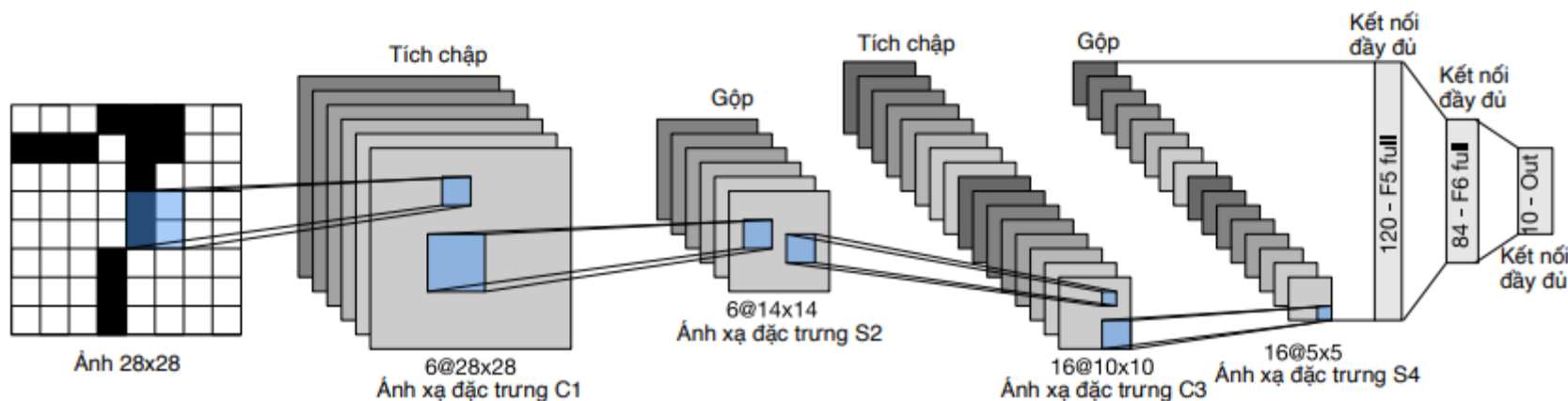
- Tầng tích chập dùng để nhận dạng các mẫu không gian trong ảnh: các đường cạnh và các bộ phận của vật thể.
 - Kernel size = 5×5 , hàm kích hoạt: sigmoid
 - Tầng tích chập đầu tiên có 6 kênh, tầng tích chập thứ hai có 16 kênh



Dòng dữ liệu trong LeNet 5. Đầu vào là một chữ số viết tay, đầu ra là một xác suất đối với 10 kết quả khả thi.

Mạng Nơ-ron Tích chập (LeNet)

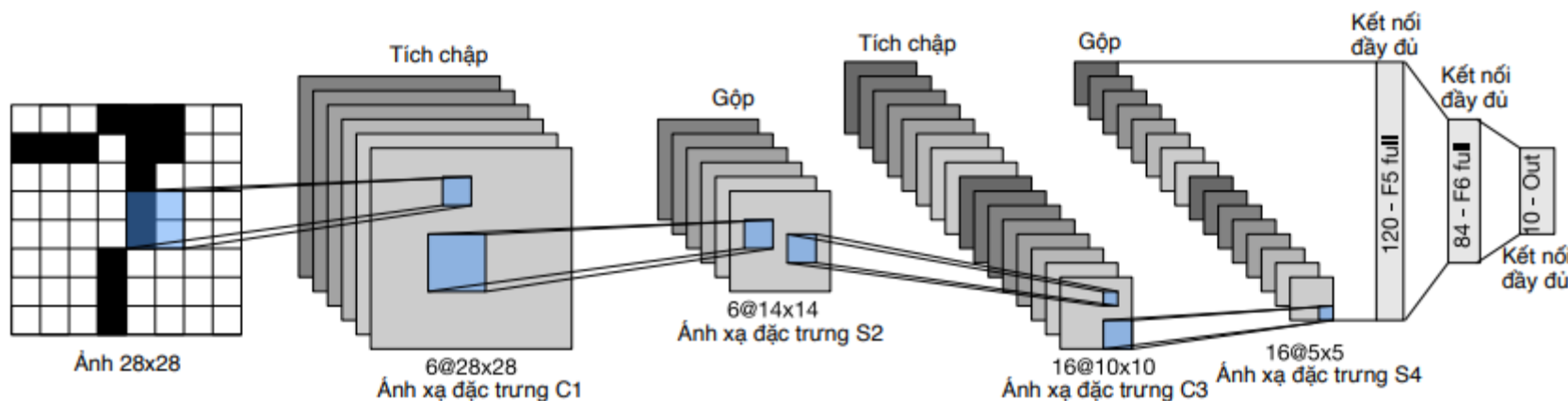
- Lớp gộp trung bình phía sau được dùng để giảm số chiều
 - Kernel size = 2x2, stride = 2



Dòng dữ liệu trong LeNet 5. Đầu vào là một chữ số viết tay, đầu ra là một xác suất đối với 10 kết quả khả thi.

Mạng Nơ-ron Tích chập (LeNet)

- Khối tầng kết nối đầy đủ của LeNet có ba tầng với số lượng đầu ra lần lượt là: 120, 84 và 10. (Tầng đầu ra 10 chiều tương ứng với số lượng các lớp đầu ra khả thi (10 chữ số từ 0 đến 9))

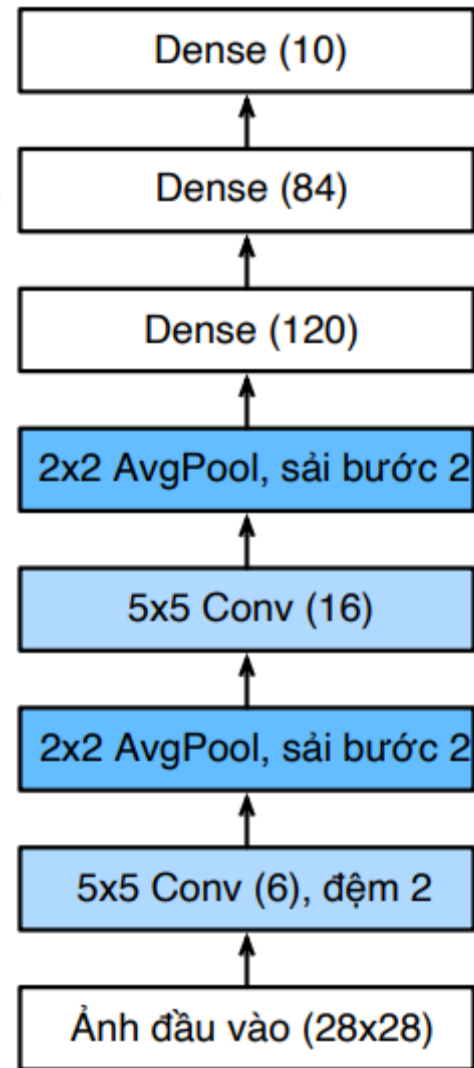


Dòng dữ liệu trong LeNet 5. Đầu vào là một chữ số viết tay, đầu ra là một xác suất đối với 10 kết quả khả thi.

Mạng Nơ-ron Tích chập (LeNet)

```
from d2l import mxnet as d2l
from mxnet import autograd, gluon, init, np, npx
from mxnet.gluon import nn
npx.set_np()

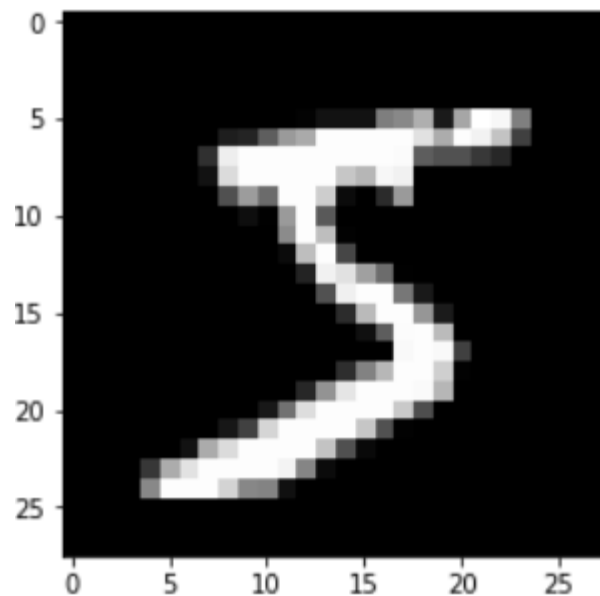
net = nn.Sequential()
net.add(nn.Conv2D(channels=6, kernel_size=5, padding=2, activation='sigmoid'),
        nn.AvgPool2D(pool_size=2, strides=2),
        nn.Conv2D(channels=16, kernel_size=5, activation='sigmoid'),
        nn.AvgPool2D(pool_size=2, strides=2),
        # Dense will transform the input of the shape (batch size, channel,
        # height, width) into the input of the shape (batch size,
        # channel * height * width) automatically by default
        nn.Dense(120, activation='sigmoid'),
        nn.Dense(84, activation='sigmoid'),
        nn.Dense(10))
```



Kí hiệu văn tắt cho mô hình LeNet5

Bài toán

- Bạn có ảnh xám kích thước 28×28 của chữ số từ 1 đến 9 và bạn muốn dự đoán số đây là số mấy. Ví dụ, dự đoán ảnh sau là số mấy.



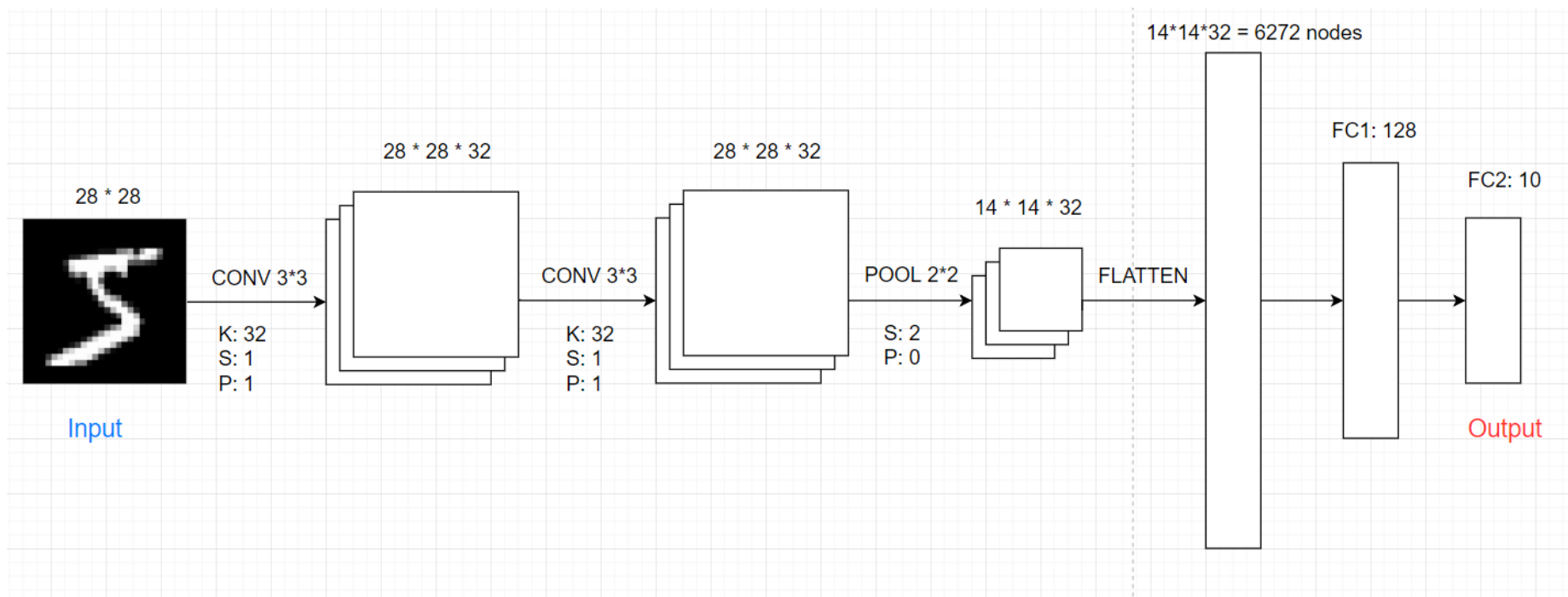


Bộ dữ liệu MNIST

- MNIST là bộ cơ sở dữ liệu về chữ số viết tay, bao gồm 2 tập con: training set gồm 60.000 ảnh các chữ số viết tay và test set gồm 10.000 ảnh các chữ số
- Chuẩn bị dữ liệu cho mô hình:
 - Training set: 50.000 dữ liệu
 - Validation set: 10.000 dữ liệu
 - Test set: 10.000 dữ liệu

Xây dựng model

- Mô hình chung bài toán CNN: Input image \rightarrow Convolutional layer (Conv) + Pooling layer (Pool) \rightarrow Fully connected layer (FC) \rightarrow Output



- Input của model là ảnh xám kích thước 28×28



Thêm các thư viện cần thiết

```
# 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, Dropout, Activation, Flatten  
from keras.layers import Conv2D, MaxPooling2D  
from keras.utils import np_utils  
from keras.datasets import mnist
```



Chuẩn bị các tập dữ liệu

- Load dữ liệu từ MNIST dataset gồm 60.000 training set, 10.000 test set
- Chia training set: 50.000 dữ liệu cho training set và 10.000 dữ liệu cho validation set

```
# 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)
```

Chuẩn bị các tập dữ liệu

- Dữ liệu input cho mô hình CNN là 1 tensor 4 chiều (N, W, H, D), trong bài này là ảnh xám nên $W = H = 28$, $D = 1$, N là số lượng ảnh cho mỗi lần training
- Do dữ liệu ảnh ở trên có kích thước là (N, 28, 28) tức là (N, W, H) nên cần reshape lại thành kích thước $N * 28 * 28 * 1$ để giống kích thước mà keras yêu cầu

X_train là tập dữ liệu 2 chiều nên phải reshape thành 4 chiều:
(28,28: kích thước. Mỗi 1 ảnh 1 ma trận nên là 1)

```
# 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)
```

Chuẩn bị các tập dữ liệu

- Chuyển đổi one-hot encoding label Y của ảnh
- Ví dụ số 5 thành vector [0, 0, 0, 0, 0, 1, 0, 0, 0, 0]

```
# 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])
```

y_train có 10 lớp, chuyển thành
các vector tương ứng có 10 phần

Định nghĩa model

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())
```

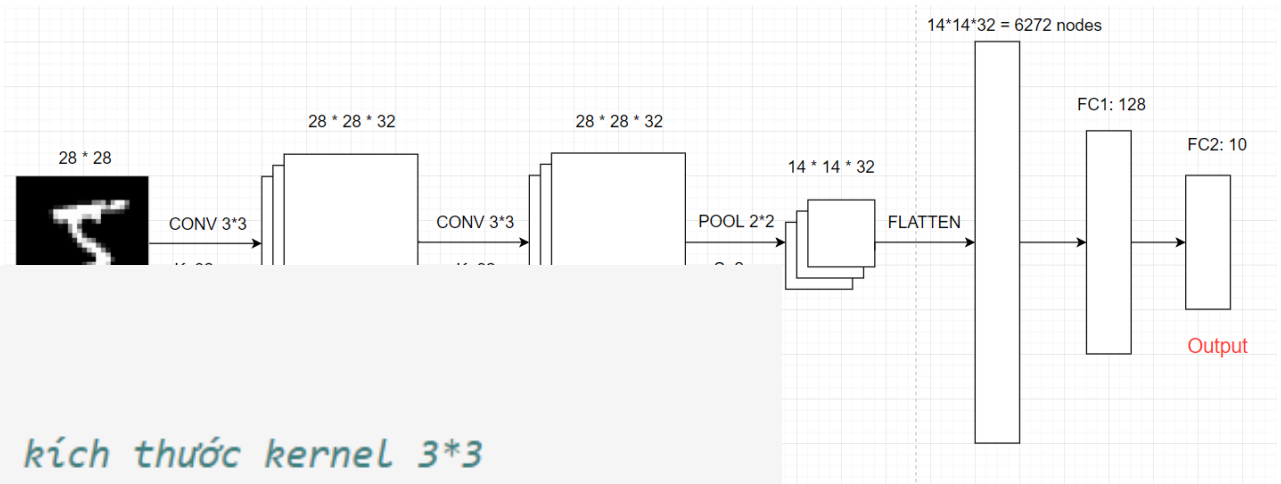
đuổi thẳng qua tính đc thành vector, tầng Flatten

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 suất.

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



lớp tích chập có 32 layer, cửa sổ là 3x3.



Compile model

```
# 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'])
```

- hàm mất mát của softmax là categorical...
-vì là bài toán phân lớp dùng độ đo
accuracy

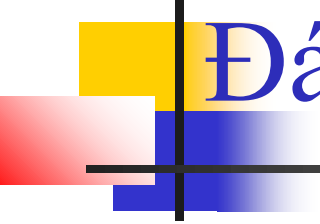


Train model với data

```
# 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)
```

batch size: kích thước tính adam
verbose: nếu = 1 thì sau mỗi 1 verbose
n sẽ hiển thị kqua X_train y_train
accuracy



Đánh giá model với test set

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

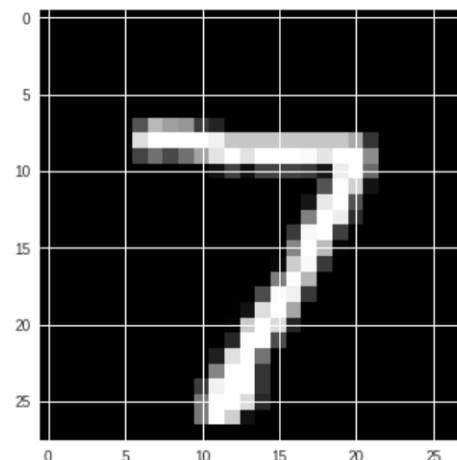
Dự đoán ảnh

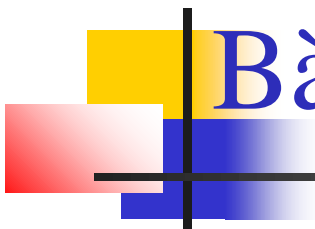
10. Dự đoán ảnh

```
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))
```

Giá trị dự đoán: 7





Bài tập

Xây dựng model cho CIFAR10 dataset bao gồm 50,000 training set và 10.000 test set ảnh màu kích thước 32×32 cho 10 thể loại khác nhau (máy bay, ô tô, thuyền, chim, chó, mèo, ngựa,...).

```
# Load dữ liệu cifar10
```

```
from keras.datasets import cifar10
```

```
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```