U-Net: Convolutional Networks for Biomedical Image Segmentation

Contents

Figures

UNet Introduction

UNet, a convolutional neural network dedicated for biomedical image segmentation, was first designed, and applied in 2015 by (Ronneberger, Fischer and Brox, 2015). In general, the use cases for a typical convolutional neural network focuses on image classification tasks, where the output to an image is a single class label, however in biomedical image visual tasks, it requires not only to distinguish whether there is a medical condition, but also to localize the area of infection i.e., a class label is supposed to be assigned to each pixel.

Segmentation - A Bio-Medical use case

The task of segmentation in biomedical images typically deals with partitioning the image into multiple regions representing anatomical objects and regions of interest. Segmenting a biomedical image comes with significant technical challenges such as heterogeneous pixel intensities, noisy/ill-defined boundaries, and irregular shapes with high variability. A typical use case is the task to detect the presence of cancerous cells during the early stages. Cancer of any type is always considered fatal if not detected in its early stages. Detecting cancerous cells as quickly as possible can potentially reduce the risk and help recover from the condition. The shape, size and morphology of the cancerous cells plays a vital role in determining the severity. Cancerous cells have a darker & larger nucleus than a normal cell. Detecting a cancer cell in a microscopic image requires segmenting the image into multiple regions. Then filtering out those regions that are representative of cancer cells.

Image Segmentation techniques help in localizing the region of abnormality in a more granular way to extract more meaningful information. The goal in any biomedical image segmentation task is to achieve accurate segmentation while ensuring the other details and information of the cellular anatomy are kept intact and not lost during the process.

UNet helps in achieving accurate segmentation in the biomedical images while maintaining the high resolution input features which help in achieving better localization.

UNET – Network Architecture

The complete network design is seen in the figure below -

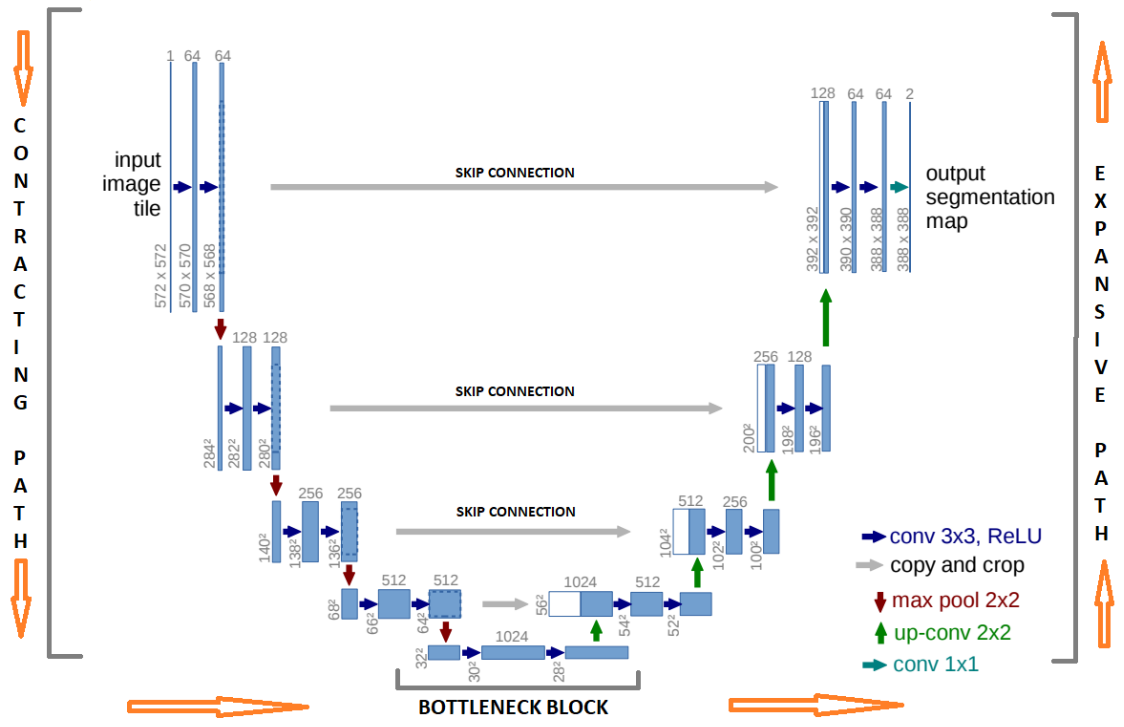

Figure 1 - UNet Architecture

Network Components

The UNet network model has 3 parts:

- The Contracting/Downsampling Path.
- Bottleneck Block.
- The Expansive/Upsampling Path.

Contracting Path

The contracting path starts with the input image undergoing two 3x3 unpadded convolutions (denoted by the blue right headed arrows) in sequence, each followed by a rectified linear unit and a 2x2 max pooling operation with stride 2 (denoted by the downward red arrow) for downsampling. At each downsampling step the depth of the image is increased by doubling the number of feature channels. The following components from Figure 1 - UNet Architecture represents the contracting path. The input image originally chosen in the paper is of shape - *(572, 572, 1)*

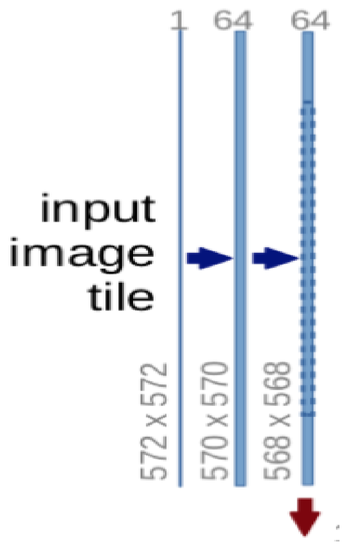

Figure 2 - Input layer

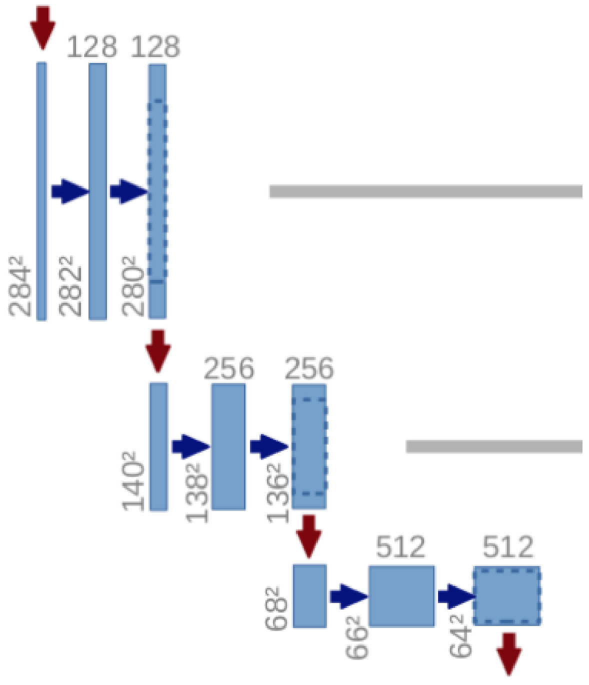

Figure 3 - Contracting Path

Code Implementation

The code implementation of the contracting blocks would look like -

*****Note**: The statements in the code blocks are colored as per the arrowhead colors seen in the complete network architecture in Figure 1 - UNet Architecture*

```
conv = Conv2D(filters = filters, kernel_size = kernel_size, activation = tf.nn.relu,
                    padding = padding)(input_layer)
conv = Conv2D(filters = filters, kernel_size = kernel_size, activation = tf.nn.relu,
                    padding = padding)(conv)
pool = MaxPooling2D(pool_size = 2, strides = 2)(conv)
```

Table 1 - Contracting Block Code Implementation

There are 4 such down sampling blocks implemented along the contracting path.

Bottleneck Block

The bottleneck block connects the contracting and the expansive paths. This block performs two unpadded convolutions each with 1024 filters and prepares for the expansive path. There is no pooling operation involved in this part of the network.

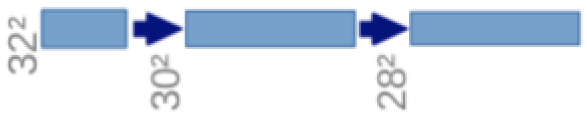

Figure 4 - Bottleneck Block

Code Implementation

The code implementation of the contracting blocks would look like -

```
conv = Conv2D(filters = filters, kernel_size = kernel_size, activation = tf.nn.relu,
                    padding = padding)(input_layer)
conv = Conv2D(filters = filters, kernel_size = kernel_size, activation = tf.nn.relu,
                    padding = padding)(conv)
```

Table 2- Bottleneck Block Code Implementation

Expansive Path

Every step in the expansive path consists of an up sampling of the feature map followed by a 2x2 *up-convolution using transposed convolutions* (denoted by the upward green arrows), a concatenation with the correspondingly feature map from the contracting path. The skip connections provide features from earlier layers that are sometimes lost due to the depth of the network. It is followed by two 3x3 convolutions, each followed by a ReLU (denoted by the blue right headed arrows). Transposed convolution is an up sampling technique to expand the size of images. Figure 5 - Expansive Path shows the expansive path of the network.

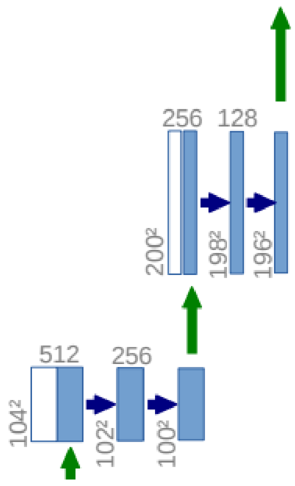

Figure 5 - Expansive Path

Code Implementation

The code implementation of the contracting blocks would look like -

```
transConv = Conv2DTranspose(filters = filters, kernel_size = (2, 2), strides = 2,
                                                    padding = padding)(input_layer)
cropped = crop_tensor(skip_conn_layer, transConv)
concat = Concatenate()([transConv, cropped])
up_conv = Conv2D(filters = filters, kernel_size = kernel_size, padding = padding,
                         activation = tf.nn.relu)(concat)
up_conv = Conv2D(filters = filters, kernel_size = kernel_size, padding = padding,
                         activation = tf.nn.relu)(up_conv)
```

Table 3 - Expansive Block Code Implementation

Skip Connections

The skip connections from the contracting path are concatenated with the corresponding feature maps in the expansive path. These connections ensure the higher resolution features are maintained to better localize and learn representations from the input image. For a proper connection channel to be established, *the feature maps from the contracting paths are first cropped to be of the same dimension as the corresponding feature maps in the expansive path*. They also help in recovering any spatial information that could have been lost during downsampling. Figure 6 - Skip Connection shows a connection from the contracting to the expansive path. As mentioned by the authors in the original paper, *cropping is necessary due to the loss of border pixels in every convolution.*

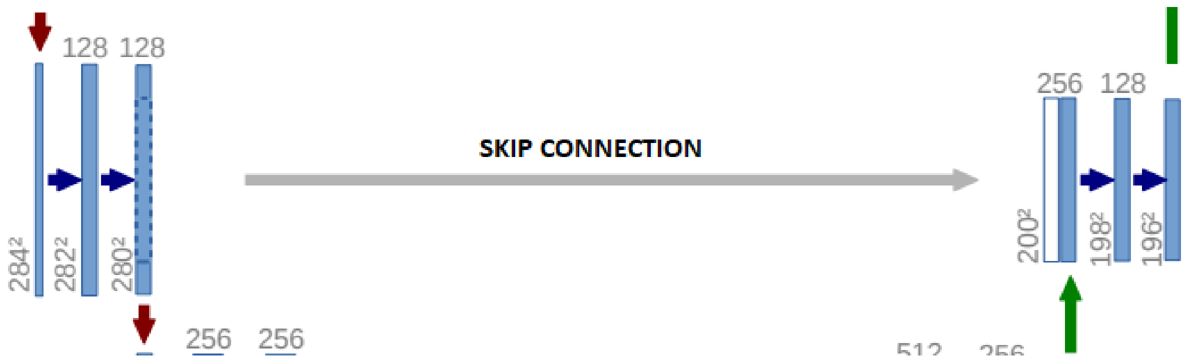

Figure 6 - Skip Connection

Code Implementation

The code implementation of the contracting blocks would look like -

```
cropped = crop_tensor(skip_conn_layer, transConv)
concat = Concatenate()([transConv, cropped])
```

Table 4 - Skip Connection Code Implementation

Output Layer

The final (output) layer is a 1x1 convolution is used to map each (64 component) feature vector to the desired number of classes.

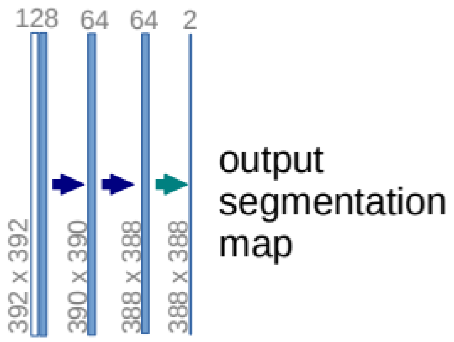

Figure 7 - Output Block

Network as Designed

The UNet model as designed comes out as seen below in Figure 8 - UNet as Designed

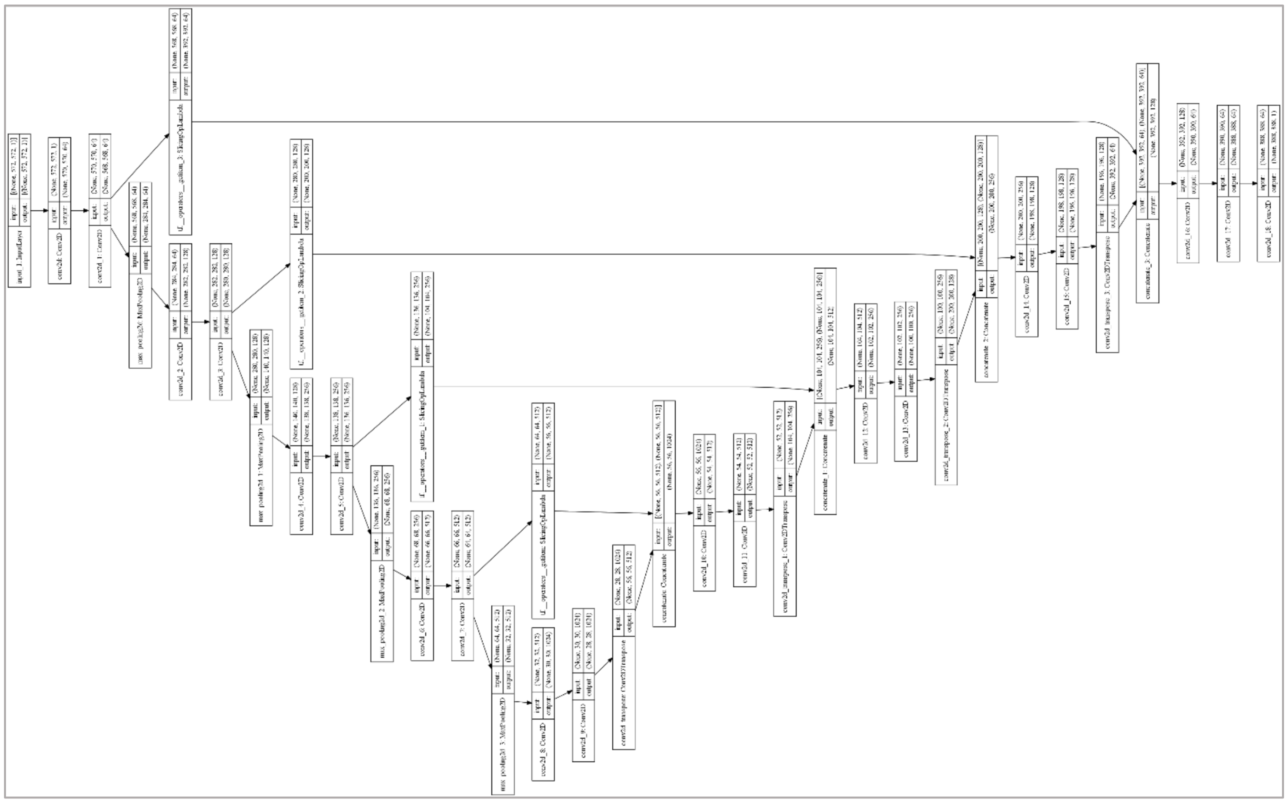

Figure 8 - UNet as Designed

The expansive path of the network has a large number of feature channels, which allows the network to propagate context information to higher resolution layers. Due to this, the expansive path is almost symmetric to the contracting path, yielding a U-shaped architecture. The network does not have any fully connected layers and only uses the valid part of each convolution, i.e., the segmentation map only contains the pixels, for which the full context is available in the input image. The pixels in the border region of the image are classified by extrapolating the missing context via mirroring the input image.

Further Reading

1. *UNet++: A Nested U-Net Architecture for Medical Image Segmentation*. The original paper is available at https://arxiv.org/abs/1807.10165.
2. *UNet++: Redesigning Skip Connections to Exploit Multiscale Features in Image Segmentation*. The original paper is available at https://arxiv.org/abs/1912.05074v2.
3. *Attention U-Net: Learning Where to Look for the Pancreas*. The original paper is available at https://arxiv.org/abs/1804.03999.
4. *TernausNet: U-Net with VGG11 Encoder Pre-Trained on ImageNet for Image Segmentation.* The original paper is available at *https://arxiv.org/abs/1801.05746*.
5. *U-Net and its variants for medical image segmentation: theory and applications. The original paper is available at* https://arxiv.org/abs/2011.01118.

References

1. Ronneberger, O., Fischer, P. and Brox, T. (2015) 'U-Net: Convolutional Networks for Biomedical Image Segmentation', *CoRR*, abs/1505.0. Available at: http://arxiv.org/abs/1505.04597.

2. Implementing original U-Net from scratch using PyTorch by Abhishek Thakur. Available at https://www.youtube.com/watch?v=u1loyDCoGbE