A Simple U-Net in PyTorch

```python
import torch
import torch.nn as nn
import torch.nn.functional as F

class UNet(nn.Module):
    def __init__(self, in_channels=3, out_channels=1):
        super().__init__()

        # Down path
        self.conv1 = nn.Conv2d(in_channels, 64, kernel_size=3, padding=1)
        self.conv2 = nn.Conv2d(64, 128, kernel_size=3, padding=1)
        self.conv3 = nn.Conv2d(128, 256, kernel_size=3, padding=1)

        self.bottleneck = nn.Conv2d(256, 512, kernel_size=3, padding=1)

        # Up path
        self.up3 = nn.ConvTranspose2d(512, 256, kernel_size=2, stride=2)
        self.conv3u = nn.Conv2d(512, 256, kernel_size=3, padding=1)

        self.up2 = nn.ConvTranspose2d(256, 128, kernel_size=2, stride=2)
        self.conv2u = nn.Conv2d(256, 128, kernel_size=3, padding=1)

        self.up1 = nn.ConvTranspose2d(128, 64, kernel_size=2, stride=2)
        self.conv1u = nn.Conv2d(128, 64, kernel_size=3, padding=1)

        self.out_conv = nn.Conv2d(64, out_channels, kernel_size=1)
        self.pool = nn.MaxPool2d(2)

    def forward(self, x):
        # Down
        d1 = F.relu(self.conv1(x))
        d2 = F.relu(self.conv2(self.pool(d1)))
        d3 = F.relu(self.conv3(self.pool(d2)))

        b = F.relu(self.bottleneck(self.pool(d3)))

        # Up (with skip connections)
        u3 = F.relu(self.conv3u(torch.cat([self.up3(b),  d3], dim=1)))
        u2 = F.relu(self.conv2u(torch.cat([self.up2(u3), d2], dim=1)))
        u1 = F.relu(self.conv1u(torch.cat([self.up1(u2), d1], dim=1)))

        return self.out_conv(u1)
```

Exercise: Draw the U-Net Architecture

Draw a block diagram of the U-Net defined on the previous page.

Include:

- Tensor shapes at each stage (assume 256×256 input)
- The skip connections

Questions

1. What is the spatial resolution at the bottleneck?
2. Why does `conv3u` take 512 input channels instead of 256?
3. What does `torch.cat([...], dim=1)` do? Why `dim=1`?
4. How would you modify this model for 128×128 inputs?