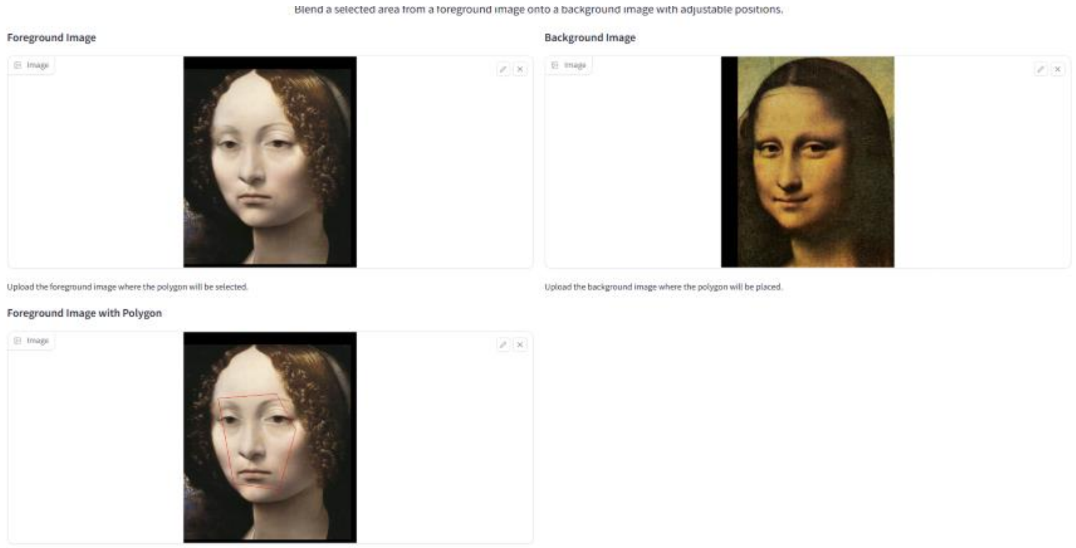
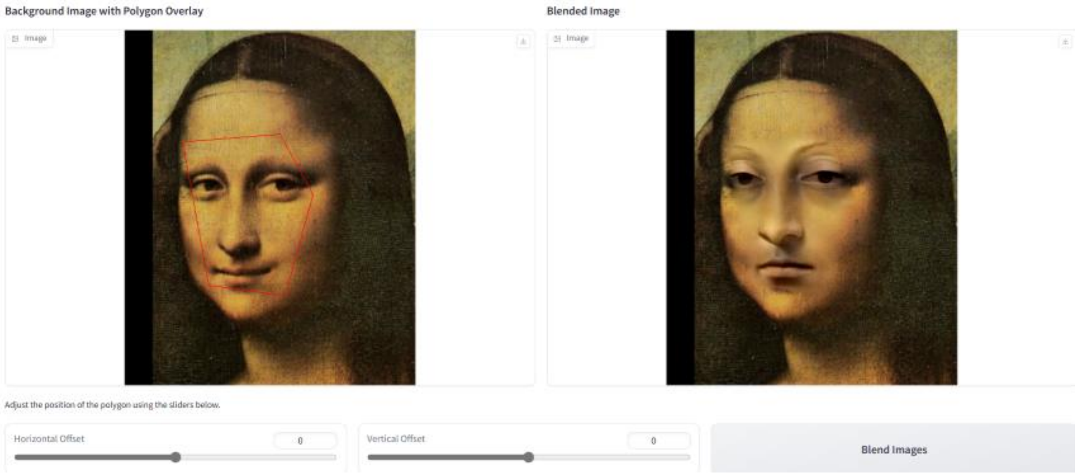
首先是 Poisson Image Blending 的实现。



Blend a selected area from a foreground image onto a background image with adjustable positions.

Upload the foreground image where the polygon will be selected.

Upload the background image where the polygon will be placed.



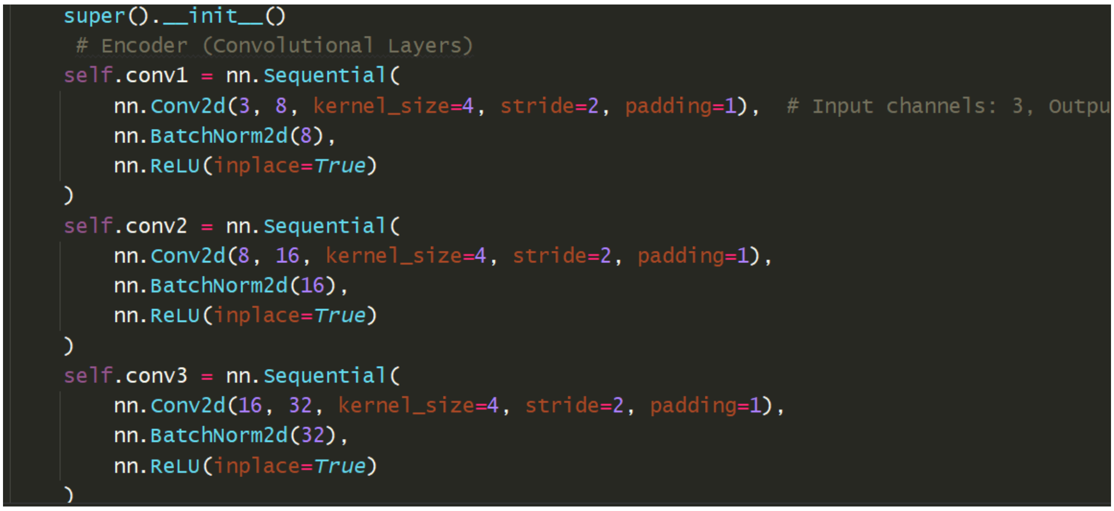
Adjust the position of the polygon using the sliders below.

Horizontal Offset 0

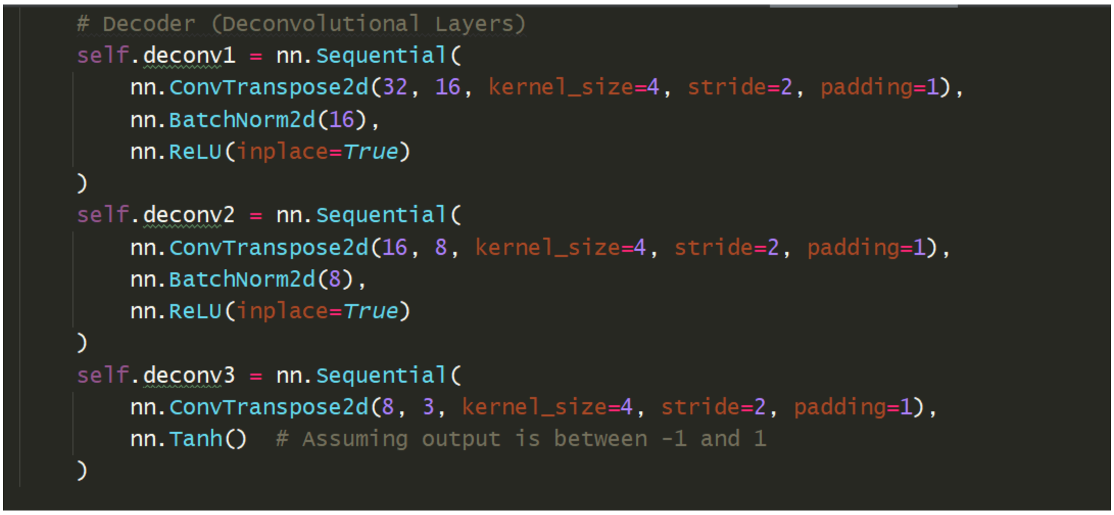
Vertical Offset 0

首先在 source image 框选多边形区域，而后进行融合，融合效果如上。

Implement [Pix2Pix](#) with [Fully Convolutional Layers](#)。



```
super().__init__()
 # Encoder (Convolutional Layers)
self.conv1 = nn.Sequential(
    nn.Conv2d(3, 8, kernel_size=4, stride=2, padding=1),  # Input channels: 3, Outpu
    nn.BatchNorm2d(8),
    nn.ReLU(inplace=True)
)
self.conv2 = nn.Sequential(
    nn.Conv2d(8, 16, kernel_size=4, stride=2, padding=1),
    nn.BatchNorm2d(16),
    nn.ReLU(inplace=True)
)
self.conv3 = nn.Sequential(
    nn.Conv2d(16, 32, kernel_size=4, stride=2, padding=1),
    nn.BatchNorm2d(32),
    nn.ReLU(inplace=True)
)
```



```
# Decoder (Deconvolutional Layers)
self.deconv1 = nn.Sequential(
    nn.ConvTranspose2d(32, 16, kernel_size=4, stride=2, padding=1),
    nn.BatchNorm2d(16),
    nn.ReLU(inplace=True)
)
self.deconv2 = nn.Sequential(
    nn.ConvTranspose2d(16, 8, kernel_size=4, stride=2, padding=1),
    nn.BatchNorm2d(8),
    nn.ReLU(inplace=True)
)
self.deconv3 = nn.Sequential(
    nn.ConvTranspose2d(8, 3, kernel_size=4, stride=2, padding=1),
    nn.Tanh()  # Assuming output is between -1 and 1
)
```

通过卷积和反卷积构建编码器和解码器，从而搭建一个全卷积网络。