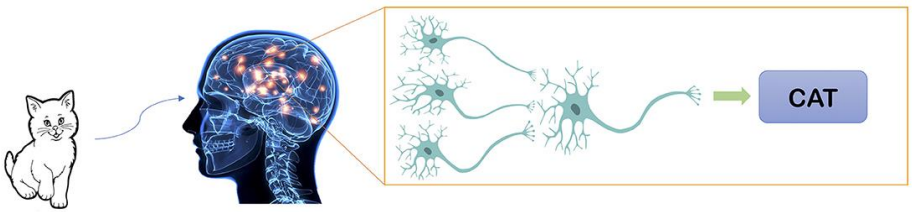
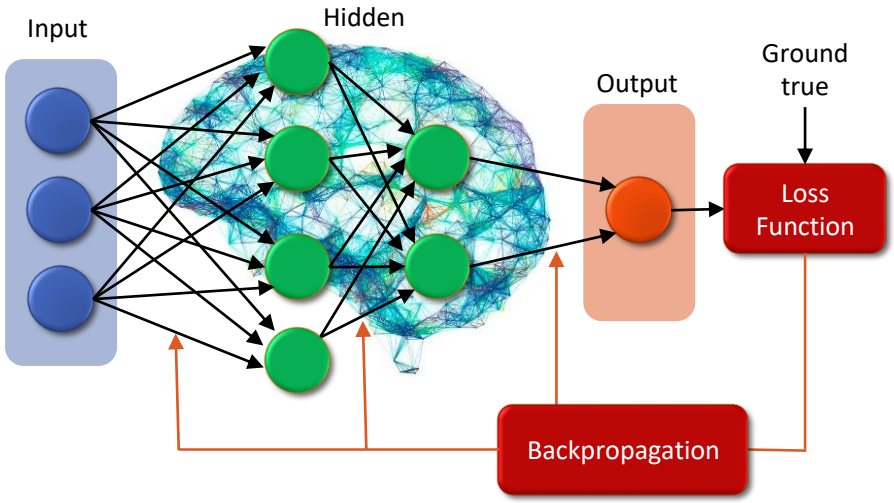
DEEP LEARNING
CÁC KIẾN TRÚC CNN (3)





Tôn Quang Toại
Khoa Công nghệ thông tin
Trường đại học Ngoại ngữ - Tin học TP.HCM (HUFLIT)

Nội dung

- LeNet (1998)
- AlexNet (2012)
- VGG (2014)
- Một số vấn đề của mạng sâu
  - Vanishing
  - Class Functions
- Mạng sâu
  - ResNet (2015)
  - DenseNet (2016), U-Net
  - EfficientNet (2019)
  - ConvNeXt (2022)

DENSENET

DENSELY CONNECTED CONVOLUTIONAL NETWORKS

DenseNet

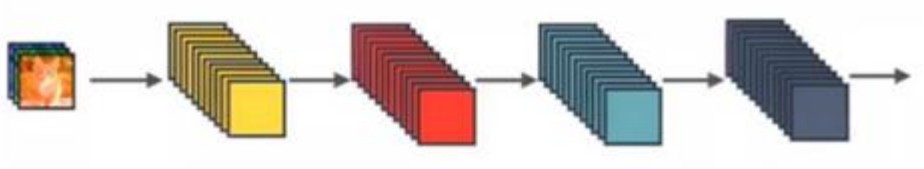
- DenseNet: Densely Connected Convolutional Networks

- Tác giả
  - Gao Huang
  - Zhuang Liu
  - Laurens van der Maaten
  - Kilian Q. Weinberger

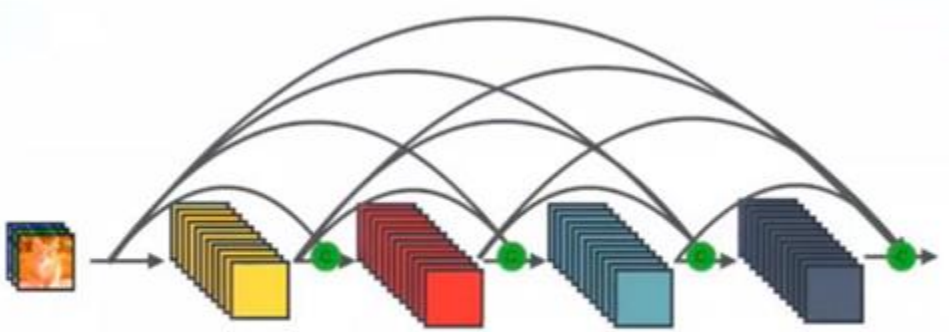
- Bài báo
  - Densely Connected Convolutional Networks, 2016, 2017, 2018

- Link
  - https://arxiv.org/abs/1608.06993

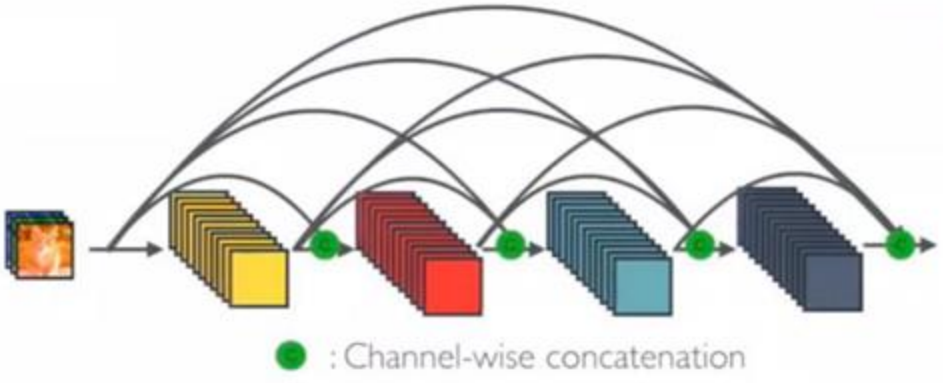
CNN thông thường, ResNet, DenseNet



Standard CNN



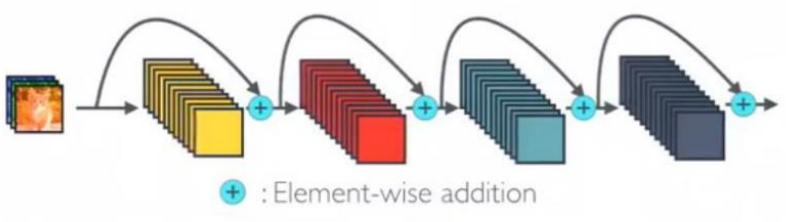
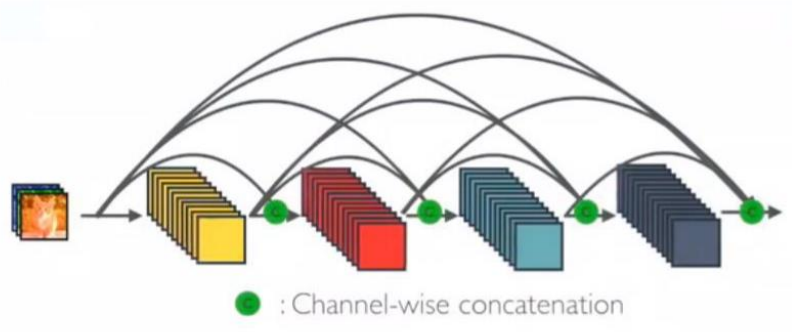
DenseNet



DenseNet

ResNet và DenseNet

ResNet vs. DenseNet

ResNet	DenseNet
	

Skip Connection

Addition

vs.

Dense Connection

Concatenating

Nhận xét 1

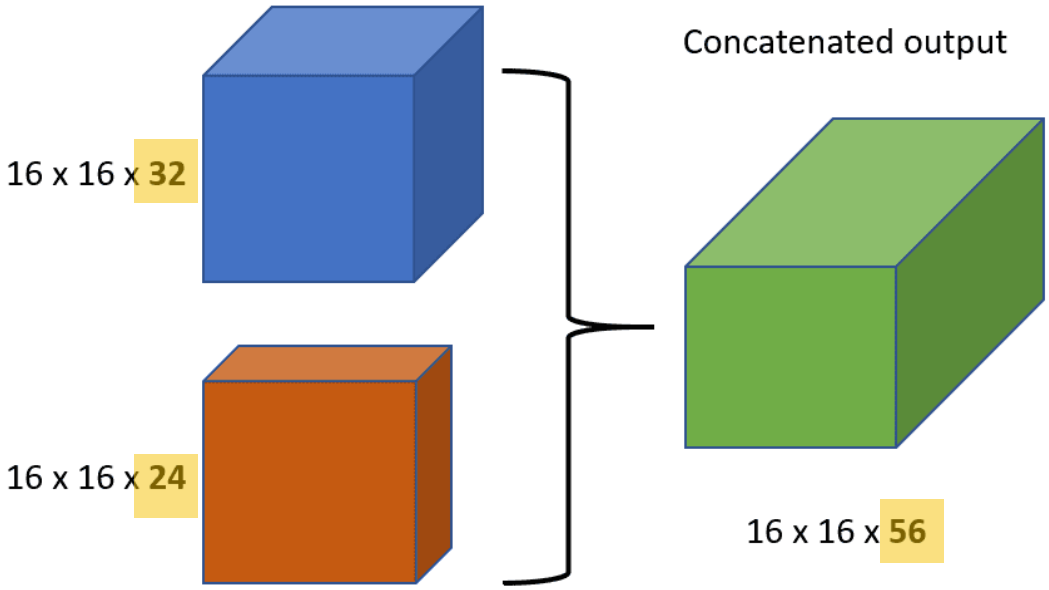
1. ResNet: element-wise addition
2. DenseNet: channel-wise concatenation

Nhận xét 2

1. ResNet: Kết nối nhảy tầng
2. DenseNet: Kết nối dày đặc

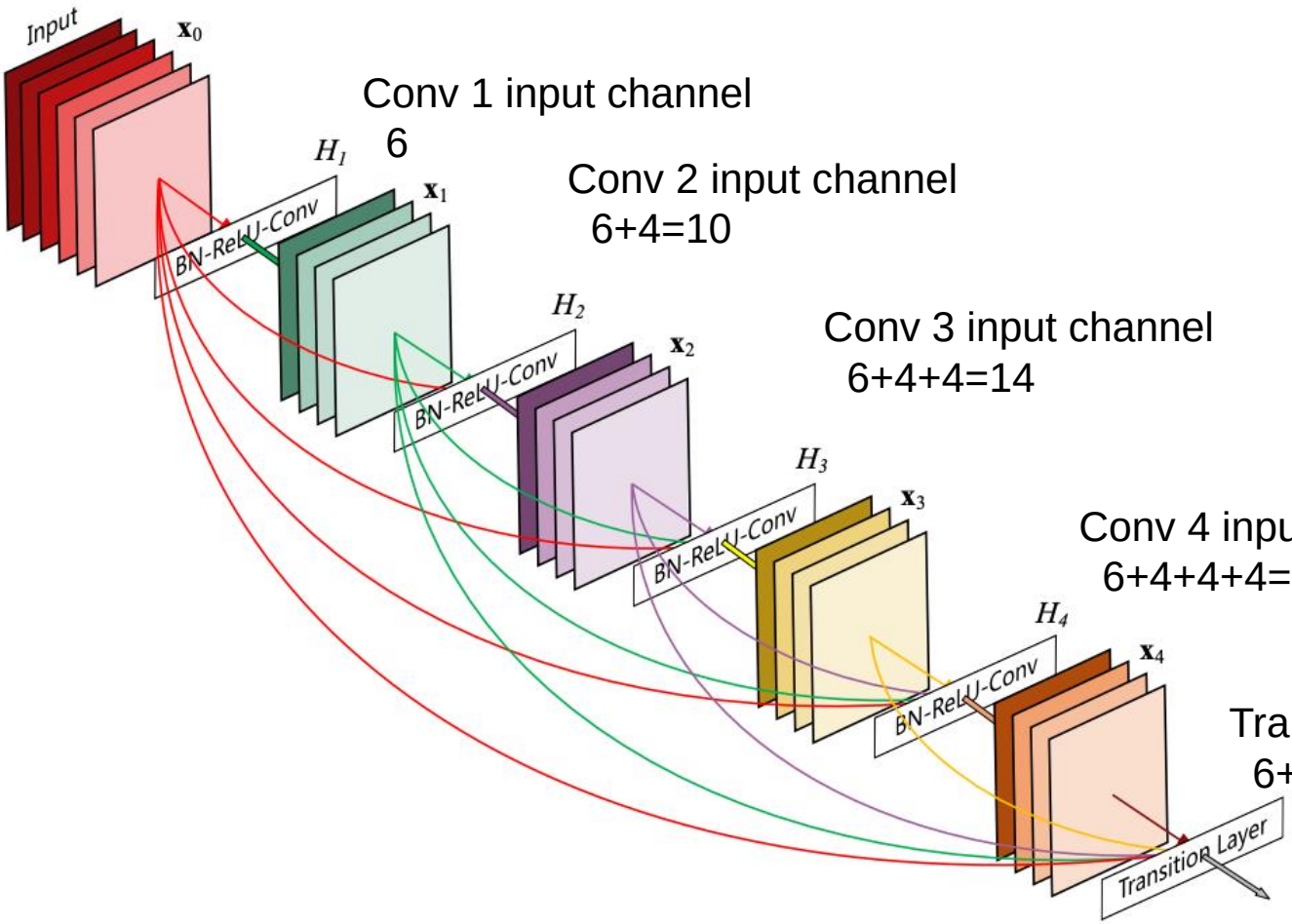
ResNet và DenseNet

- Phép toán Concatenated




```
tf.keras.layers.concatenate(inputs, axis=-1)
```

Kiến trúc DenseNet



Conv 1 input channel
6

Conv 2 input channel
6+4=10

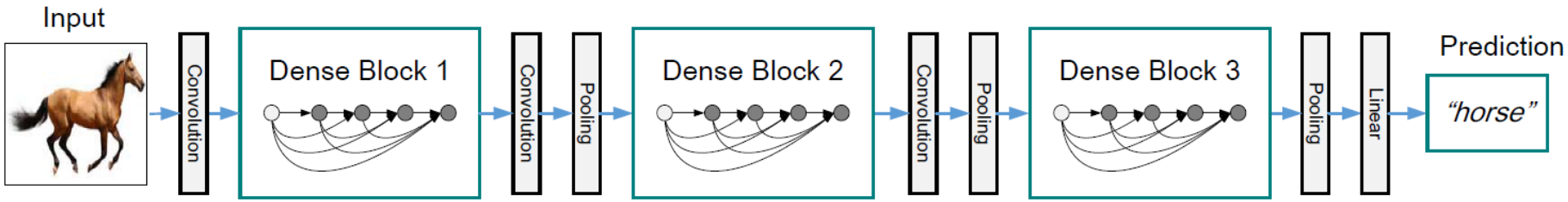
Conv 3 input channel
6+4+4=14

Conv 4 input channel
6+4+4+4=18

Transition Layer input channel
6+4+4+4+2=22

Kiến trúc DenseNet

- Các khối trong DenseNet
  - Convolution Block (conv_block): **BN + ReLU + Conv**
  - Dense Block
    - Gồm nhiều conv_block có cùng số kênh đầu ra
    - Concatenate đầu ra và đầu ra của mỗi conv_block



  - Transition Layer: Mỗi DenseBlock sẽ tăng số channel → Mô hình phức tạp quá mức
    - Điều khiển độ phức tạp của mô hình
    - Giảm số kênh bằng convolution 1×1

Kiến trúc DenseNet

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv; 3 × 3 conv] × 24	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48	[1 × 1 conv; 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv; 3 × 3 conv] × 16	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Hiện thực đơn giản

```
def conv_block(inputs):
    block = BatchNormalization()(inputs)
    block = ReLU()(block)
    block = Conv2D(12, (3,3), padding='same')(block)
    return block


def dense_block(inputs):
    concatenated_inputs = inputs
    for i in range(3):
        x = conv_block(concatenated_inputs)
        concatenated_inputs = concatenate([concatenated_inputs, x], axis=3)
    return concatenated_inputs
```

Hiện thực đơn giản

```
def mini_dense_model(shape):
    inputs = Input(shape)

    x = dense_block(inputs)
    x = Flatten()(x)
    x = Dense(64, activation='softmax')(x)
    predictions = Dense(10, activation='softmax')(x)

    model = Model(inputs=inputs, outputs=predictions)

    return model
```