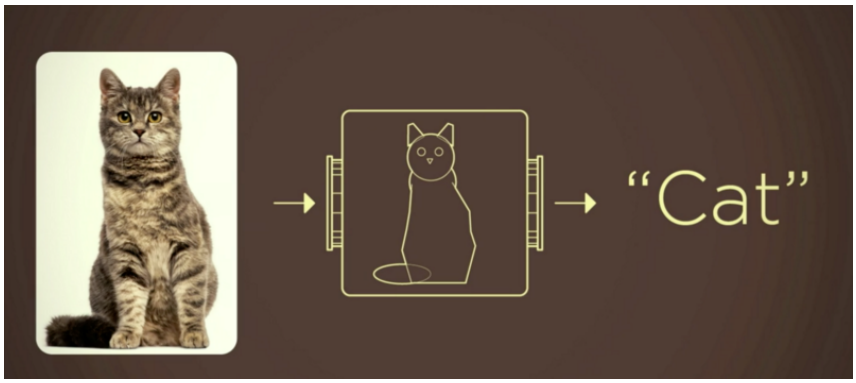
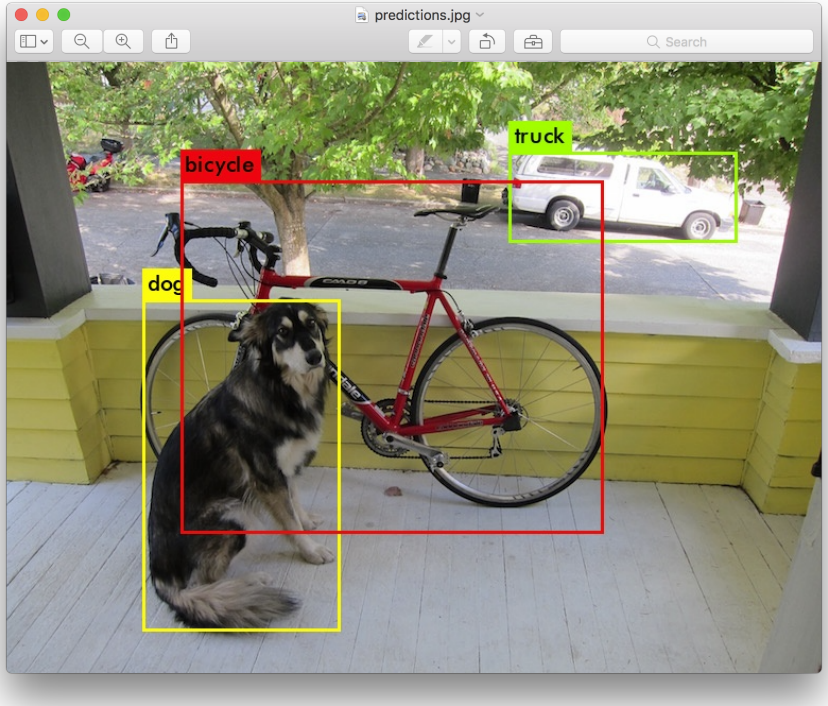
Unet语义分割

1️⃣知识前导

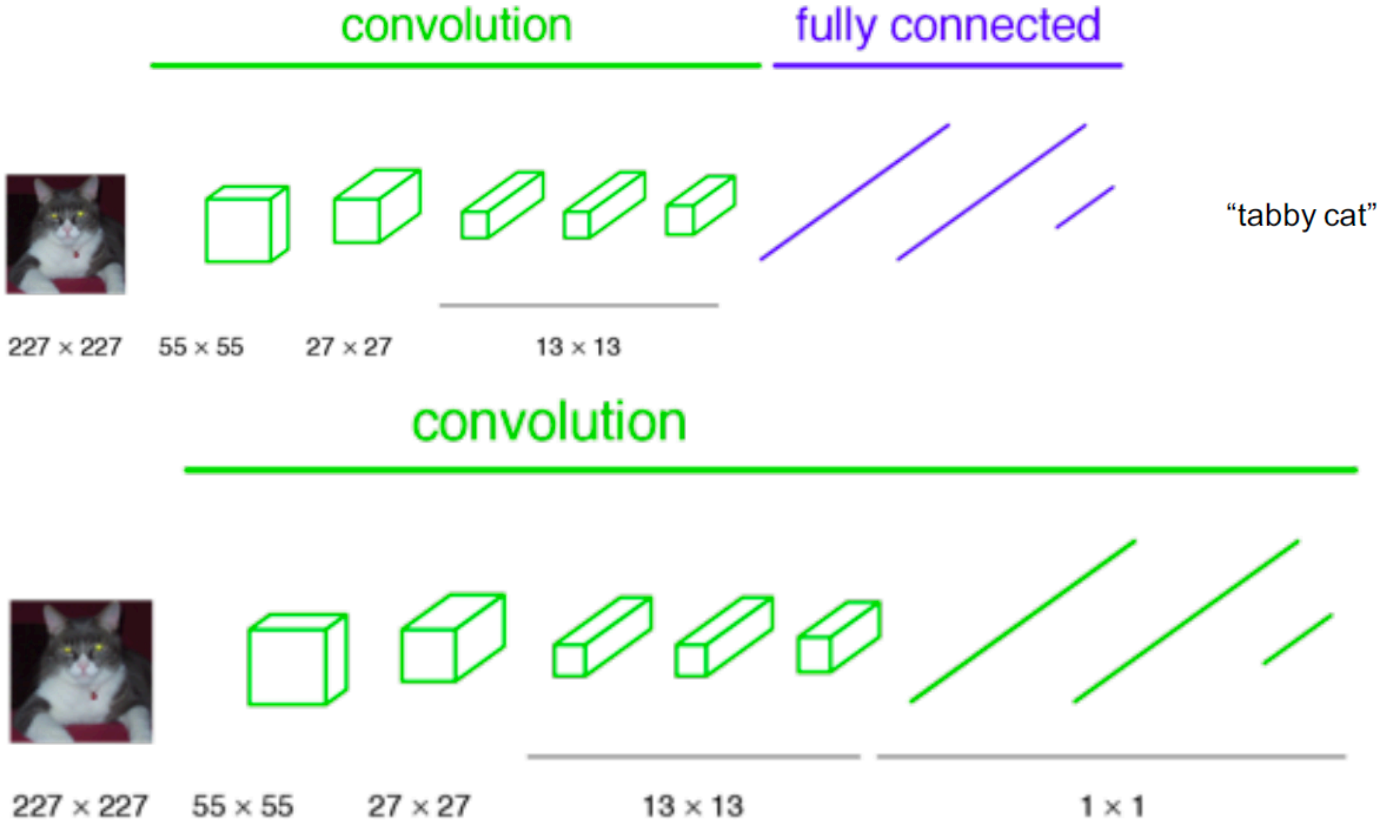
1.1图像分类、目标检测和语义分割



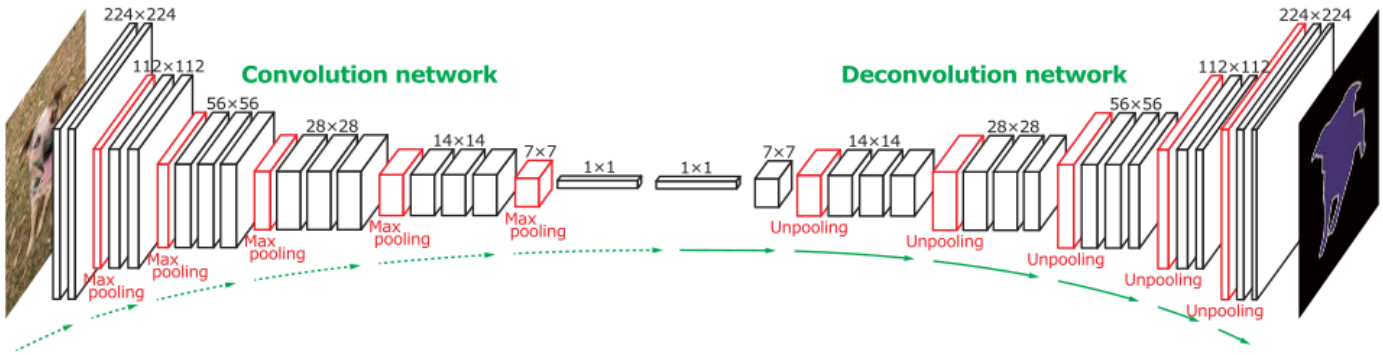


1.2 FCN模型

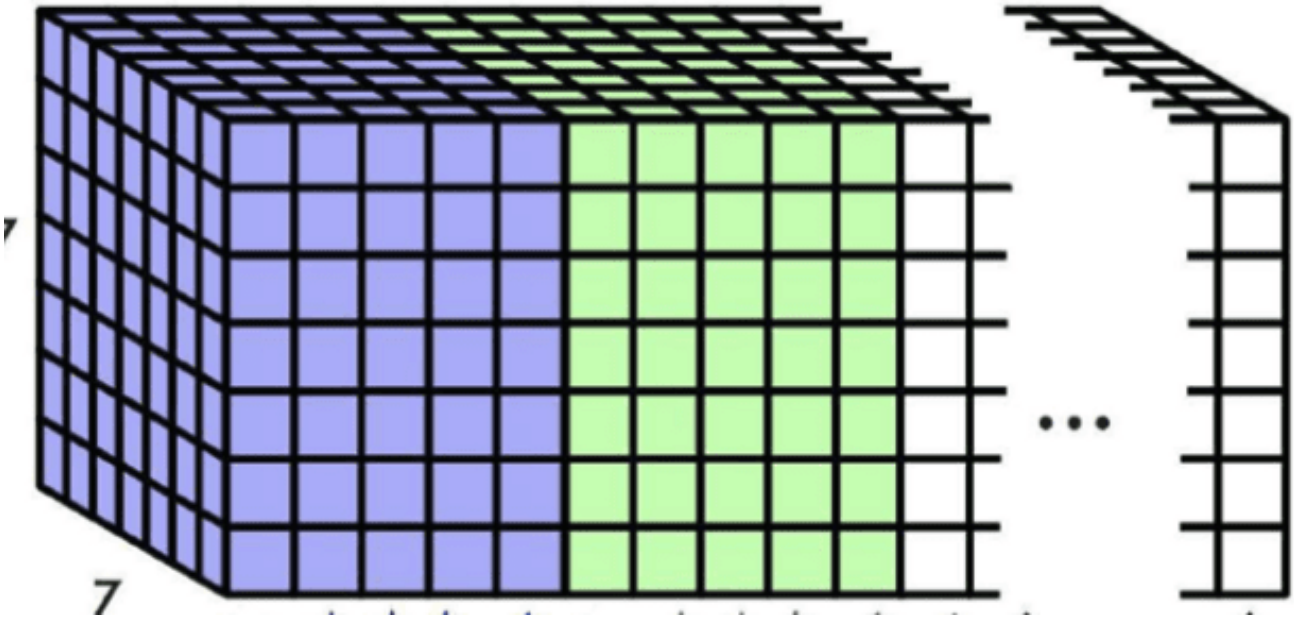
FCN的核心思想是使用**全卷积网络**（Fully Convolutional Network），并替换了传统CNN中的全连接层（FC层），使得网络能够处理任意尺寸的输入图像，并能够生成与输入图像尺寸相同的输出图像，其中每个像素都有对应的分类结果。



转置卷积：上采样（将特征图尺寸放大）



softmax的作用



```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torchvision.models as models

class FCN32(nn.Module):
    def __init__(self, num_classes):
        super(FCN32, self).__init__()

        # 使用预训练的 VGG16 作为特征提取部分
        vgg16 = models.vgg16(pretrained=True)

        # 使用 VGG16 的前面部分，直到最后一个池化层
        self.features = vgg16.features  # VGG16的卷积部分

        # FCN-32 使用反卷积将最后的特征图恢复到输入图像大小
        # 这里是一个简单的反卷积，利用一个大的卷积核进行上采样
        self.fc1 = nn.Conv2d(512, num_classes, kernel_size=1)  # 转换为类数通道
        self.deconv = nn.ConvTranspose2d(num_classes, num_classes, kernel_size=64,
stride=32, padding=16)

    def forward(self, x):
```

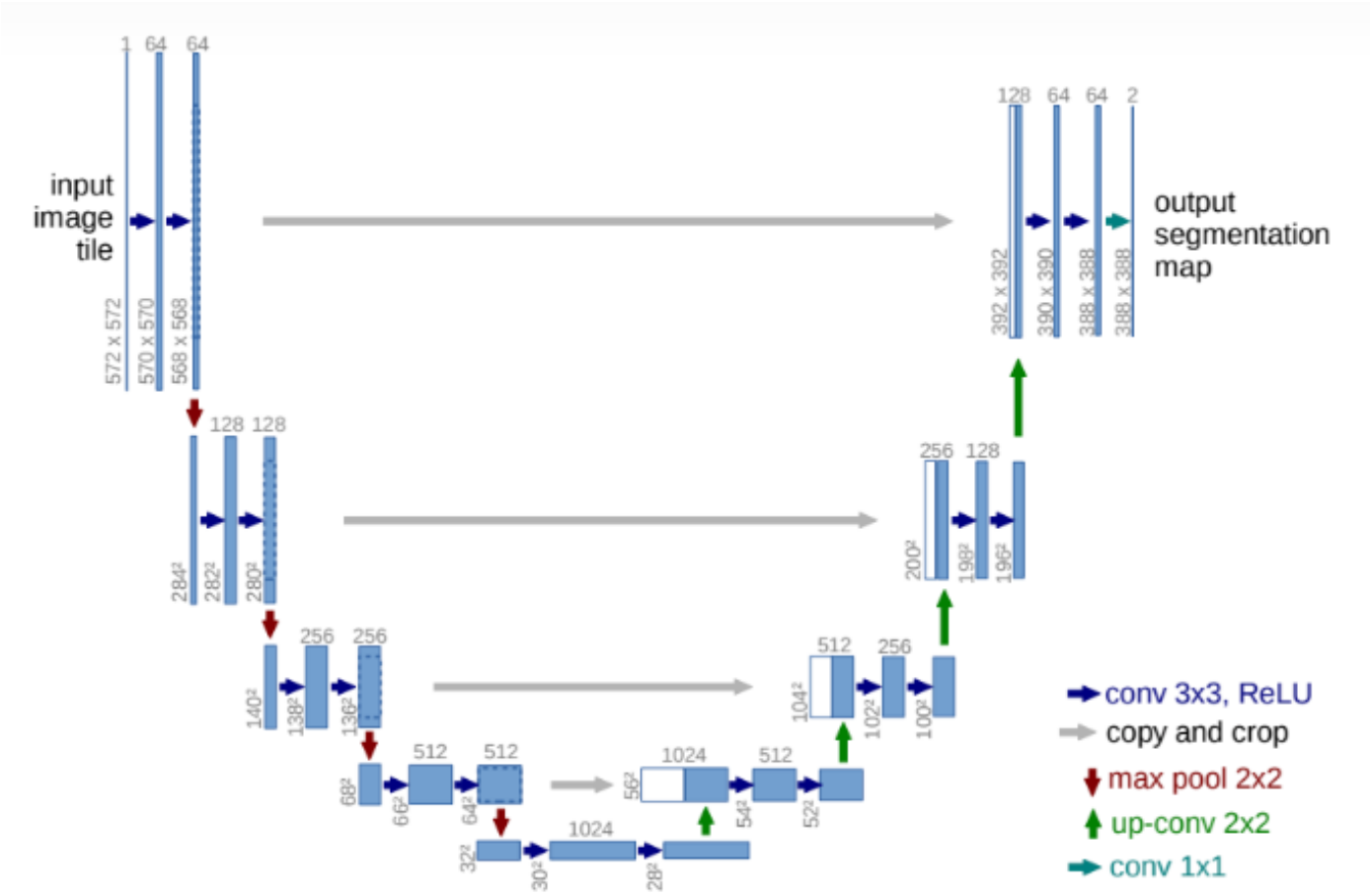
```
        # 1. 使用 VGG16 特征提取部分
        x = self.features(x)

        # 2. 将特征图通过1x1卷积层进行分类
        x = self.fc1(x)

        # 3. 使用转置卷积（反卷积）将特征图恢复到原图尺寸
        x = self.deconv(x)
        # x = F.softmax(x, dim=1)
        # 返回预测结果
        return x

# 创建模型并输出网络结构
model = FCN32(num_classes=21)  # 21类是常见的VOC数据集中的类别数
print(model)
```

2Unet



2.1U-Net的主要特点

1. **编码器（Encoder）**：
    - U-Net 的编码器部分类似于传统的卷积神经网络（CNN），由一系列的卷积层和池化层组成，目的是逐步提取图像的特征，同时逐步减少图像的空间分辨率。
2. **解码器（Decoder）**：

- 解码器部分则通过反卷积（转置卷积）或者上采样操作恢复图像的空间分辨率，使得最后的输出图像与输入图像具有相同的大小。
- 解码器使用了反卷积层或 **上采样** 来逐步还原空间维度，确保最终分割图像和输入图像具有相同尺寸。

3. **跳跃连接（Skip Connections）**：
   - 在 U-Net 中，跳跃连接是一个重要的设计。它将编码器中的低层特征图与解码器中的高层特征图进行结合。这样可以帮助网络恢复图像细节，使得网络能更好地进行像素级的预测。
   - 在解码器的每一层中，跳跃连接会将对应编码器部分的特征图拼接到当前的解码器特征图中，从而保留细节信息。