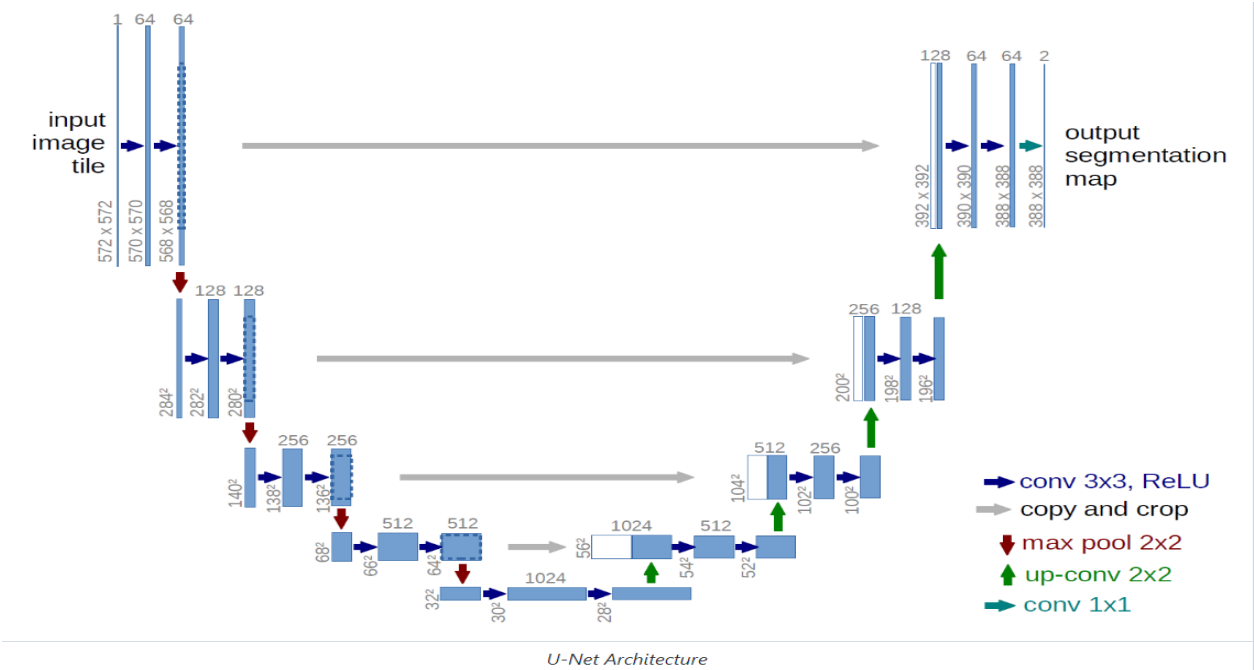
Lung Segmentation using U-NET

Aim: The main objective of this project is to perform lung segmentation using the chest x-ray images dataset on the famous UNET architecture

U-Net Network Architecture:

UNET is a U-shaped encoder-decoder network architecture, which consists of four encoder blocks and four decoder blocks that are connected via a bridge. The encoder network (contracting path) half the spatial dimensions and double the number of filters (feature channels) at each encoder block. Likewise, the decoder network doubles the spatial dimensions and half the number of feature channels.



U-Net Architecture

Convolution Block:

The entire UNET architecture consists of repeated use of two 3 x 3 convolution, each followed by a ReLU activation.

```
def conv_block(input, num_filters):
x = Conv2D(num_filters, 3, padding="same")(input)
x = BatchNormalization()(x) x = Activation("relu")(x)
 x = Conv2D(num_filters, 3, padding="same")(x)
x = BatchNormalization()(x)
x = Activation("relu")(x)
```

return x

1. **input:** The *input* represents the feature maps from the previous block.
2. **num_filters:** The *num_filters* refer to the number of output feature channels for the convolutional layers present in the *conv_block* function.

In the above UNET architecture, the two 3×3 convolutions are followed by a ReLU activation function. Here, we have introduced batch normalization in between the convolutional and the ReLU layer.

Encoder:

The encoder network acts as the feature extractor and learns an abstract representation of the input image through a sequence of the encoder blocks. Each encoder block consists of two 3×3 convolutions, where each convolution is followed by a ReLU (Rectified Linear Unit) activation function. The ReLU activation function introduces non-linearity into the network, which helps in the better generalization of the training data. The output of the ReLU acts as a skip connection for the corresponding decoder block.

Next, follows a 2×2 max-pooling, where the spatial dimensions (height and width) of the feature maps are reduced by half. This reduces the computational cost by decreasing the number of trainable parameters.

Code:

```
def encoder_block(input, num_filters):
  x = conv_block(input, num_filters)
  p = MaxPool2D((2, 2))(x)
  return x, p
```

Where,

1. **input:** The *input* represents the feature maps from the previous block.
2. **num_filters:** The n*um filters* refers to the number of output feature channels.

Skip Connections:

These skip connections provide additional information that helps the decoder to generate better semantic features. They also act as a shortcut connection that helps in the direct flow of gradients to the earlier layers without any degradation. In simple terms, we can say that skip connection helps in better flow of gradient while backpropagation, which in turn helps the network to learn better representation.

Bridge:

The bridge connects the encoder and the decoder network and completes the flow of information. It consists of two 3×3 convolutions, where each convolution is followed by a ReLU activation function.

Decoder Block:

The decoder network is used to take the abstract representation and generate a semantic segmentation mask. The decoder block starts with a 2×2 transpose convolution. Next, it is concatenated with the corresponding skip connection feature map from the encoder block. These skip connections provide features from earlier layers that are sometimes lost due to the depth of the network. After that, two 3×3 convolutions are used, where each convolution is followed by a ReLU activation function.
The output of the last decoder passes through a 1×1 convolution with sigmoid activation. The sigmoid activation function gives the segmentation mask representing the pixel-wise classification.

Code:

```
def decoder_block(input, skip_features, num_filters):

x = Conv2DTranspose(num_filters, (2, 2), strides=2, padding="same")(input)

x = Concatenate()([x, skip_features])

x = conv_block(x, num_filters)

return x
```

Where,

1. **input:** The *input* represents the feature maps from the previous block.
2. **skip_features:** The skip features represent the feature maps from the encoder block that are fetched through the skip connection.
3. **num_filters:** The num filters represents the number of output feature channels.

U-Net Archicture Code:

```
def build_unet(input_shape):
  inputs = Input(input_shape)

  s1, p1 = encoder_block(inputs, 64)

  s2, p2 = encoder_block(p1, 128)

  s3, p3 = encoder_block(p2, 256)

  s4, p4 = encoder_block(p3, 512)
```

```
b1 = conv_block(p4, 1024)

d1 = decoder_block(b1, s4, 512)
d2 = decoder_block(d1, s3, 256)
d3 = decoder_block(d2, s2, 128)
d4 = decoder_block(d3, s1, 64)

outputs = Conv2D(1, 1, padding="same", activation="sigmoid")(d4)
model = Model(inputs, outputs, name="U-Net")
return model
```