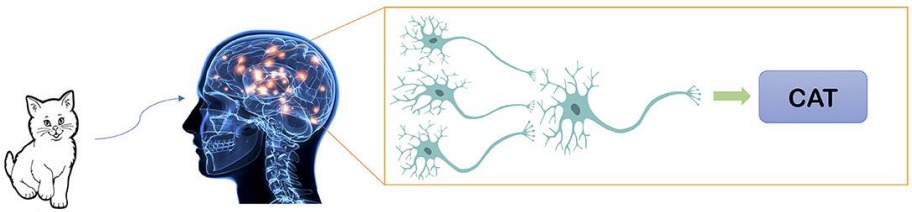
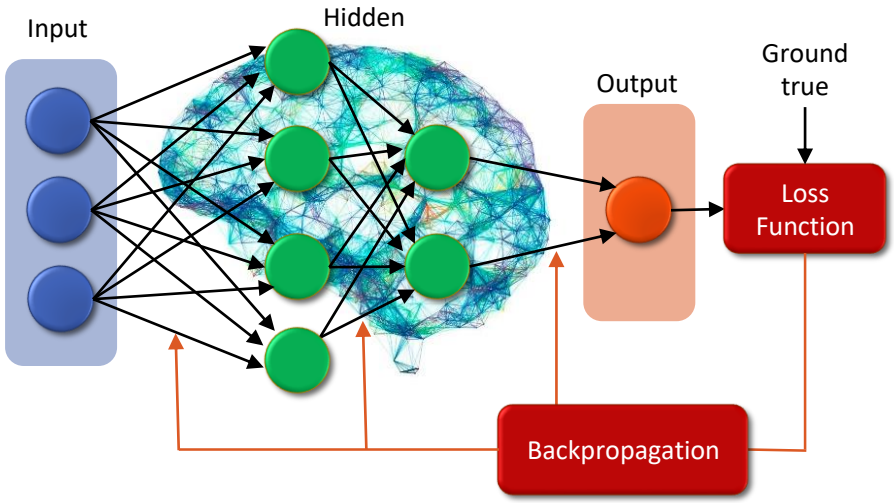
DEEP LEARNING
CÁC KIẾN TRÚC CNN (3)





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)

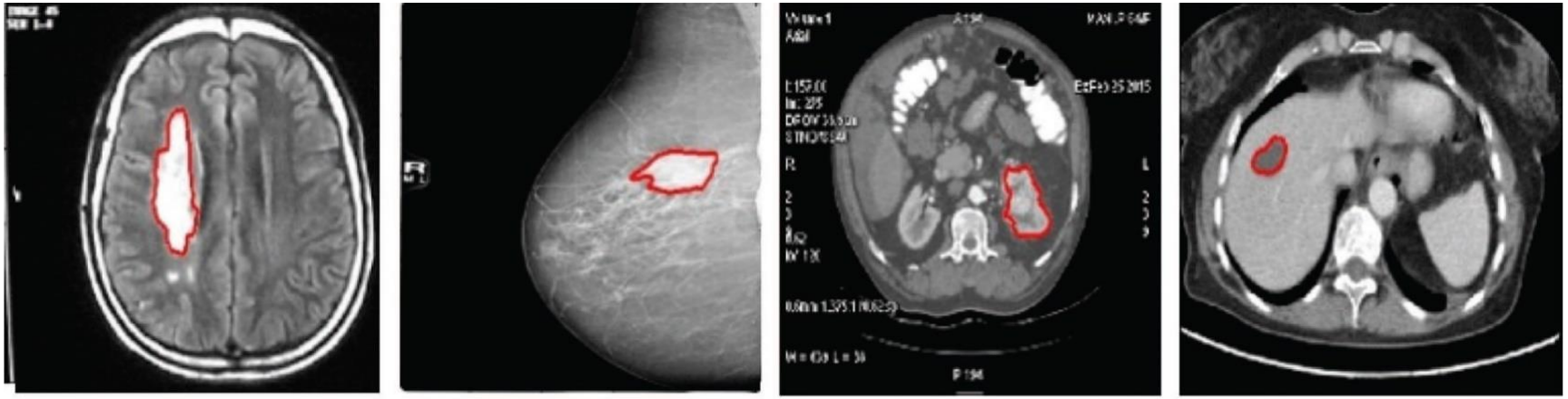
SEMANTIC SEGMENTATION

Phân đoạn hình ảnh

- Phân đoạn hình ảnh (Image segmentation): Phân đoạn hình ảnh phân đoạn một hình ảnh thành nhiều vùng bằng cách gán nhãn cho mỗi pixel của hình ảnh.

Ứng dụng của Phân đoạn hình ảnh

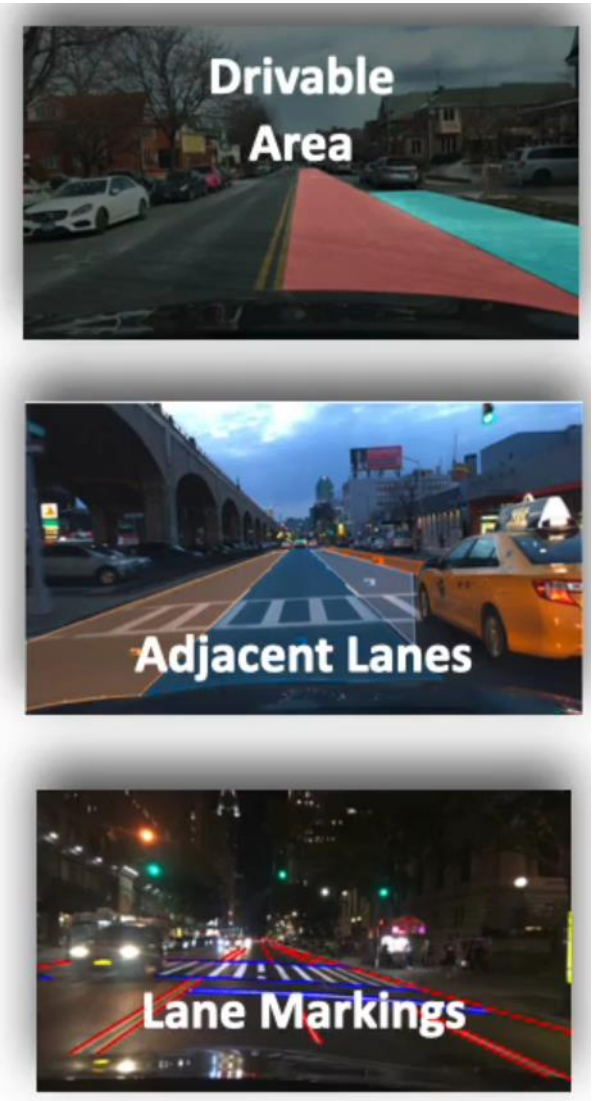
- Y tế



A B C D

Ứng dụng của Phân đoạn hình ảnh

- Xe tự lái
  - Đường
  - Các làn đường
  - Hình dán đường
  - Phần đường có thể lái
  - Phần đường đối diện
  - Vật cản trên đường
  - …



Các kiểu phân đoạn hình ảnh

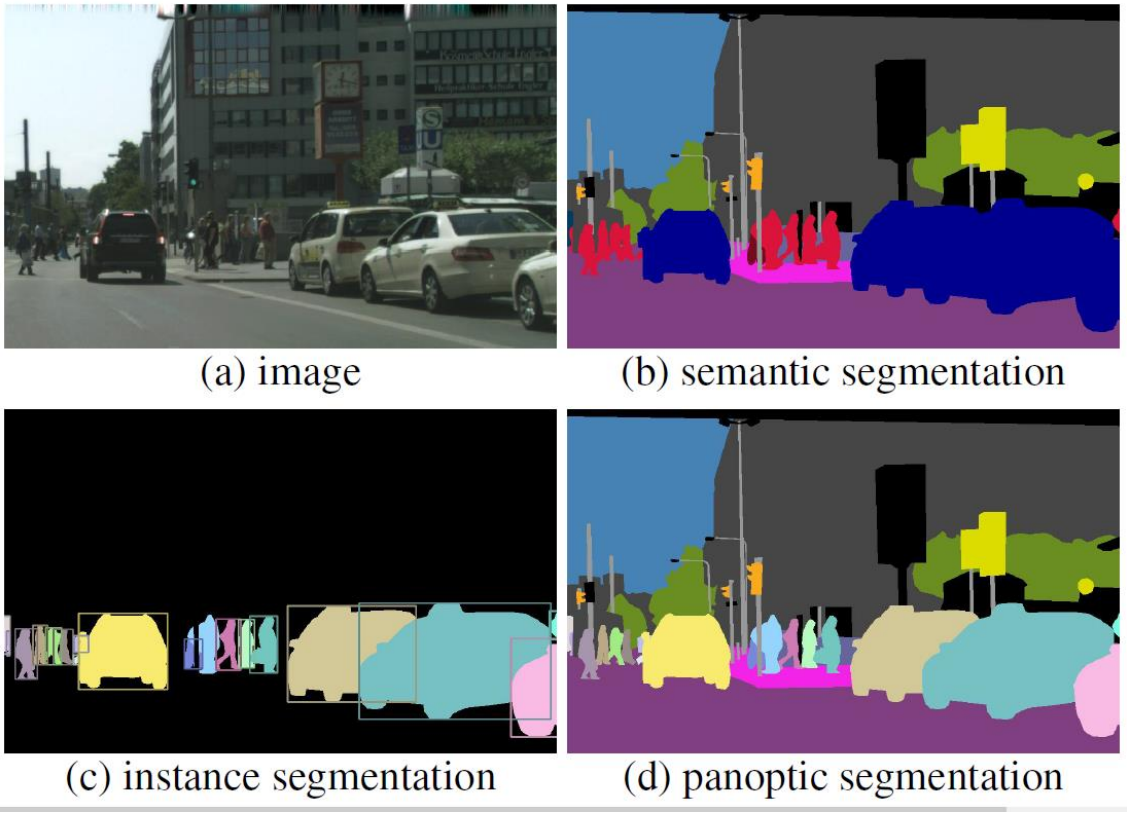
- Phân đoạn ngữ nghĩa (Semantic segmentation): phân loại mỗi pixel bằng một nhãn.

- Phân đoạn cá thể (Instance segmentation): phân loại từng pixel và phân biệt từng cá thể đối tượng.

Input Image | Semantic Segmentation | Instance Segmentation

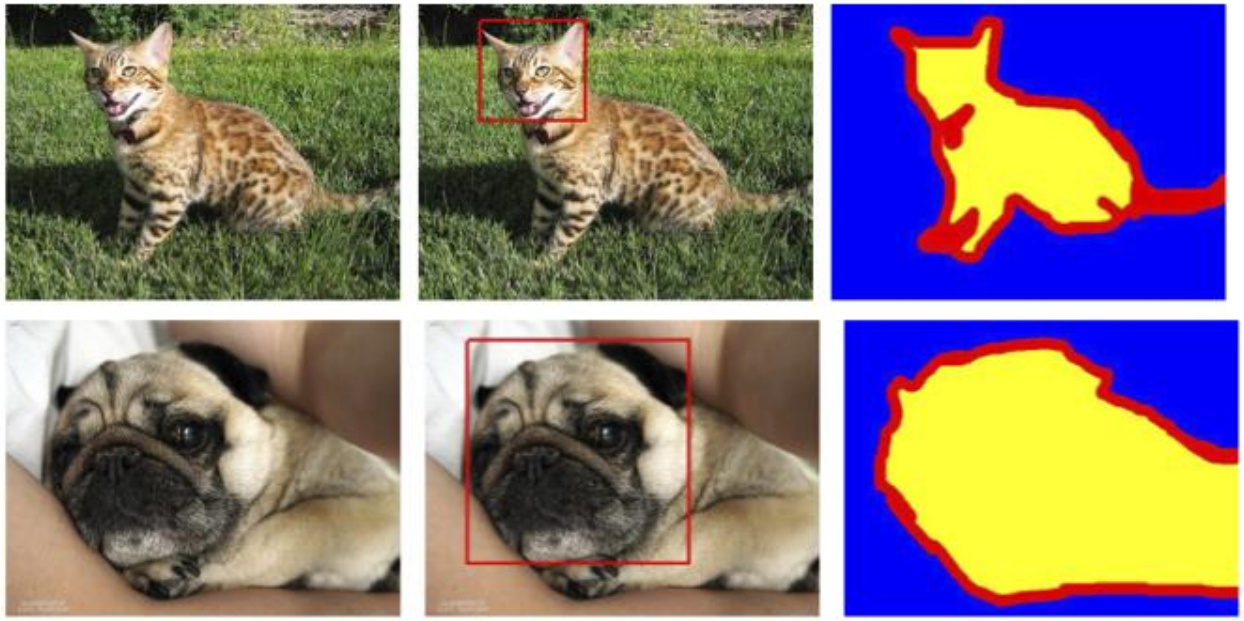
Các kiểu phân đoạn hình ảnh

- Phân đoạn toàn cảnh (Panoptic Segmentation): phương pháp lai kết hợp giữa phân đoạn ngữ nghĩa và phân đoạn cá thể.



Dataset

- Dataset (Cat and Dog): The Oxford-IIIT Pet Dataset



- Link: https://www.robots.ox.ac.uk/~vgg/data/pets/
- Kích thước: 800MB
- Số ảnh: 7349 ảnh

Dataset

- Tổng quan
  - 37 loại vật nuôi
  - Khoảng 200 ảnh/1 loại

Breed	Count
American Bulldog	200
American Pit Bull Terrier	200
Basset Hound	200
Beagle	200
Boxer	199
Chihuahua	200
English Cocker Spaniel	196
English Setter	200
German Shorthaired	200
Great Pyrenees	200
Havanese	200
Japanese Chin	200
Keeshond	199
Leonberger	200
Miniature Pinscher	200
Newfoundland	196
Pomeranian	200
Pug	200
Saint Bernard	200
Samyoed	200
Scottish Terrier	199
Shiba Inu	200
Staffordshire Bull Terrier	189
Wheaten Terrier	200
Yorkshire Terrier	200
Total	4978

1.Dog Breeds

Breed	Count
Abyssinian	198
Bengal	200
Birman	200
Bombay	200
British Shorthair	184
Egyptian Mau	200
Main Coon	190
Persian	200
Ragdoll	200
Russian Blue	200
Siamese	199
Sphynx	200
Total	2371

2.Cat Breeds

Family	Count
Cat	2371
Dog	4978
Total	7349

3.Total Pets

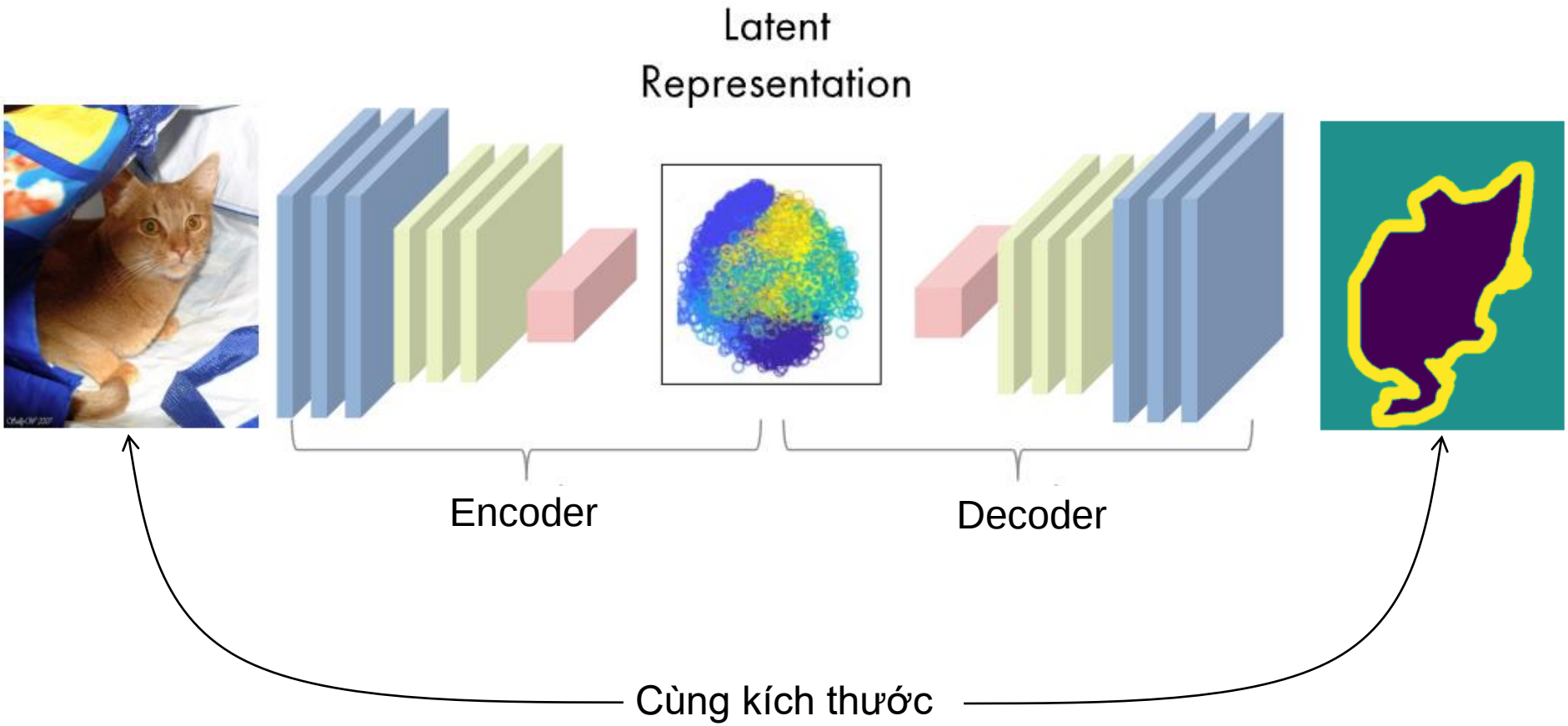
Dataset

- Nhận xét
  - Ảnh có biến thể lớn về tỷ lệ, tư thế và ánh sáng
  - Mỗi ảnh đều có ground truth annotation theo loại
  - Phân đoạn 3 mức cấp độ pixel

- Dataset: https://thor.robots.ox.ac.uk/~vgg/data/pets/images.tar.gz
- Annotations: https://thor.robots.ox.ac.uk/~vgg/data/pets/annotations.tar.gz

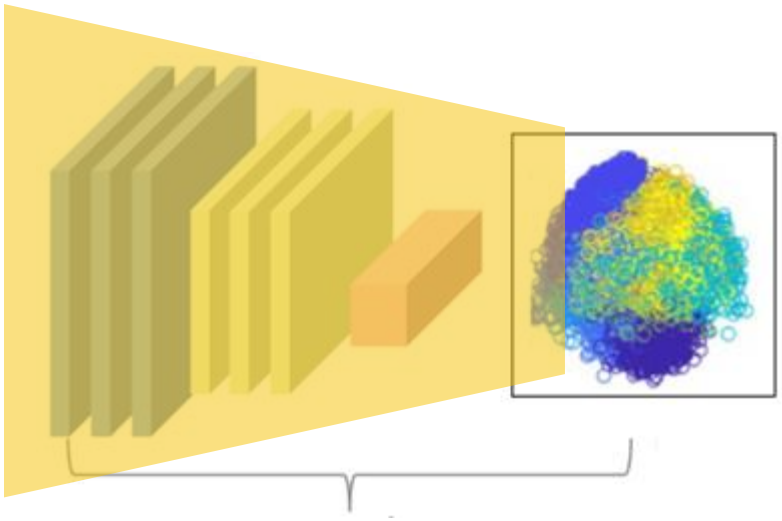
KIẾN TRÚC CHO SEGMENTATION

Autoencoders



Encoders

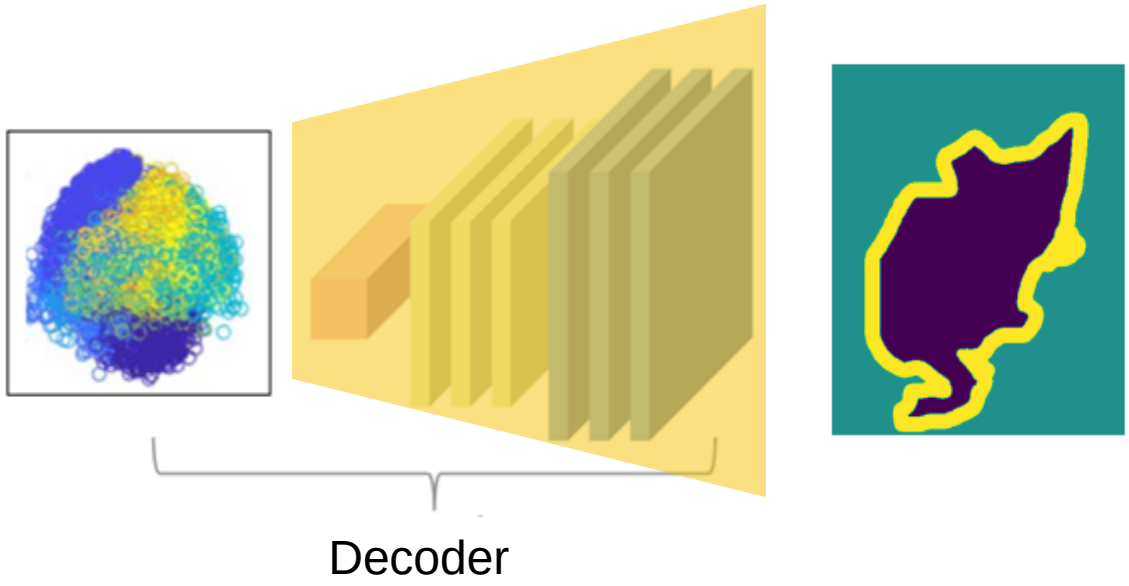
- Extract features



Encoder

Decoder

- Tạo output



4 ý tưởng để hiểu Autoencoder

- Latent Space Representation: feature maps được nén cực hạn sau chuỗi convolutions
- Upsampling: Tạo lại ảnh có kích thước bằng input
- Skip Connections: Tăng cường chất lượng mạng và train mạng sâu
- Pixelwise classification: Phân lớp mỗi điểm ảnh

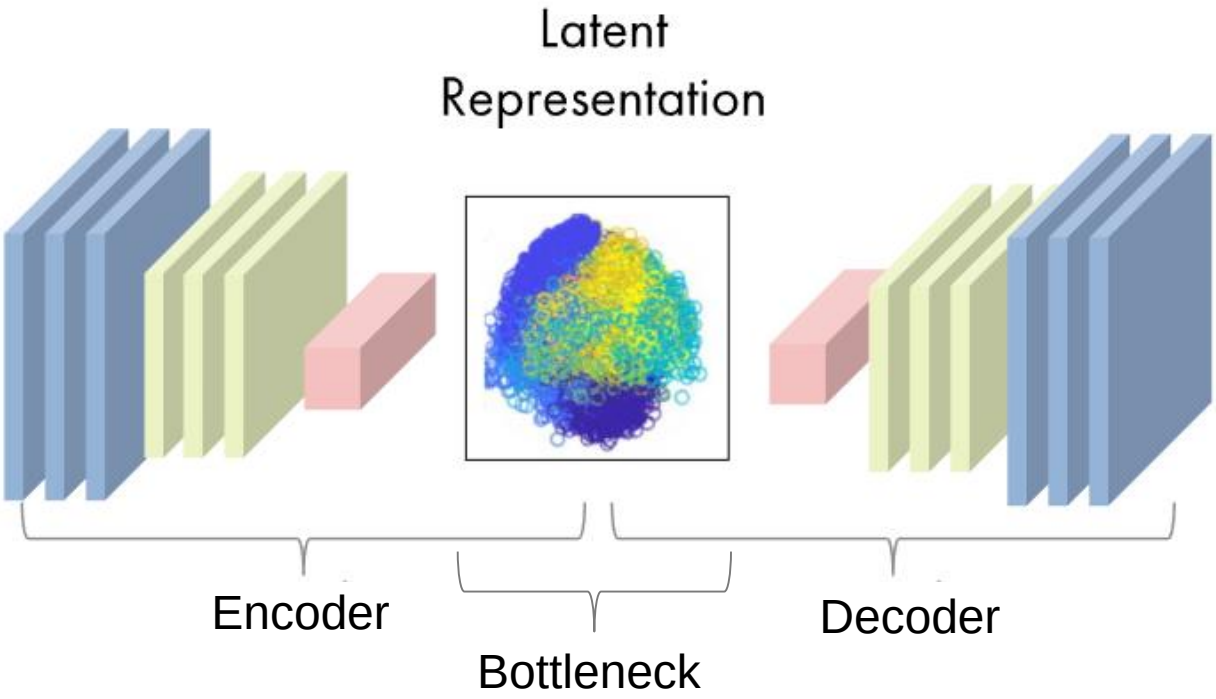
Latent Space Representation

- Từ “latent" có nghĩa là “hidden".
- Latent Space Representation: Không gian tiềm ẩn nơi các điểm dữ liệu tương tự thì gần nhau hơn.

- Ví dụ
  - Không gian pixel:
    - hai ghế không gần nhau
  - Trong Laten space:
    - Hai ghế gần nhau

Latent Space Representation

- Biểu diễn nén của một ảnh sau khi qua encoder
- Nó chứa mọi thứ mà chúng ta đã học, chúng ta có thể dùng để phân lớp

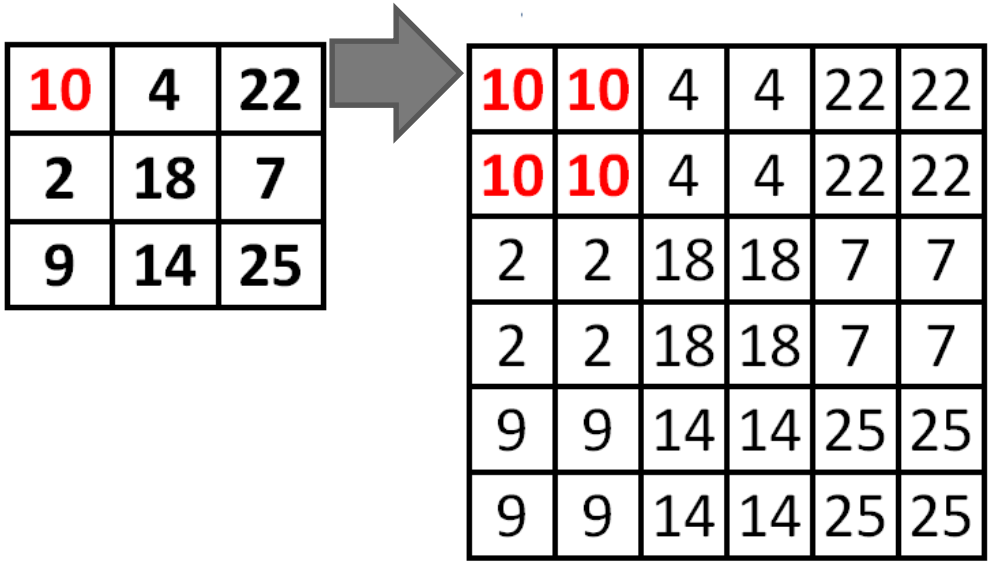


Upsampling

- Có 4 cách để upsampling một vector đã bị nén thành ảnh
  - Nearest Neighbor Interpolation
  - Bilinear Upsampling
  - Transposed Convolutions
  - Max Unpooling

Upsampling

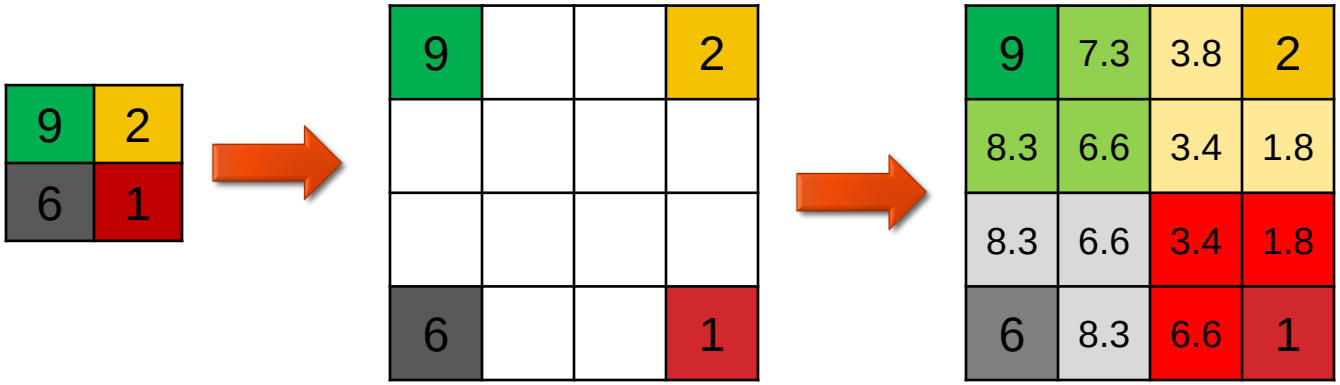
- Nearest Neighbor Interpolation




```
tf.keras.layers.UpSampling2D(
        size=(2, 2), data_format=None,
        interpolation="nearest")
```

Upsampling

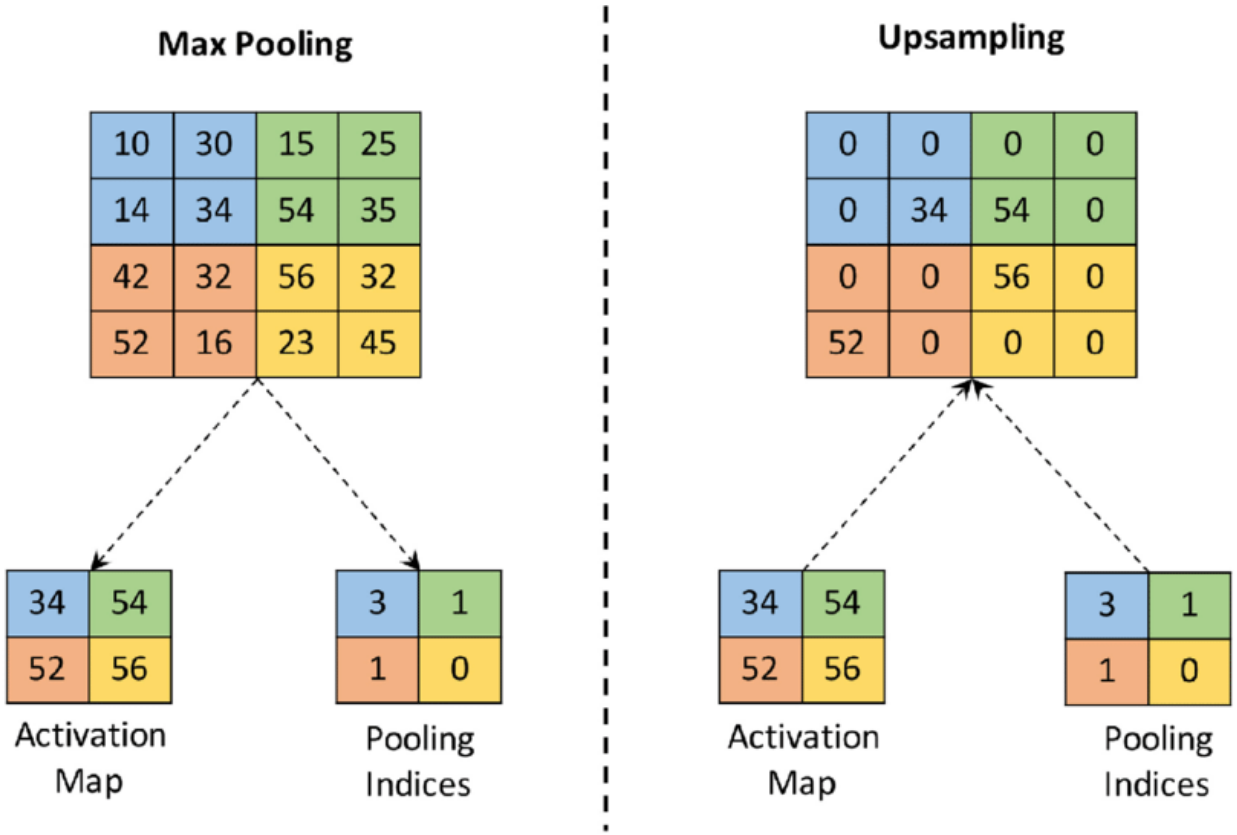
- Bilinear Upsampling




```
tf.keras.layers.UpSampling2D(
        size=(2, 2), data_format=None,
        interpolation="bilinear")
```

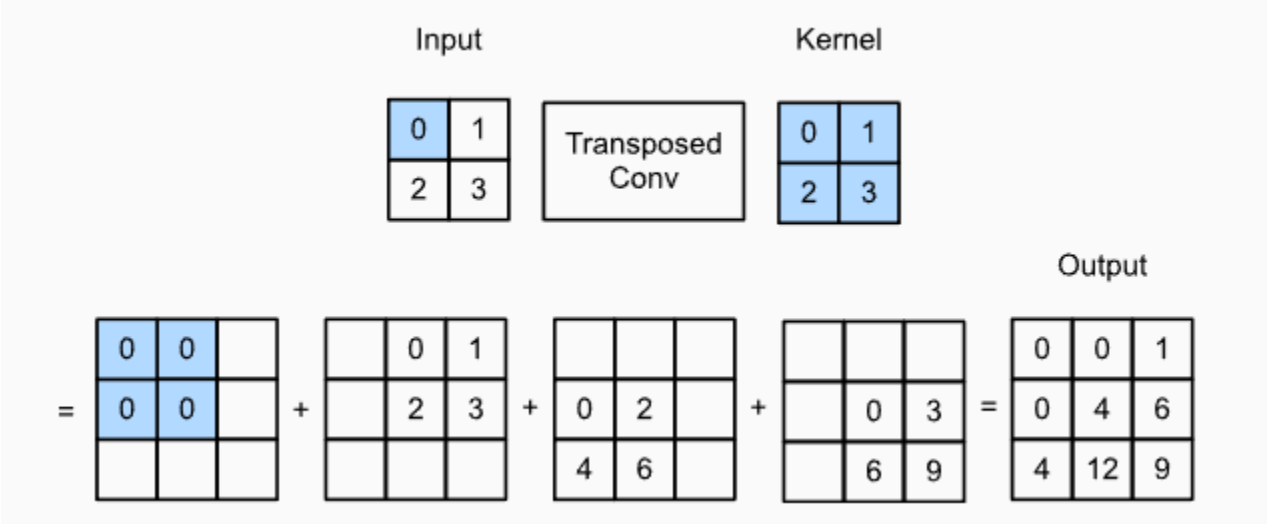
Upsampling

- Max Unpooling



Upsampling

- Transposed Convolutions




```
tf.keras.layers.Conv2DTranspose(
    filters,
    kernel_size,
    strides=(1, 1),
    padding="valid", ...
)
```

Pixelwise classification

- Đánh số các lớp: 0, 1, …
- Dùng 1×1 đề giảm số channel thành 1
- Dùng hàm softmax để classification

U-NET

U-Net

- U-Net
  - Kiến trúc có hình dáng chữ U Nên gọi là U-Net
  - U-Net là một kỹ thuật phân đoạn ngữ nghĩa

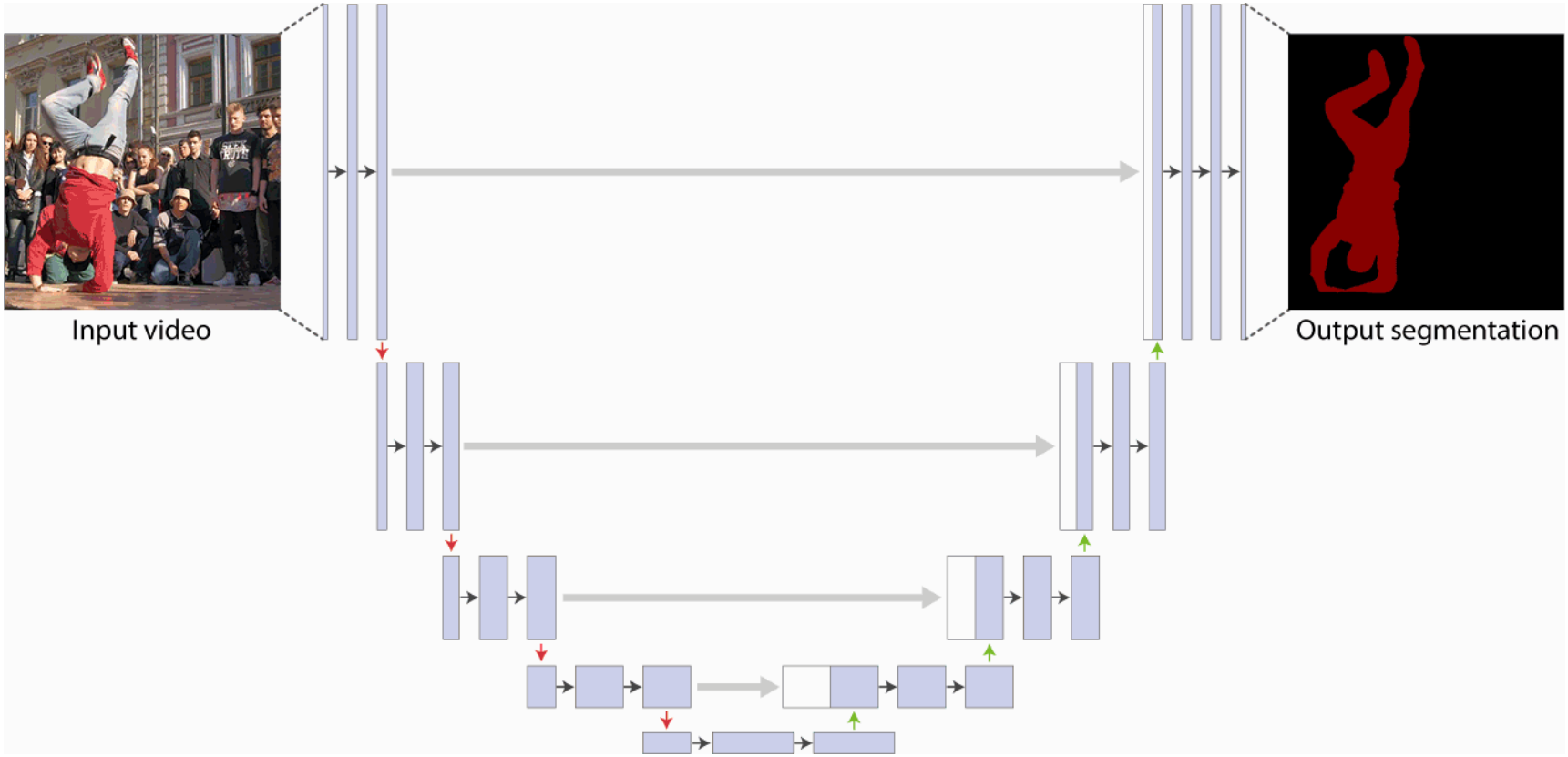
Olaf Ronneberger

- Tác giả
  - Olaf Ronneberger
  - Philipp Fischer
  - Thomas Brox

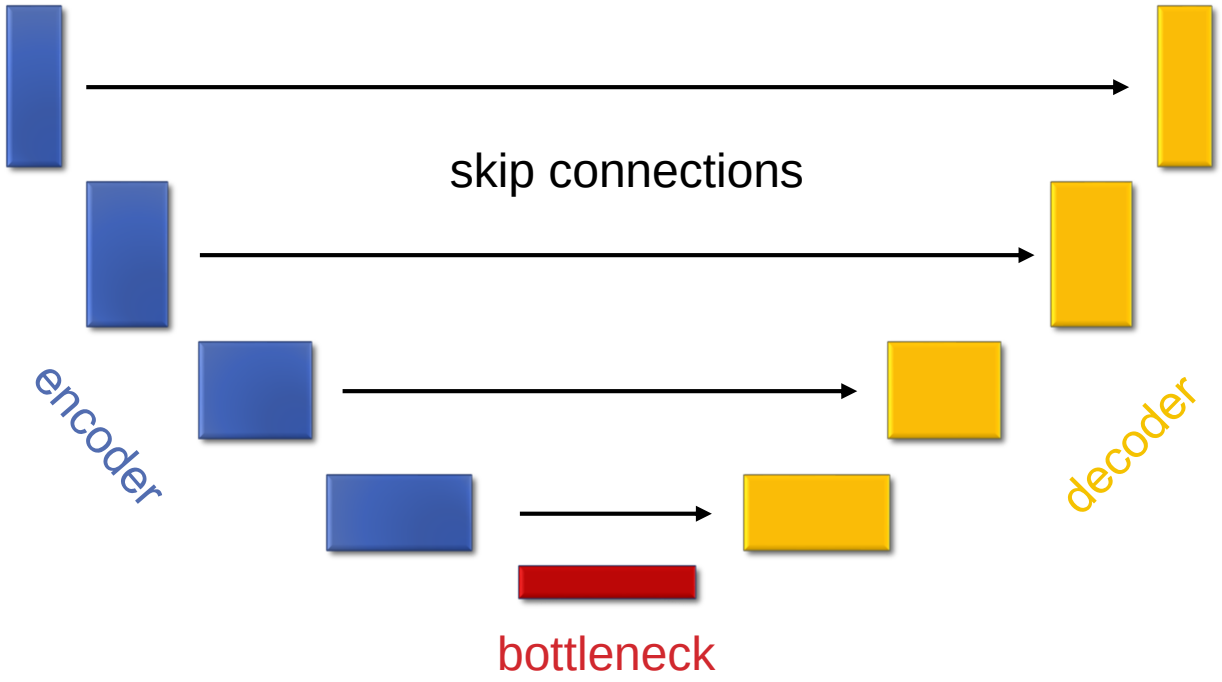
- Bài báo
  - U-Net: Convolutional Networks for Biomedical Image Segmentation, 2015

- Link
  - https://arxiv.org/abs/1505.04597

U-Net

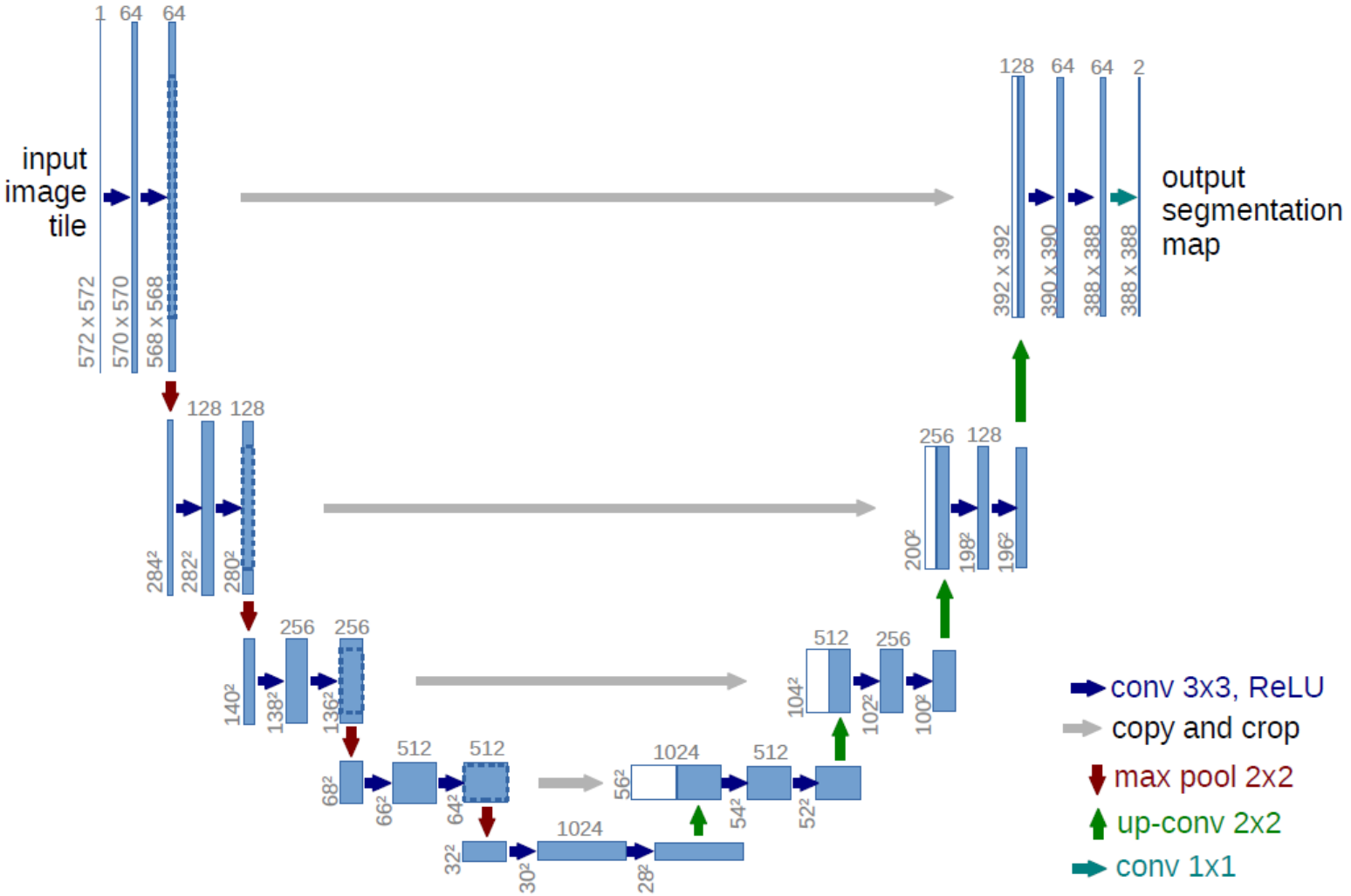


Các thành phần của U-Net



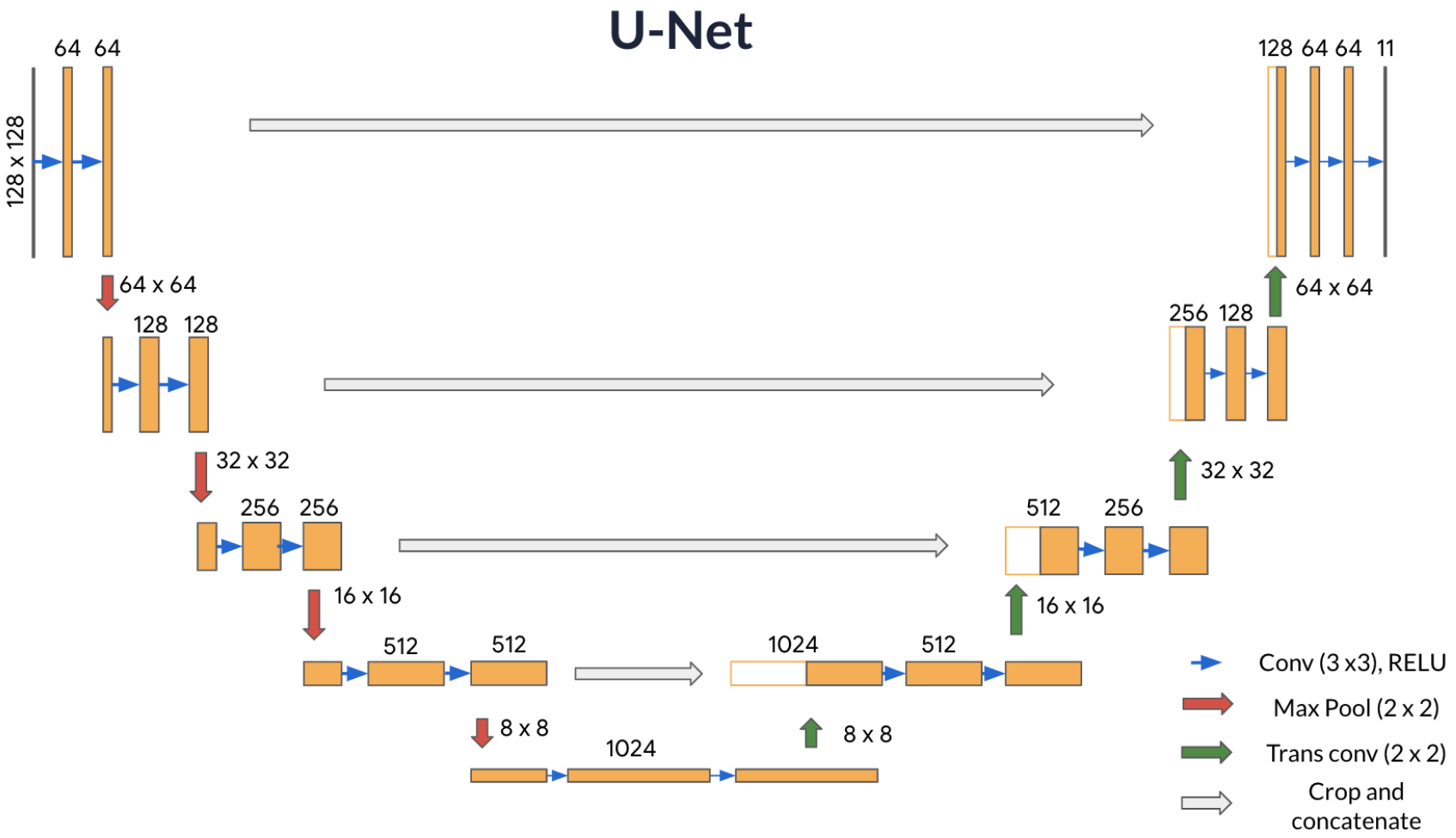
1. Encoder (để downsampling)
2. Decoder (để upsampling)
3. Skip connections (để xây dựng xâu)
4. Bottleneck (tăng hiệu quả)

U-Net

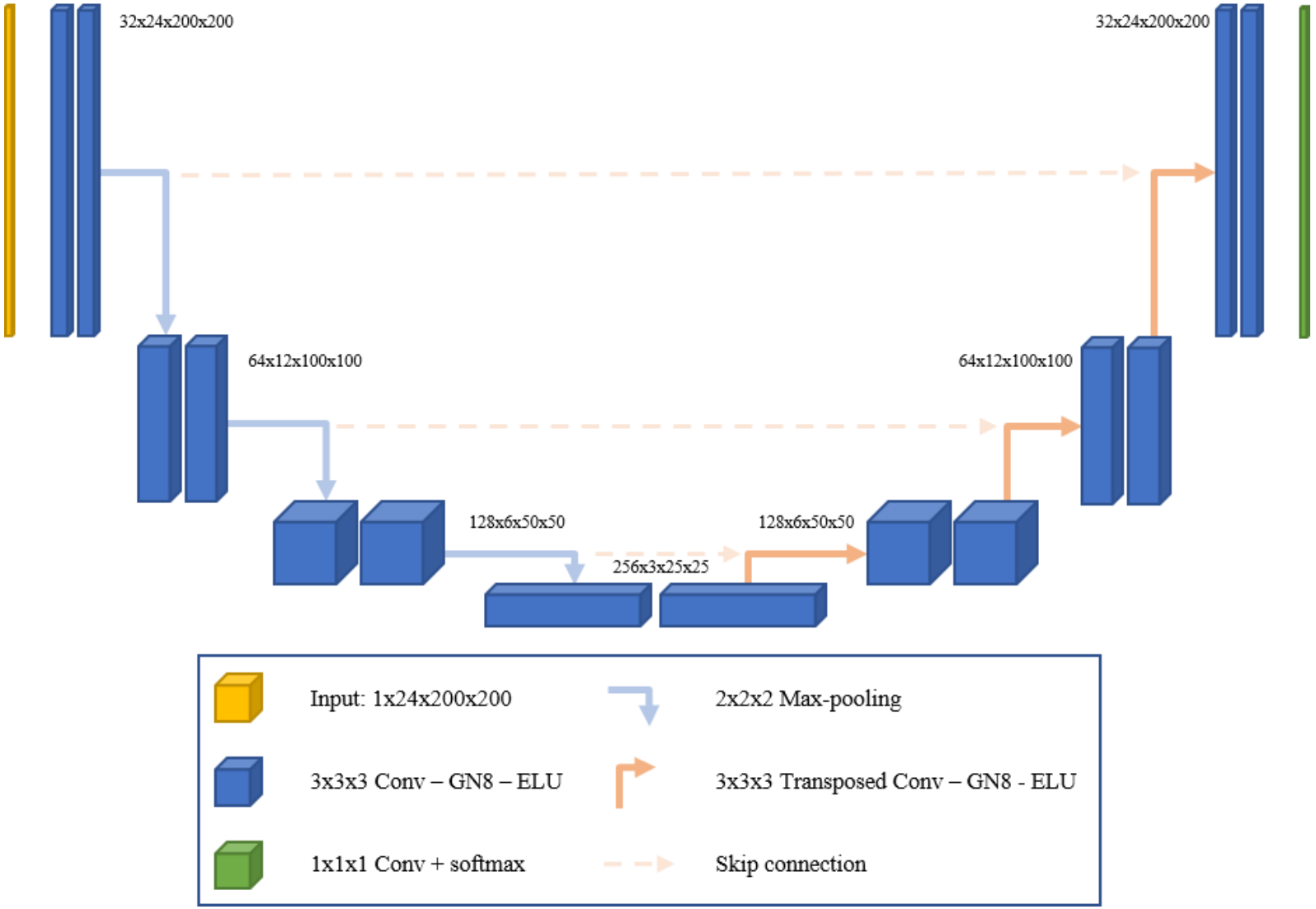


U-Net: Convolutional Networks for Biomedical Image Segmentation, Olaf Ronneberger, …, 2015

U-Net


U-Net
64 64
128 x 128
128 64 64 11
64 x 64
128 128
64 x 64
256 128
32 x 32
256 256
32 x 32
512 256
16 x 16
512 512
16 x 16
1024 512
8 x 8
1024
8 x 8
Conv (3 x3), RELU
Max Pool (2 x 2)
Trans conv (2 x 2)
Crop and concatenate

U-Net


32x24x200x200
32x24x200x200
64x12x100x100
64x12x100x100
128x6x50x50
128x6x50x50
256x3x25x25
Input: 1x24x200x200
2x2x2 Max-pooling
3x3x3 Conv – GN8 – ELU
3x3x3 Transposed Conv – GN8 - ELU
1x1x1 Conv + softmax
Skip connection

U-Net Block

```
def double_conv_block(x, num_filters):
   # Conv2D then ReLU activation
   x = layers.Conv2D (
               num_filters, 3,
               padding = "same",
               activation = "relu",
               kernel_initializer = "he_normal")(x)

   # Conv2D then ReLU activation
   x = layers.Conv2D (
               num_filters, 3,
               padding = "same",
               activation = "relu",
               kernel_initializer = "he_normal")(x)
   return x
```

U-Net Block

```
def downsample_block(x, num_filters):
    f = double_conv_block(x, num_filters)
    p = layers.MaxPool2D(2)(f)
    p = layers.Dropout(0.3)(p)
    return f, p
```

```
def upsample_block(x, conv_features, num_filters):
    # upsample
    x = layers.Conv2DTranspose(n_filters, 3, 2, padding="same")(x)

    # concatenate
    x = layers.concatenate([x, conv_features])

    # dropout
    x = layers.Dropout(0.3)(x)

    # Conv2D twice with ReLU activation
    x = double_conv_block(x, num_filters)
    return x
```

U-Net model

```
def build_unet_model()
    # inputs
    inputs = layers.Input(shape=(128,128,3))

    # encoder: contracting path - downsample
    # 1 - downsample
    f1, p1 = downsample_block(inputs, 64)
    # 2 - downsample
    f2, p2 = downsample_block(p1, 128)
    # 3 - downsample
    f3, p3 = downsample_block(p2, 256)
    # 4 - downsample
    f4, p4 = downsample_block(p3, 512)
    # 5 - bottleneck
    bottleneck = double_conv_block(p4, 1024)
```

U-Net model

```
def build_unet_model()
    # ...
    # decoder: expanding path - upsample
    # 6 - upsample
    u6 = upsample_block(bottleneck, f4, 512)
    # 7 - upsample
    u7 = upsample_block(u6, f3, 256)
    # 8 - upsample
    u8 = upsample_block(u7, f2, 128)
    # 9 - upsample
    u9 = upsample_block(u8, f1, 64)
    # outputs
    outputs = layers.Conv2D(3, 1, padding="same", activation = "softmax")(u9)

    # unet model with Keras Functional API
    unet_model = tf.keras.Model(inputs, outputs, name="U-Net")
    return unet_model
```

DENSENET PRETRAINED

DenseNet Pretrained

- VGG được train trên ImageNet (>14 triệu ảnh + 1000 lớp)
- Input: ảnh $224 \times 224 \times 3$
- Output: vector 1000 giá trị

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
DenseNet121	33	75.0%	92.3%	8.1M	242	77.1	5.4
DenseNet169	57	76.2%	93.2%	14.3M	338	96.4	6.3
DenseNet201	80	77.3%	93.6%	20.2M	402	127.2	6.7

DenseNet Pretrained

```
tf.keras.applications.DenseNet121(
    include_top=True,
    weights="imagenet",
    input_tensor=None,
    input_shape=None,
    pooling=None,
    classes=1000,
    classifier_activation="softmax",
)
```

```
tf.keras.applications.DenseNet169(
    include_top=True,
    weights="imagenet",
    input_tensor=None,
    input_shape=None,
    pooling=None,
    classes=1000,
    classifier_activation="softmax",
)
```

```
tf.keras.applications.DenseNet201(
    include_top=True,
    weights="imagenet",
    input_tensor=None,
    input_shape=None,
    pooling=None,
    classes=1000,
    classifier_activation="softmax",
)
```

Tóm tắt

- Bài toán segmentation
- Kiến trúc chung cho segmentation
- U-Net