UNet — Line by Line Explanation

Example UNet Implementation

Jeremy Zhang · Follow

Published in Towards Data Science
4 min read · Oct 18, 2019

UNet, evolved from the traditional convolutional neural network, was first designed and applied in 2015 to process biomedical images. As a general convolutional neural network focuses its task on image classification, where input is an image and output is one label, but in biomedical cases, it requires us not only to distinguish whether there is a disease, but also to localise the area of abnormality.

UNet is dedicated to solving this problem. **The reason it is able to localise and distinguish borders is by doing classification on every pixel, so the input and output share the same size.** For example, for an input image of size 2x2:

```
[[255, 230], [128, 12]]  # each number is a pixel
```

the output will have the same size of 2x2:

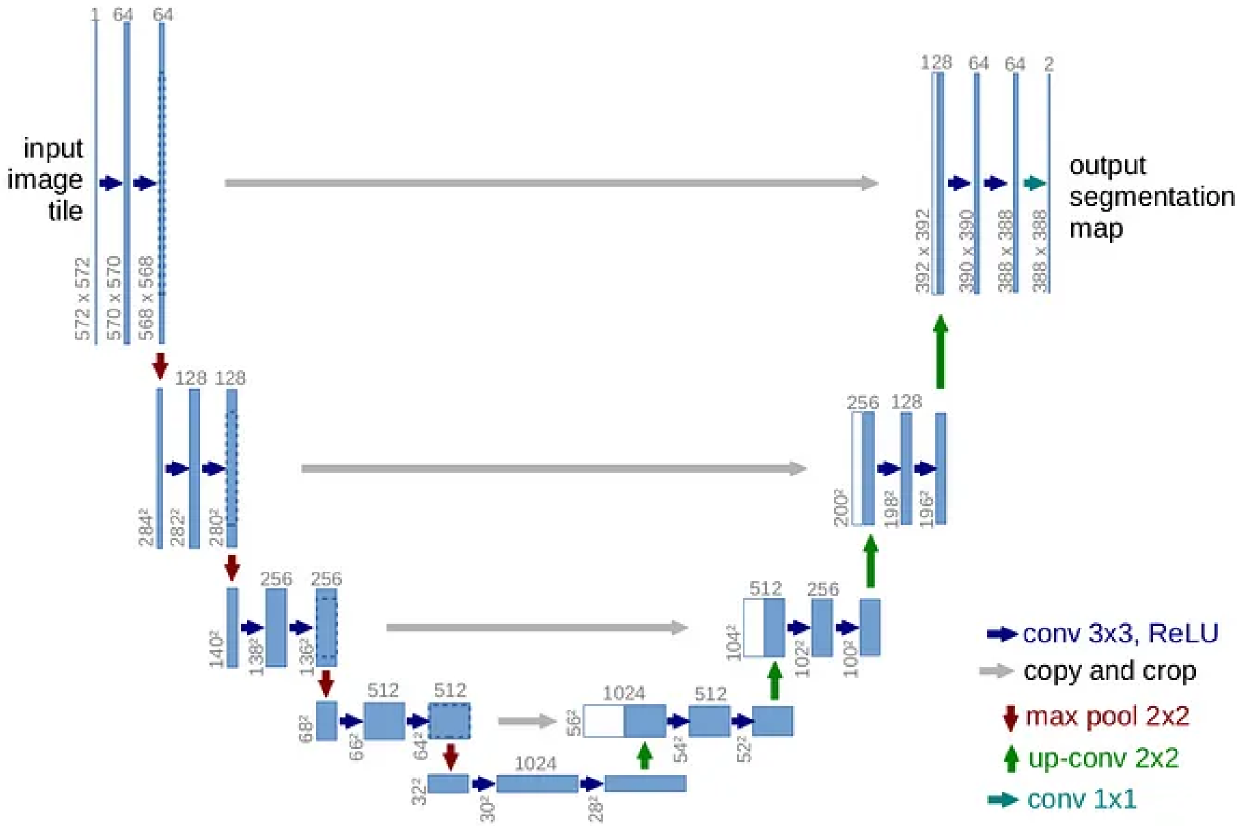
```
[[1, 0], [1, 1]]  # could be any number between [0, 1]
```

Now let’s get to the detail implementation of UNet. I will:

1. Show the overview of UNet
2. Breakdown the implementation line by line and further explain it

Overview

The network has basic foundation looks like:



UNet architecture

First sight, it has a "U" shape. The architecture is symmetric and consists of two major parts — the left part is called contracting path, which is constituted by the general convolutional process; the right part is expansive path, which is constituted by transposed 2d convolutional layers(you can think it as an upsampling technic for now).

Now let's have a quick look at the implementation:

```
def build_model(input_layer, start_neurons):
    conv1 = Conv2D(start_neurons * 1, (3, 3), activation="relu", padding="same")(input_
    conv1 = Conv2D(start_neurons * 1, (3, 3), activation="relu", padding="same")(conv1)
    pool1 = MaxPooling2D((2, 2))(conv1)
    pool1 = Dropout(0.25)(pool1)

    conv2 = Conv2D(start_neurons * 2, (3, 3), activation="relu", padding="same")(pool1)
    conv2 = Conv2D(start_neurons * 2, (3, 3), activation="relu", padding="same")(conv2)
    pool2 = MaxPooling2D((2, 2))(conv2)
    pool2 = Dropout(0.5)(pool2)

    conv3 = Conv2D(start_neurons * 4, (3, 3), activation="relu", padding="same")(pool2)
    conv3 = Conv2D(start_neurons * 4, (3, 3), activation="relu", padding="same")(conv3)
    pool3 = MaxPooling2D((2, 2))(conv3)
    pool3 = Dropout(0.5)(pool3)

    conv4 = Conv2D(start_neurons * 8, (3, 3), activation="relu", padding="same")(pool3)
    conv4 = Conv2D(start_neurons * 8, (3, 3), activation="relu", padding="same")(conv4)
    pool4 = MaxPooling2D((2, 2))(conv4)
    pool4 = Dropout(0.5)(pool4)

    # Middle
    convm = Conv2D(start_neurons * 16, (3, 3), activation="relu", padding="same")(pool4
    convm = Conv2D(start_neurons * 16, (3, 3), activation="relu", padding="same")(convm

    deconv4 = Conv2DTranspose(start_neurons * 8, (3, 3), strides=(2, 2), padding="same'
    uconv4 = concatenate([deconv4, conv4])
    uconv4 = Dropout(0.5)(uconv4)
    uconv4 = Conv2D(start_neurons * 8, (3, 3), activation="relu", padding="same")(uconv
    uconv4 = Conv2D(start_neurons * 8, (3, 3), activation="relu", padding="same")(uconv

    deconv3 = Conv2DTranspose(start_neurons * 4, (3, 3), strides=(2, 2), padding="same'
    uconv3 = concatenate([deconv3, conv3])
    uconv3 = Dropout(0.5)(uconv3)
    uconv3 = Conv2D(start_neurons * 4, (3, 3), activation="relu", padding="same")(uconv
    uconv3 = Conv2D(start_neurons * 4, (3, 3), activation="relu", padding="same")(uconv

    deconv2 = Conv2DTranspose(start_neurons * 2, (3, 3), strides=(2, 2), padding="same'
    uconv2 = concatenate([deconv2, conv2])
    uconv2 = Dropout(0.5)(uconv2)
    uconv2 = Conv2D(start_neurons * 2, (3, 3), activation="relu", padding="same")(uconv
    uconv2 = Conv2D(start_neurons * 2, (3, 3), activation="relu", padding="same")(uconv

    deconv1 = Conv2DTranspose(start_neurons * 1, (3, 3), strides=(2, 2), padding="same'
    uconv1 = concatenate([deconv1, conv1])
    uconv1 = Dropout(0.5)(uconv1)
    uconv1 = Conv2D(start_neurons * 1, (3, 3), activation="relu", padding="same")(uconv
    uconv1 = Conv2D(start_neurons * 1, (3, 3), activation="relu", padding="same")(uconv
```

```
    uconv1 = Conv2D(start_neurons * 1, (3, 3), activation="relu", padding="same")(ucon

    output_layer = Conv2D(1, (1,1), padding="same", activation="sigmoid")(uconv1)

    return output_layer

input_layer = Input((img_size_target, img_size_target, 1))
output_layer = build_model(input_layer, 16)
```

unet.py hosted with ❤ by GitHub view raw

The code is referred from a kernel of Kaggle competition, in general, most UNet follows the same structure.

Now let's break down the implementation line by line and maps to the corresponding parts on the image of UNet architecture.

Line by Line Explanation

Contracting Path

The contracting path follows the formula:

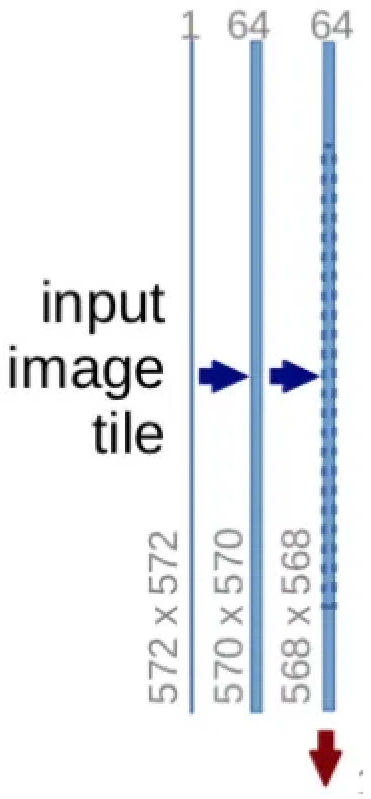
```
conv_layer1 -> conv_layer2 -> max_pooling -> dropout(optional)
```

So the first part of our code is:

```
conv1 = Conv2D(start_neurons * 1, (3, 3), activation="relu", padding="same")(input_laye
conv1 = Conv2D(start_neurons * 1, (3, 3), activation="relu", padding="same")(conv1)
pool1 = MaxPooling2D((2, 2))(conv1)
pool1 = Dropout(0.25)(pool1)
```

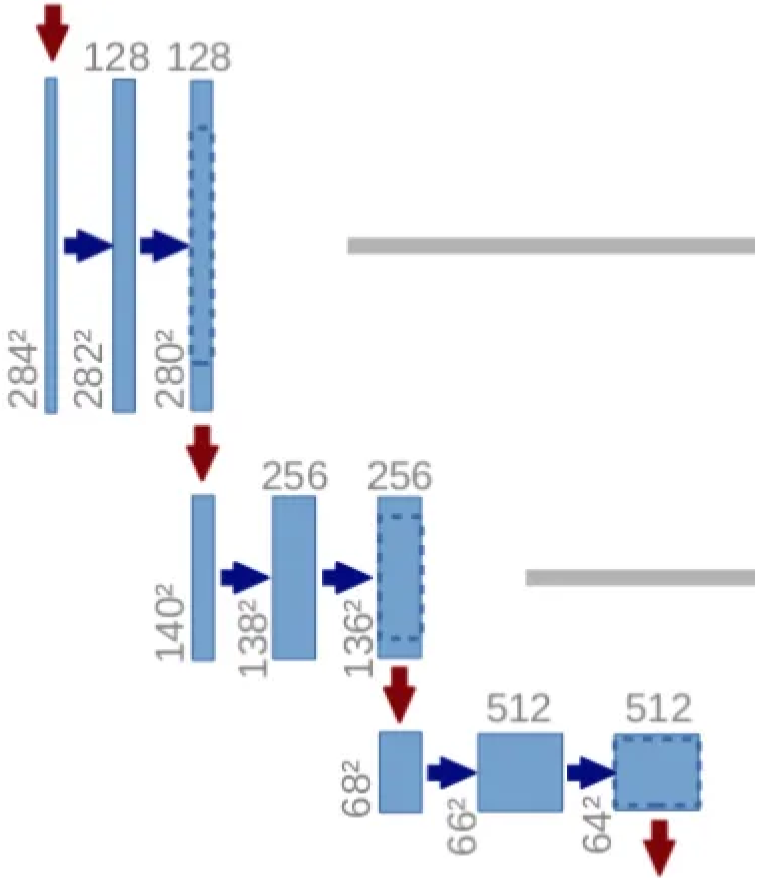
unet2.py hosted with ❤ by GitHub view raw

which matches to:



Notice that each process constitutes two convolutional layers, and the number of channel changes from 1 → 64, as convolution process will increase the depth of the image. The red arrow pointing down is the max pooling process which halves down size of image(the size reduced from 572x572 → 568x568 is due to padding issues, but the implementation here uses padding= “same”).

The process is repeated 3 more times:



with code:

```
conv2 = Conv2D(start_neurons * 2, (3, 3), activation="relu", padding="same")(pool1)
conv2 = Conv2D(start_neurons * 2, (3, 3), activation="relu", padding="same")(conv2)
pool2 = MaxPooling2D((2, 2))(conv2)
pool2 = Dropout(0.5)(pool2)

conv3 = Conv2D(start_neurons * 4, (3, 3), activation="relu", padding="same")(pool2)
conv3 = Conv2D(start_neurons * 4, (3, 3), activation="relu", padding="same")(conv3)
pool3 = MaxPooling2D((2, 2))(conv3)
pool3 = Dropout(0.5)(pool3)

conv4 = Conv2D(start_neurons * 8, (3, 3), activation="relu", padding="same")(pool3)
conv4 = Conv2D(start_neurons * 8, (3, 3), activation="relu", padding="same")(conv4)
pool4 = MaxPooling2D((2, 2))(conv4)
pool4 = Dropout(0.5)(pool4)
```

unet3.py hosted with ❤ by GitHub view raw

and now we reaches at the bottommost:

still 2 convolutional layers are built, but with no max pooling:

```
# Middle
convm = Conv2D(start_neurons * 16, (3, 3), activation="relu", padding="same")(pool4)
convm = Conv2D(start_neurons * 16, (3, 3), activation="relu", padding="same")(convm)
```

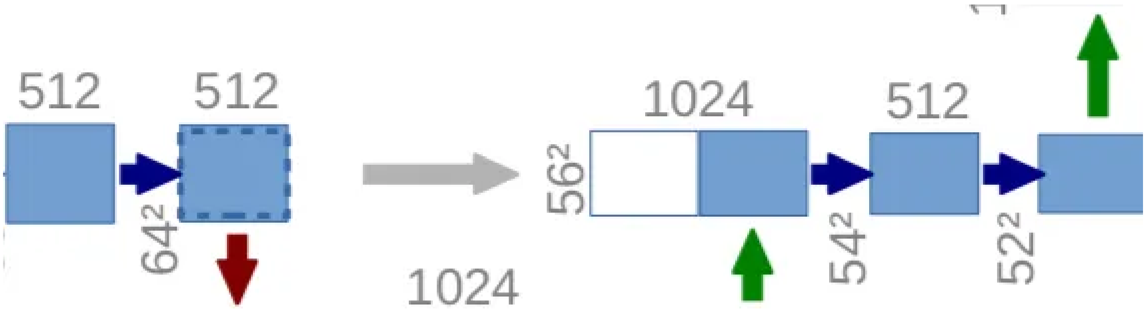
unet4.py hosted with ❤ by GitHub view raw

The image at this moment has been resized to 28x28x1024. Now let's get to the expansive path.

Expansive Path

In the expansive path, the image is going to be upsized to its original size. The formula follows:

```
conv_2d_transpose -> concatenate -> conv_layer1 -> conv_layer2
```




```
deconv4 = Conv2DTranspose(start_neurons * 8, (3, 3), strides=(2, 2), padding="same")(co
uconv4 = concatenate([deconv4, conv4])
uconv4 = Dropout(0.5)(uconv4)
uconv4 = Conv2D(start_neurons * 8, (3, 3), activation="relu", padding="same")(uconv4)
uconv4 = Conv2D(start_neurons * 8, (3, 3), activation="relu", padding="same")(uconv4)
```

unet5.py hosted with ❤ by GitHub view raw

Transposed convolution is an upsampling technic that expands the size of images. There is a visualised demo here and an explanation here. Basically, it does some padding on the original image followed by a convolution operation.

After the transposed convolution, the image is upsized from 28x28x1024 → 56x56x512, and then, this image is concatenated with the corresponding image from the contracting path and together makes an image of size 56x56x1024. The reason here is to combine the information from the previous layers in order to get a more precise prediction.

In line 4 and line 5, 2 other convolution layers are added.

Same as before, this process is repeated 3 more times:

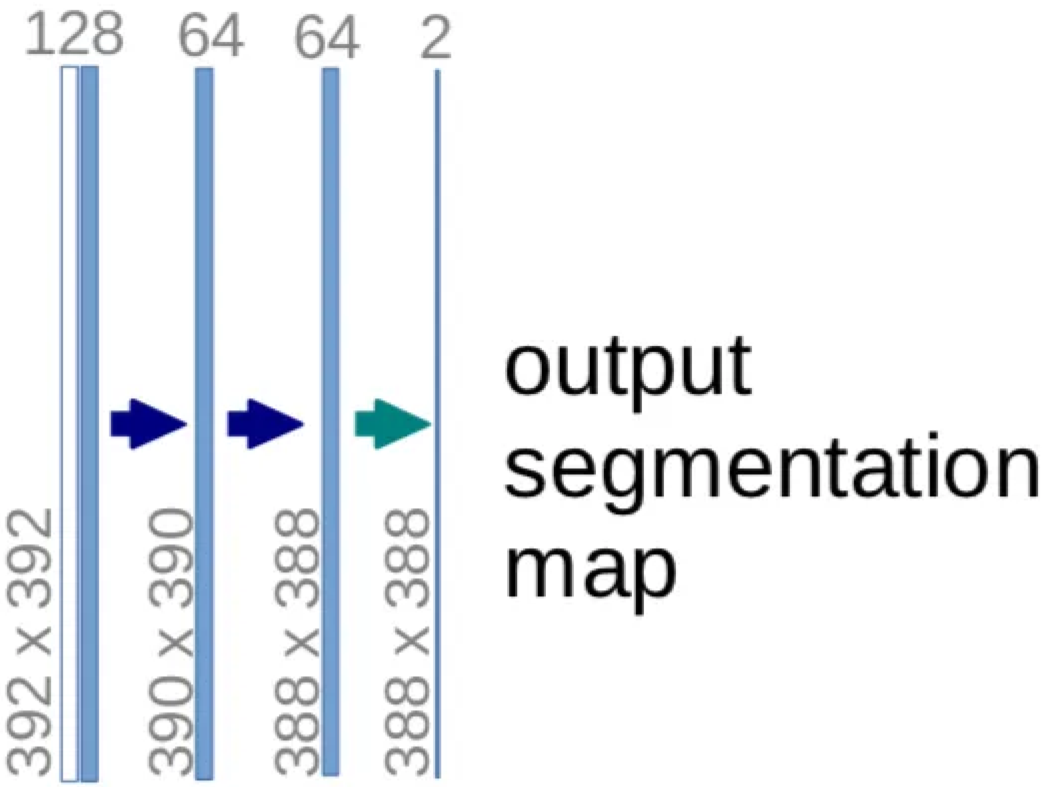
```
deconv3 = Conv2DTranspose(start_neurons * 4, (3, 3), strides=(2, 2), padding="same")(uc
uconv3 = concatenate([deconv3, conv3])
uconv3 = Dropout(0.5)(uconv3)
uconv3 = Conv2D(start_neurons * 4, (3, 3), activation="relu", padding="same")(uconv3)
uconv3 = Conv2D(start_neurons * 4, (3, 3), activation="relu", padding="same")(uconv3)

deconv2 = Conv2DTranspose(start_neurons * 2, (3, 3), strides=(2, 2), padding="same")(uc
uconv2 = concatenate([deconv2, conv2])
uconv2 = Dropout(0.5)(uconv2)
uconv2 = Conv2D(start_neurons * 2, (3, 3), activation="relu", padding="same")(uconv2)
uconv2 = Conv2D(start_neurons * 2, (3, 3), activation="relu", padding="same")(uconv2)

deconv1 = Conv2DTranspose(start_neurons * 1, (3, 3), strides=(2, 2), padding="same")(uc
uconv1 = concatenate([deconv1, conv1])
uconv1 = Dropout(0.5)(uconv1)
uconv1 = Conv2D(start_neurons * 1, (3, 3), activation="relu", padding="same")(uconv1)
uconv1 = Conv2D(start_neurons * 1, (3, 3), activation="relu", padding="same")(uconv1)
```

unet6.py hosted with ❤ by GitHub view raw

Now we've reached the uppermost of the architecture, the last step is to reshape the image to satisfy our prediction requirements.




```
1   output_layer = Conv2D(1, (1,1), padding="same", activation="sigmoid")(uconv1)
```

unet7.py hosted with ❤ by GitHub view raw

The last layer is a convolution layer with 1 filter of size 1x1(notice that there is no dense layer in the whole network). And the rest left is the same for neural network training.

Conclusion

UNet is able to do image localisation by predicting the image pixel by pixel and the author of UNet claims in his paper that the network is strong enough to do good prediction based on even few data sets by using excessive data augmentation techniques. There are many applications of image segmentation using UNet and it also occurs in lots of competitions. One should try out on yourself and I hope this post could be a good starting point for you.

Reference:

- https://github.com/hlamba28/UNET-TGS/blob/master/TGS%20UNET.ipynb
- https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47

- https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d
- https://medium.com/activating-robotic-minds/up-sampling-with-transposed-convolution-9ae4f2df52d0
- https://www.kaggle.com/phoenigs/u-net-dropout-augmentation-stratification

Machine Learning | Neural Networks | Data Science

Follow

Written by Jeremy Zhang

1.2K Followers · Writer for Towards Data Science

Hmm…I am a data scientist looking to catch up the tide…

More from Jeremy Zhang and Towards Data Science

Jeremy Zhang in Towards Data Science

Reinforcement Learning — Implement TicTacToe

We have implemented grid world game by iteratively updating Q value function, which is the estimating value of (state, action) pair. This...

6 min read · May 19, 2019

271 8

Open in app ↗

Search

R

Thu Vu in Towards Data Science

How to Learn AI on Your Own (a self-study guide)

If your hands touch a keyboard for work, Artificial Intelligence is going to change your job in the next few years.

· 12 min read · Jan 5

2.2K 24

Michael Berk in Towards Data Science

1.5 Years of Spark Knowledge in 8 Tips

My learnings from Databricks customer engagements

8 min read · Dec 24, 2023

1.4K 8

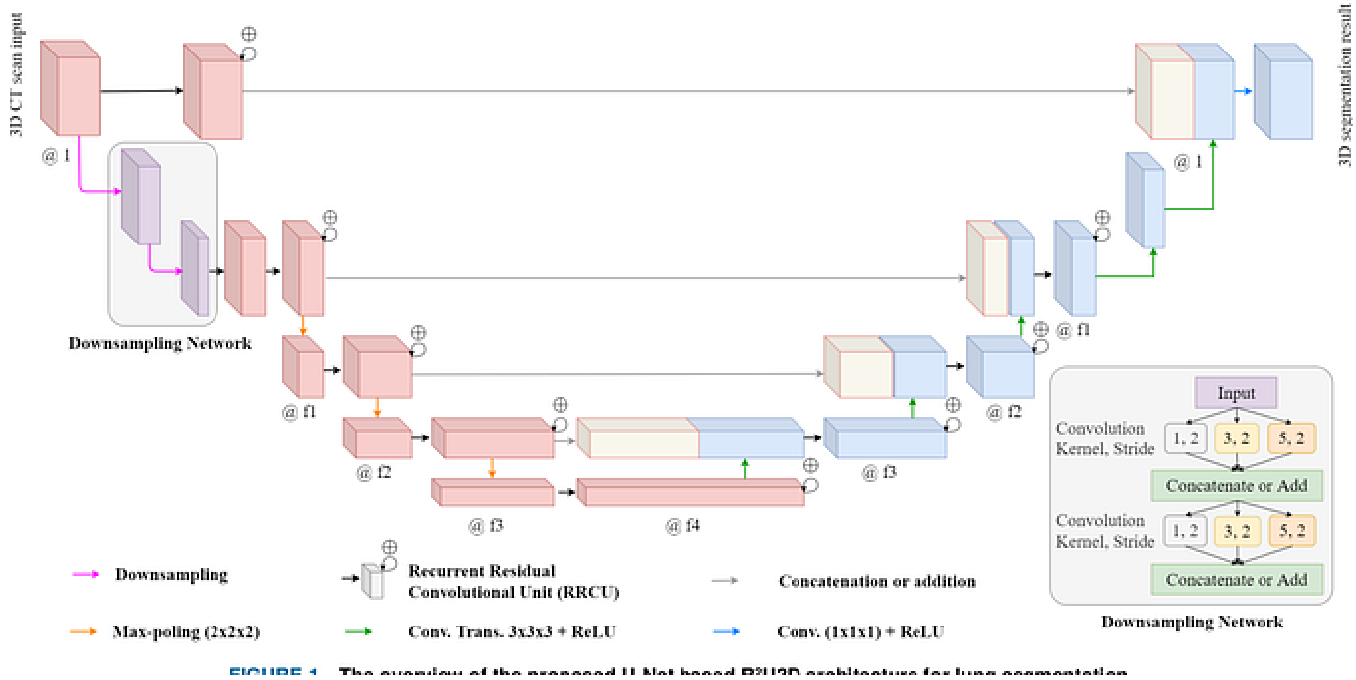
Jeremy Zhang in Towards Data Science

Importance Sampling Introduction

Estimate Expectations from a Different Distribution

4 min read · Aug 31, 2019

Recommended from Medium



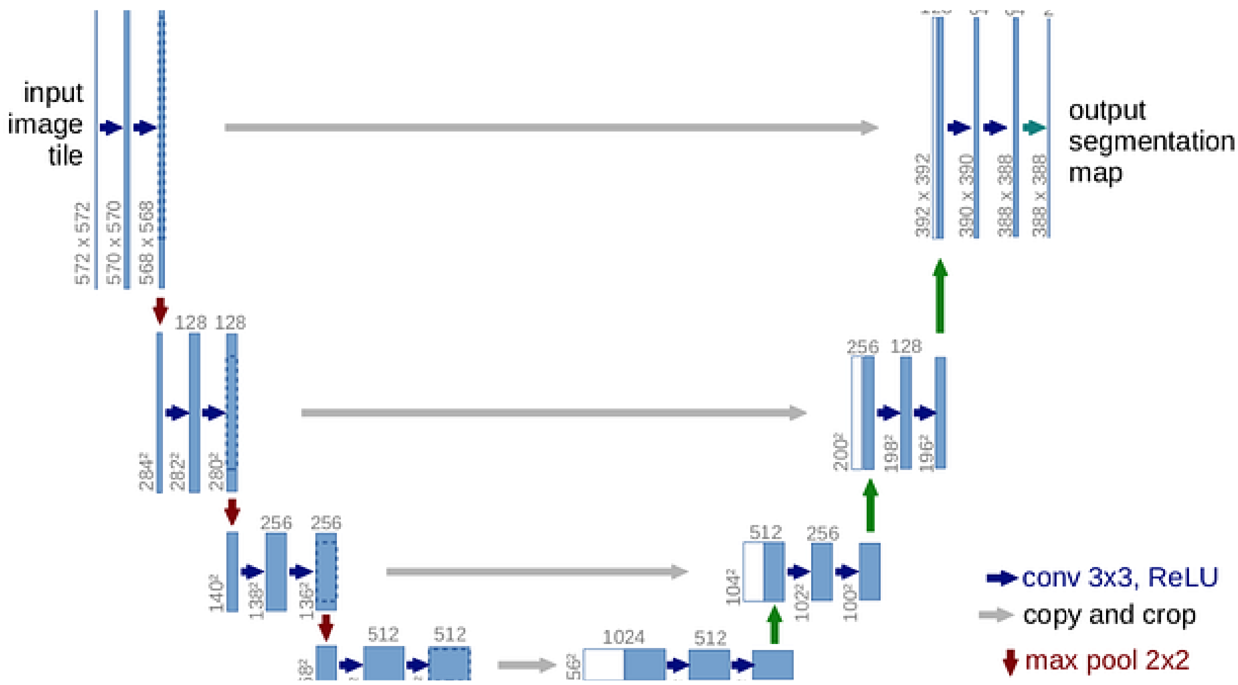
Alexquesada

U-Net: A Versatile Deep Learning Architecture for Image Segmentation

This article is one of the assigments of Data Glacier Intership 2023.

6 min read · Aug 4, 2023

18



Roushanak Rahmat, PhD in Code Like A Girl

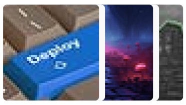
U-Net vs Residual U-Net vs Attention U-Net vs Attention Residual U-Net

UNet

· 6 min read · Aug 7, 2023

50

Lists



Predictive Modeling w/ Python

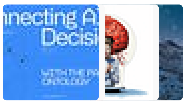
20 stories · 824 saves

Practical Guides to Machine Learning

10 stories · 953 saves

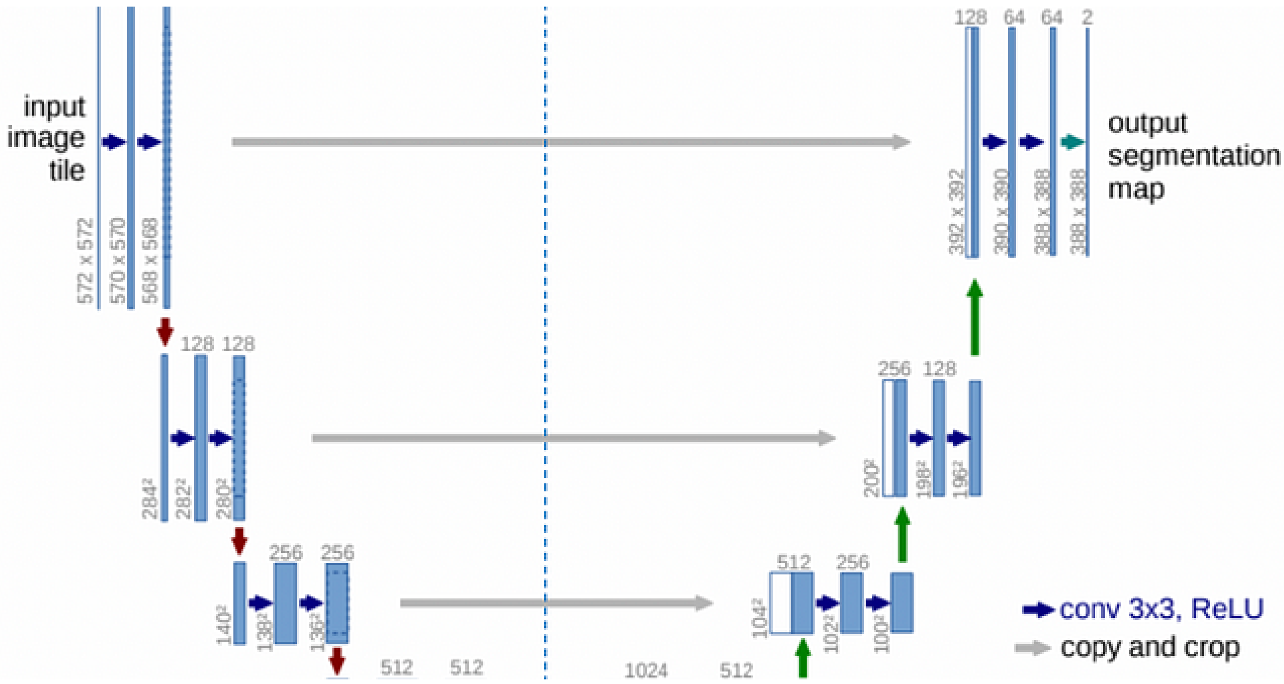
Natural Language Processing

1117 stories · 589 saves



data science and AI

39 stories · 49 saves



Shameerayaseen

U-Net: Advancing Image Segmentation with Convolutional Neural Networks

What is Image segmentation?

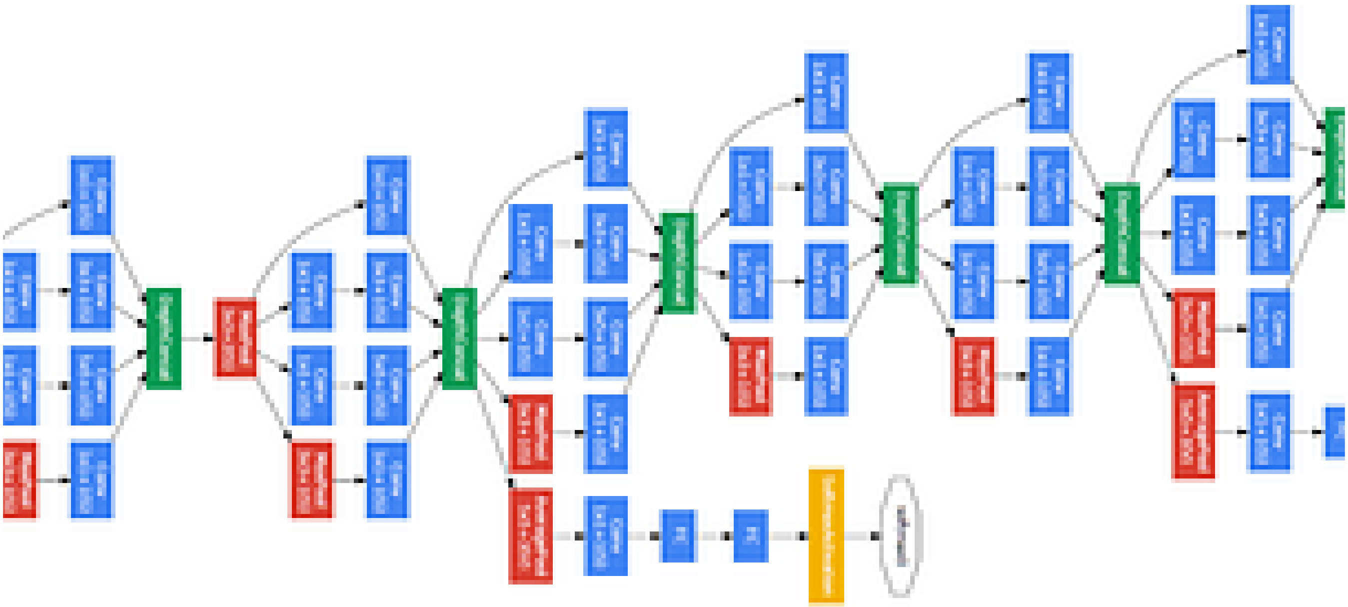
4 min read · Sep 4, 2023

Samruddhi Chitnis

Image Segmentation Transfer Learning Part 3

Table of Contents:

3 min read · Oct 11, 2023



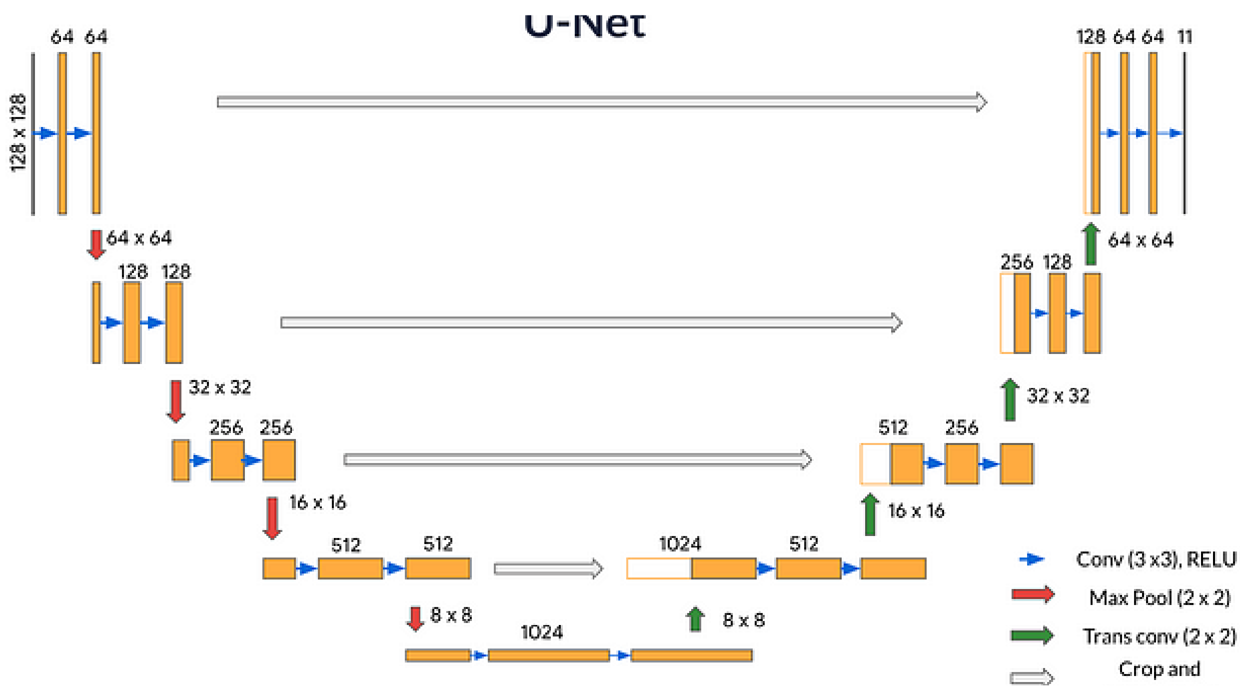
Everton Gomede, PhD

Exploring GoogLeNet: A Revolutionary Deep Learning Architecture

Introduction

7 min read · Oct 2, 2023

63



Lukman Aliyu

U-Net Architecture: Revolutionizing Computer Vision Through Innovative Image Segmentation

Introduction

5 min read · Aug 28, 2023

2

See more recommendations