Understanding U-Net

Minh Tran · Towards Data Science · November 15, 2022

Reader-formatted course copy of the public article. Original: https://towardsdatascience.com/understanding-u-net-61276b10f360/

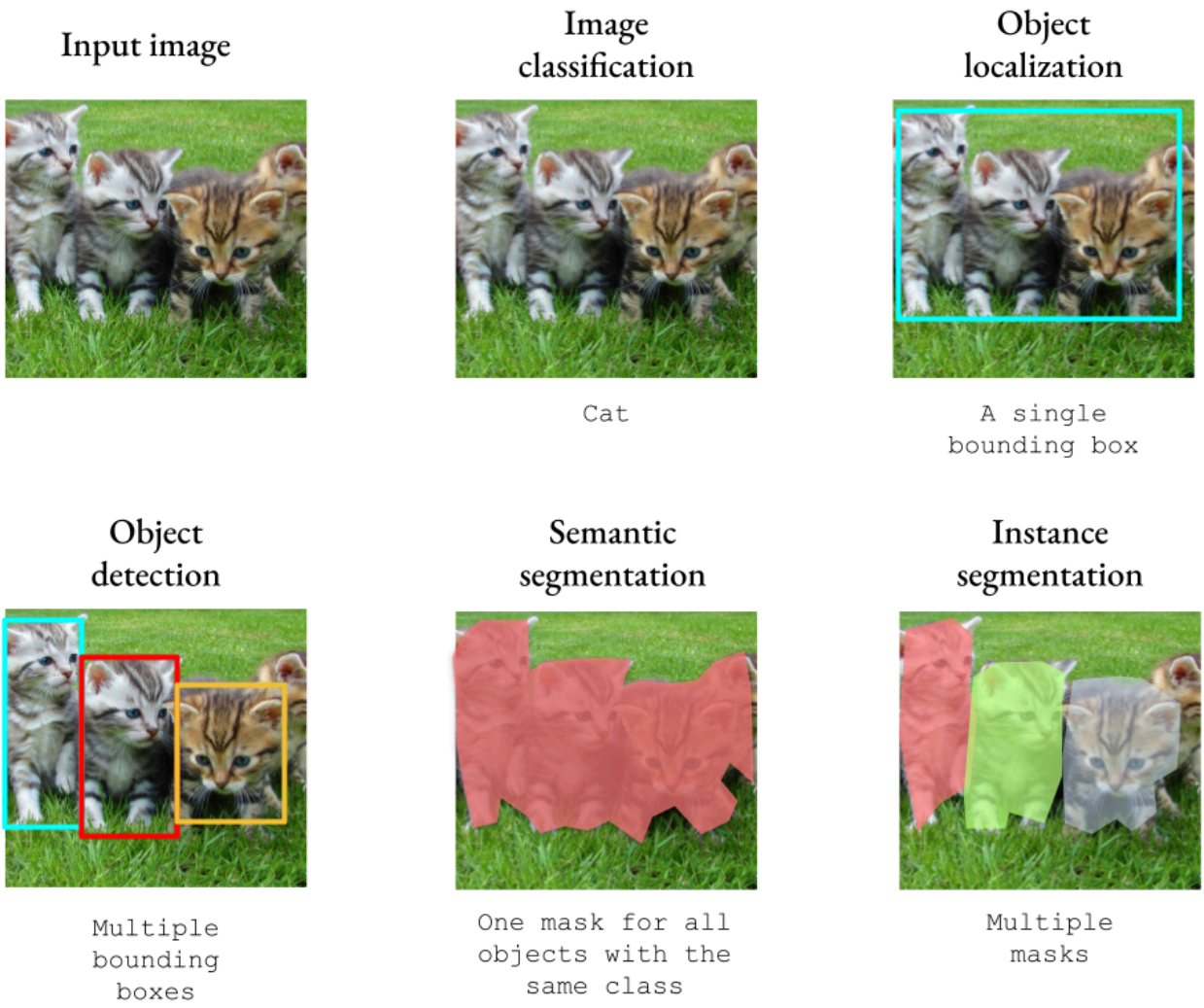
U-Net has become the go-to method for image segmentation. But how did it come to be?

Photo by Grillot edouard on Unsplash

Table of content

The task at hand

U-Net is developed for the task of semantic segmentation. When a neural network is fed images as inputs, we can choose to classify objects either generally or by instances. We can predict what object is in the image (image classification), where all objects are located(image localization/semantic segmentation), or where individual objects are located (object detection/instance segmentation). The figure below shows differences between these computer vision tasks. To simplify the matter, we only consider classification for only one class and one label (binary classification).



Different computer vision tasks for single-label. The original picture is under Creative Common license at Pexels.com. Edit is made by the author.

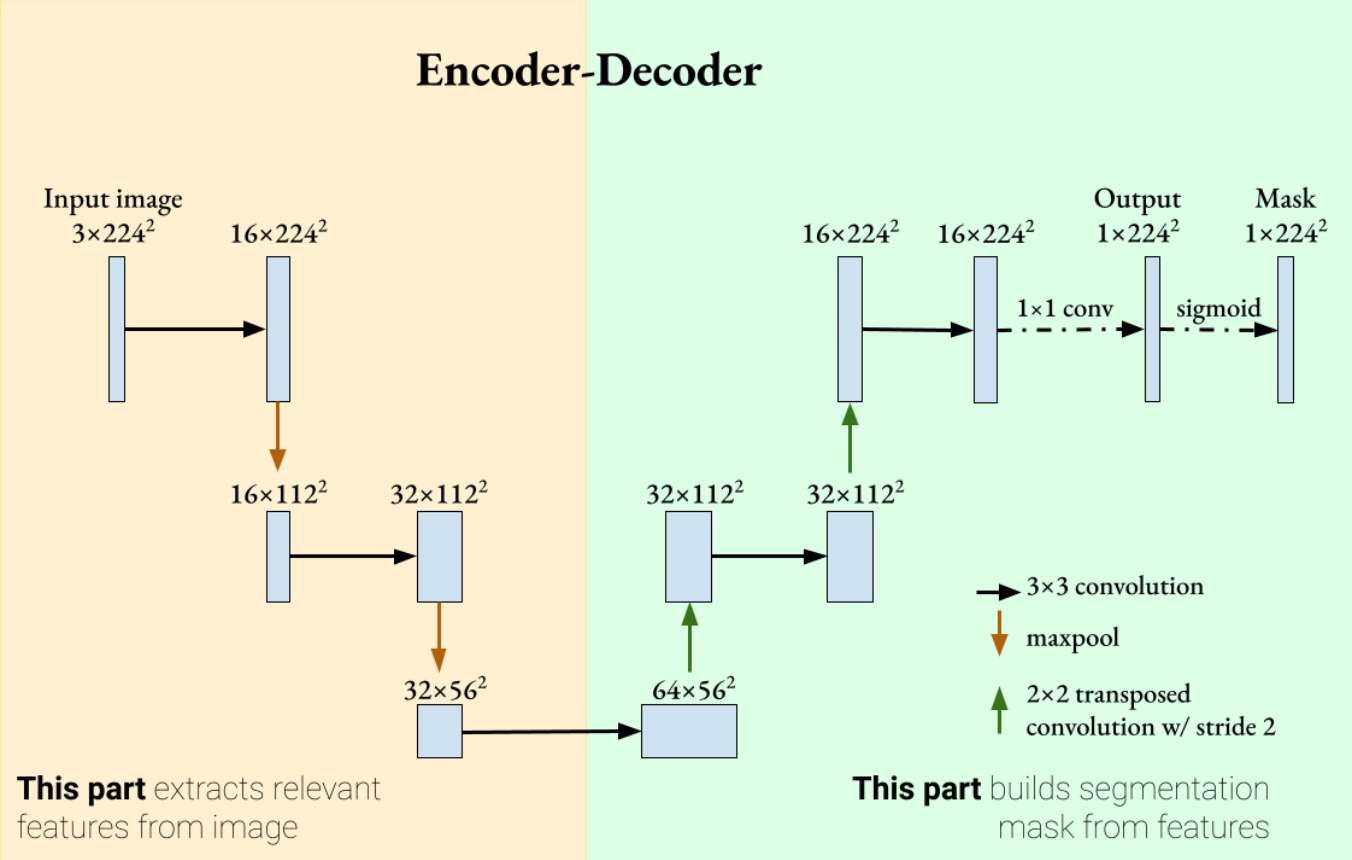
In classification task, we output a vector of size k, where k is the number of classes. In detection tasks, we need to output the vector `x, y, height, width, class`, which define bounding boxes. But in segmentation tasks, we need to output an image with the same dimension as the original input. This represents quite an engineering challenge: How can a neural network extract relevant features from the input image, and then project them into segmentation masks?

Encoder-Decoder

If you are not familiar with encoder-decoder, I recommend you to read this article:

> **Understanding Latent Space in Machine Learning**

The reason why encoder-decoders are relevant because they produce outputs similar to what we want: output that have the same dimension as the input. Can we apply the concept of encoder-decoder to images segmentation? We can generate a one-dimensional binary mask and train the network using cross-entropy loss. Our network consists of two parts: the **encoder** which extracts relevant features from images, and the **decoder** part which takes the extracted features and reconstructs a segmentation mask.



An encoder-decoder network for image segmentation. Image by the author.

In the encoder part, I used convolutional layers, followed by `ReLU` and `MaxPool` as the feature extractors. In the decoder part, I transposed convolution to increase the size of the feature map and decrease the number of channels. I used padding to keep the size of the feature maps the same after convolution operations.

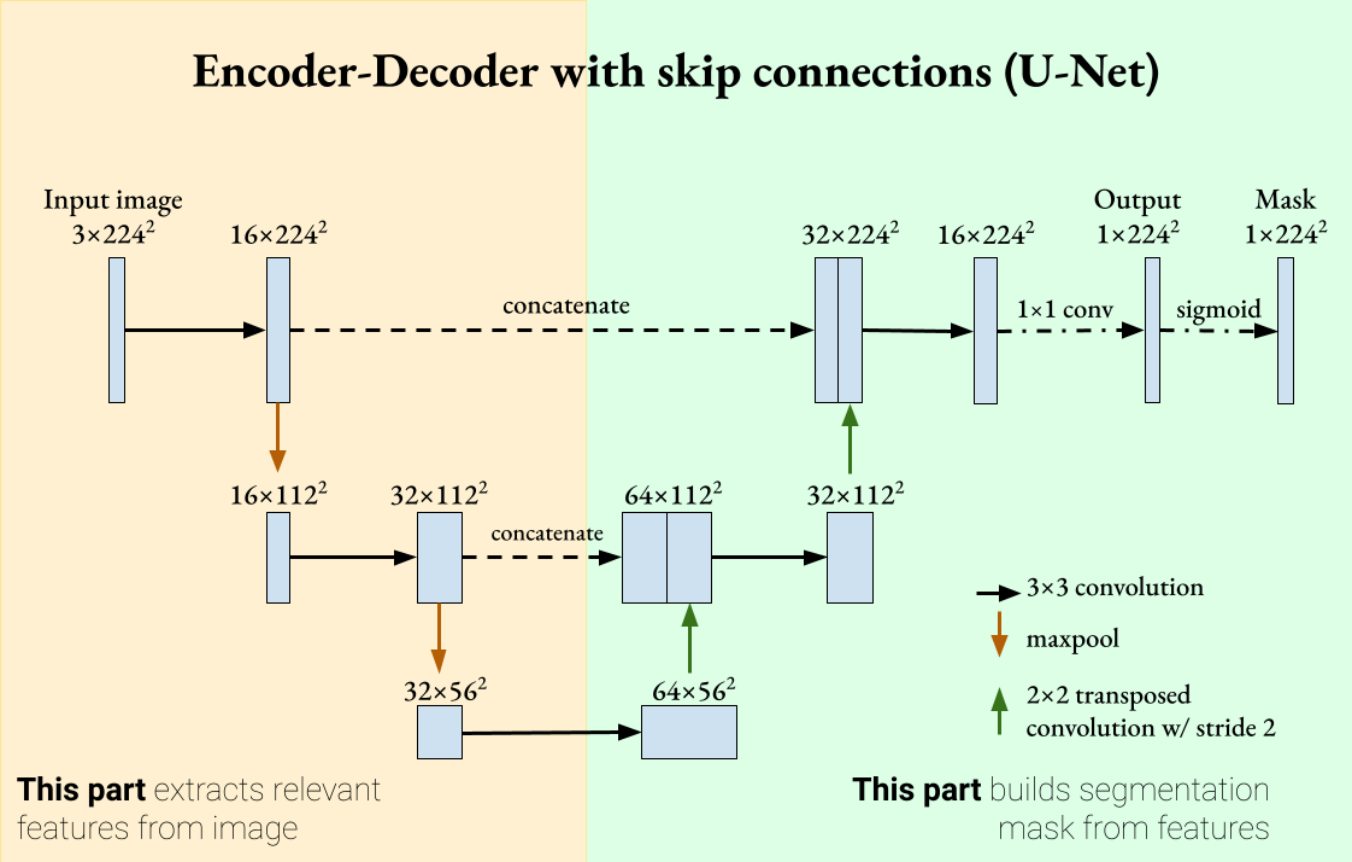
One thing you might notice is that unlike classification networks, this network doesn't have a fully connected / linear layer. This is an example of a **fully convolutional network** (FCN). FCN has been shown to work well on segmentation tasks, starting with Shelhamer *et al.* paper "Fully Convolutional Networks for Semantic Segmentation" [1].

However, this network has a problem. As we expand the number of layers in the encoder and decoder layers, we effectively "shrink" the feature map more and more. As such, the encoder may discard features that are more detailed in favor of more general features. If we are dealing with medical image segmentation, every pixels classified as diseased/normal can be important. How can we make sure that this encoder-decoder network take in features that are both general and detailed?

Skip connections

> **Introduction to ResNets**

Because deep neural networks can "forget" certain features as it pass information through successive layers, skip connections can reintroduce them to make learning stronger. Skip connection was introduced in Residual Network (ResNet) and showed classification improvements as well as smoother learning gradients. Inspired by this mechanism, we can add skip connections to U-Net such that every decoder incorporate the feature map from its corresponding encoder. This is a defining feature of **U-Net**.



U-Net is an encoder-decoder segmentation network with skip connections. Image by the author.

U-Net has two defining qualities:

1. An encoder-decoder network that extract more general features the deeper it goes.
2. A skip connection that reintroduces detailed features into the decoder.

These two qualities means that U-Net can segment using features that are both detailed and general. U-Net was originally introduced for biomedical image processing, where the accuracy of segmentation is very important [2].

Implementation details

Photo by Pavel Neznanov on Unsplash

The previous sections gave a very general overview of U-Net and why it works. However, details stand between general understanding and actual implementation. Here, I gave an overview of some implementation choices for U-Net.

Loss functions

Because the target are binary masks (pixel value is 1 when pixel contains object), a common loss function to compare output with the ground truth is the **categorical cross-entropy loss** (or binary cross-entropy in the case of single label).

$$Loss = -\sum_{c=1}^{k} truth_c \odot \log(\text{softmax}(output_c)), \quad \text{for k classes}$$

$$\text{softmax}(output_c) = \frac{\exp(output_c)}{\sum_{j=1}^{k} \exp(output_j)}, \quad \text{for class c between 1 and k}$$

In the original U-Net paper, an additional weight is added to the loss function. This weight parameter does two things: it compensates for class imbalance, and it gives higher importance to **segmentation borders**. In many implementations of U-Net that I've found online, this additional weight factor is not often used.

Another loss function commonly seen is the **dice loss**. Dice loss measures how similar two set of images are by comparing their intersection area with their total area. Note that dice loss is not the same as Intersection-over-Union (IOU). They measures similar things, but they have different denominator. The higher the dice coefficient, the lower the dice loss.

$$Loss = 1 - \frac{2\sum_{i=1}^{N} \text{softmax}(output_c) \odot truth_c + \epsilon}{\sum_{i=1}^{N} truth_c + \sum_{i=1}^{N} \text{softmax}(output_c) + \epsilon}, \text{ for class c between 1 and k}$$

Here, an epsilon term is added to avoid division by 0 (epsilon is typically 1). Some implementations, such as the one in Milletari *et al.,* squared the pixel values in the denominator before summing them [3]. Compared to cross-entropy loss, dice loss is very robust against imbalanced segmentation mask, which is typical in biomedical image segmentation tasks.

Up-sampling methods

Another detail is the choice of up-sampling method for the decoder. Here are some common methods:

Bi-linear interpolation. This method predicts the output pixel using linear interpolation. Usually, up-scaling through this method is followed by a convolution layer.

Max-Unpooling. This method is the opposite of Max-pooling. It uses the indices of the maxpool operation and populate these indices with maximum value. All other values are set to 0. Typically, a convolution layer follows max-unpooling to "smooth-out" all the missing values.

Deconvolution / Transpose convolution. Many blog post has been written about deconvolution. I recommend this article as a good visual guide.

> **An Introduction to different Types of Convolutions in Deep Learning**

Deconvolution has two steps: add padding to each pixel in the original image, then apply convolution. In the original U-Net, a 2x2 transposed convolution with stride 2 is used to change both the spatial resolution and the channel depth.

Pixel Shuffling. This method was seen in super-resolution networks such as SRGAN. To start, we use convolution to go from `C x H x W` feature map to `(Cr^2) x H x W`. Then, pixel shuffle will take this and "reshuffle" the pixels in a mosaic fashion to produce output of size `C x (Hr) x (Wr)`.

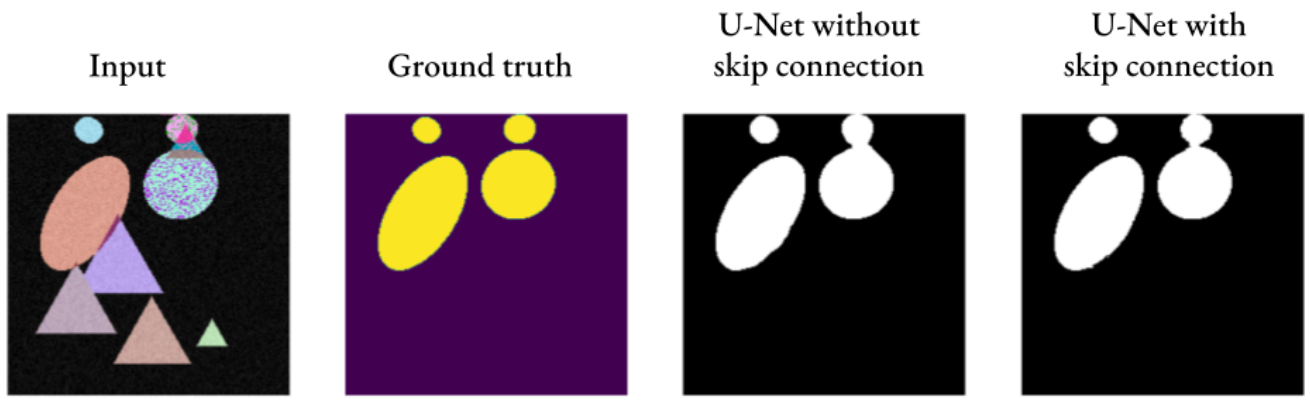
To pad or not to pad?

Convolution layer, with a kernel larger than 1x1 and without padding, will produce output that is smaller than the input. This is a problem for U-Net. Recall in the U-Net figure in earlier section, we concatenate part of the image with its decoded counterpart. If we don't use padding, then the decoded image will have smaller spatial dimension compared to the encoded image.

However, the original U-Net paper didn't use padding. Even though no justification were given, I believe it was because the authors didn't want to introduce segmentation errors at the image margin. Instead, they center-cropped the encoded image before concatenation. For an image with input size `572 x 572`, the output will be `388 x 388`, a ~50% loss. If you want to run U-Net without padding, you need to run it multiple times on overlapping tiles to get the full segmentation image.

U-Net in action

Here, I implemented a very simple U-Net-like network to segment only ellipses. The U-Net is only 3 layers deep, uses same padding, and binary cross entropy loss. More complicated networks can use more convolution layers at each resolution, or extending the depth as see fit.



The task: to segment ellipses and not segment any other shapes. Figure is generated by the author.

As we can see, the U-Net can produce acceptable segmentation even without skip connections, but the added skip connections can introduce finer details (see the join between the two ellipses on the right).

Conclusion

If I were to explain U-Net in one sentence, it would be that U-Net is like an encoder-decoder for images, but with skip connections to make sure fine details are not lost. U-Net are used often in many segmentation tasks, and in recent years have made their ways onto image generation tasks as well.

If you want to see the code that I used to produce figures and train my U-Net, here is the Github link. Happy coding!

> If you enjoy reading this article and would like to read more similar ones in the future, consider following me on Medium or Linkedin.

References:

[1] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

[2] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." *International Conference on Medical image computing and computer-assisted intervention*. Springer, Cham, 2015.

[3] Milletari, Fausto, Nassir Navab, and Seyed-Ahmad Ahmadi. "V-net: Fully convolutional neural networks for volumetric medical image segmentation." *2016 fourth international conference on 3D vision (3DV)*. IEEE, 2016.