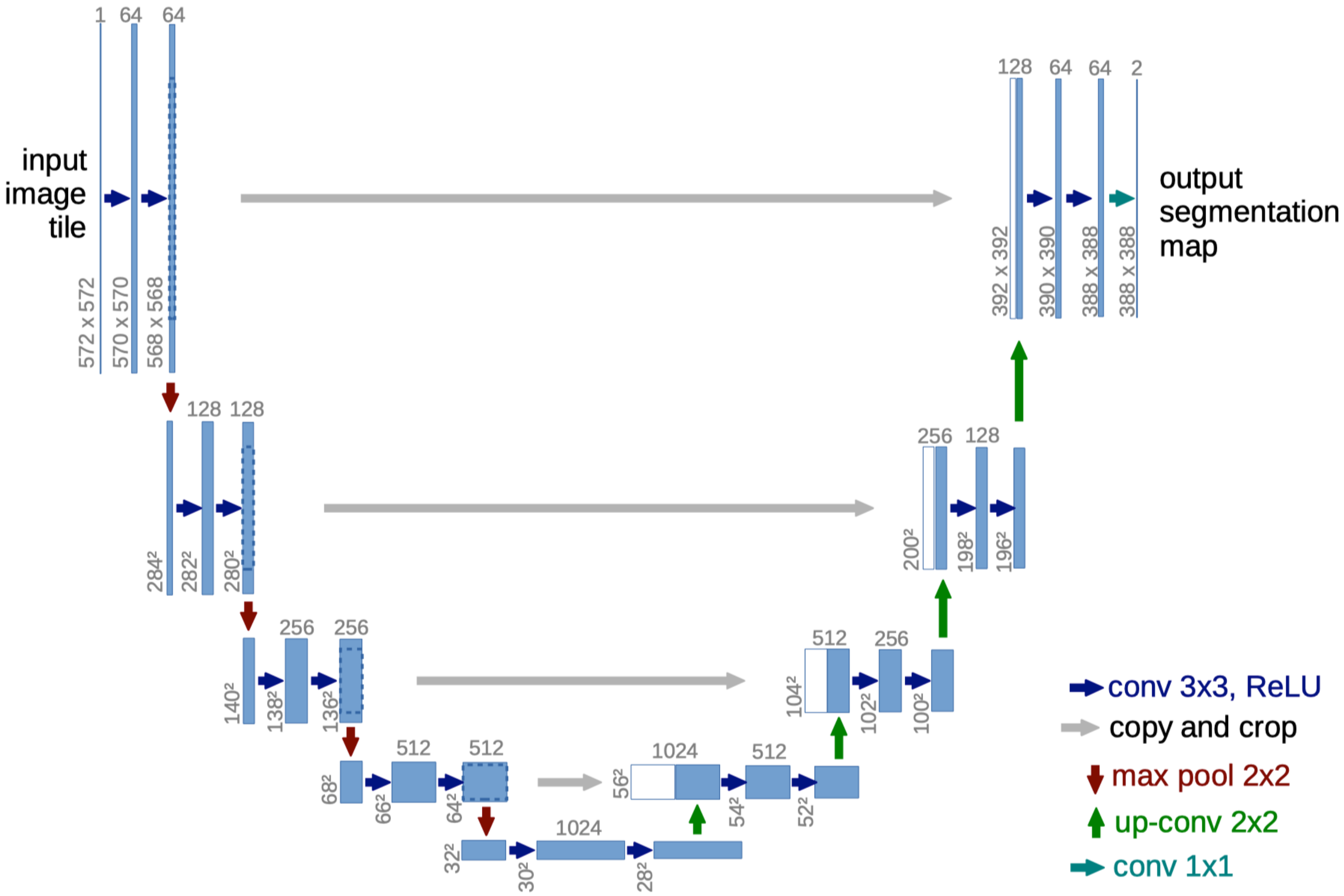
Refactor U-Net Code



Above is the network structure of U-Net, which mainly consists of the *downsampling layers* and subsequently *upsampling layers*. The following code written in Python by Facebook AI research (FAIR) can be used to build the *downsampling* and *upsampling layers* of U-Net. However, there are at least two problems with this implementation:

1. `ConvBlock` and `nn.ModuleList` can be seen as ***infrastructure code***, while the code used to calculate the input and output channels of each `ConvBlock` can be seen as ***domain code***. Clearly, the domain code and the infrastructure code (see here for more details of *domain code* and *infrastructure code*) are not well separated.
2. Code duplication. Clearly, the code lines from Line 43 to Line 49 and the code lines from Line 53 to Line 59 are heavily duplicated.

Please refactor the following U-Net code to resolve the above two problems.

```
1 """
2 Copyright (c) Facebook, Inc. and its affiliates.
3
```

```python
This source code is licensed under the MIT license found in the
LICENSE file in the root directory of this source tree.
"""

import torch
from torch import nn

from utils import ConvBlock


class UnetModel(nn.Module):
    """
    PyTorch implementation of a U-Net model.

    This is based on:
        Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional
        for biomedical image segmentation. In International Conference on Medica
        computing and computer-assisted intervention, pages 234–241. Springer, 2
    """

    def __init__(self, in_chans, out_chans, chans, num_pool_layers, drop_prob, p
        """
        Args:
            in_chans (int): Number of channels in the input to the U-Net model.
            out_chans (int): Number of channels in the output to the U-Net model
            chans (int): Number of output channels of the first convolution laye
            num_pool_layers (int): Number of down-sampling and up-sampling layer
            drop_prob (float): Dropout probability.
        """
        super().__init__()

        self.in_chans = in_chans
        self.out_chans = out_chans
        self.chans = chans
        self.num_pool_layers = num_pool_layers
        self.drop_prob = drop_prob

        outchans = chans
        self.down_sample_layers = nn.ModuleList([ConvBlock(in_chans, outchans, d
        ch = chans
        for i in range(num_pool_layers - 1):
            outchans = ch * 2
            self.down_sample_layers += [ConvBlock(ch, outchans, drop_prob)]
            ch *= 2

        self.up_sample_layers = nn.ModuleList()
        for i in range(num_pool_layers - 1):
            up_in = ch * 2
            self.up_sample_layers += [ConvBlock(up_in, ch // 2, drop_prob)]
            ch //= 2
        up_in = ch * 2
        self.up_sample_layers += [ConvBlock(up_in, ch, drop_prob)]
```