AI4EO

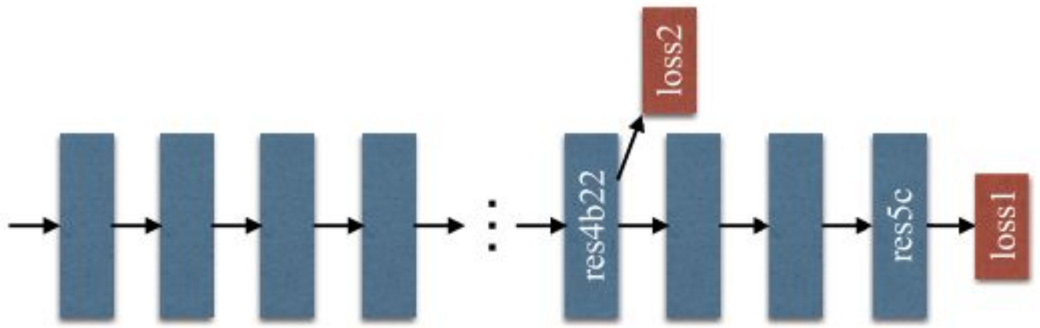
Special Topics in Applications (AIL861)
Artificial Intelligence for Earth Observation
Lecture 12

Instructor: Sudipan Saha

AI4EO

Deep Supervision

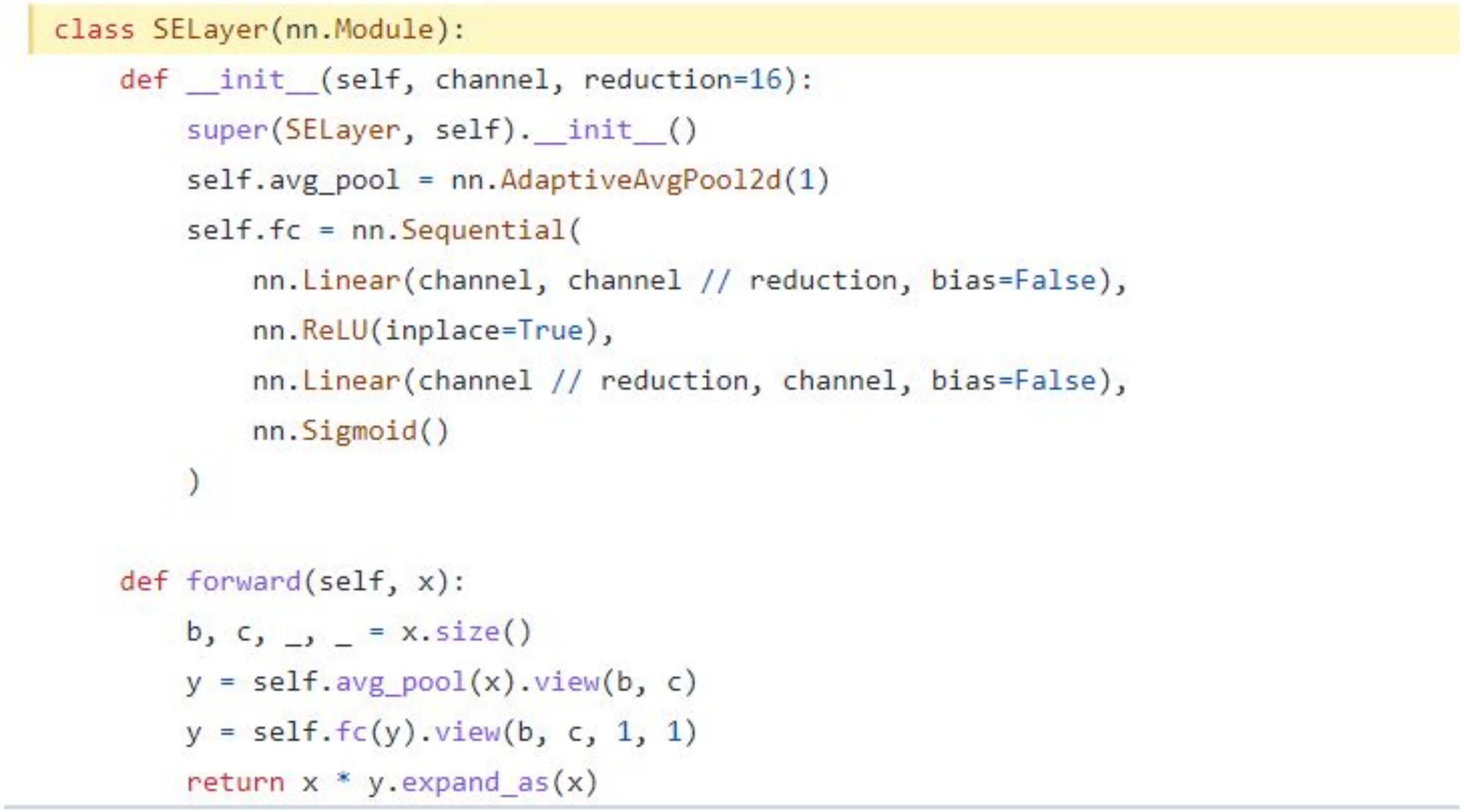
“Deep pretrained networks lead to good performance [17, 33, 13]. However, increasing depth of the network may introduce additional optimization difficulty as shown in [32, 19] for image classification. ResNet solves this problem with skip connection in each block. Latter layers of deep ResNet mainly learn residues based on previous ones. We contrarily propose generating initial results by supervision with an additional loss, and learning the residue afterwards with the final loss. Thus, optimization of the deep network is decomposed into two, each is simpler to solve.
An example of our deeply supervised ResNet101 [13] model is illustrated in Fig. 4. Apart from the main branch using softmax loss to train the final classifier, another classifier is applied after the fourth stage, i.e., the res4b22 residue block.”



Source: Pyramid Scene Parsing Network, 2017 https://github.com/kazuto1011/pspnet-pytorch/blob/master/libs/models/pspnet.py

See line 118, 122, 95-111

Channel Attention

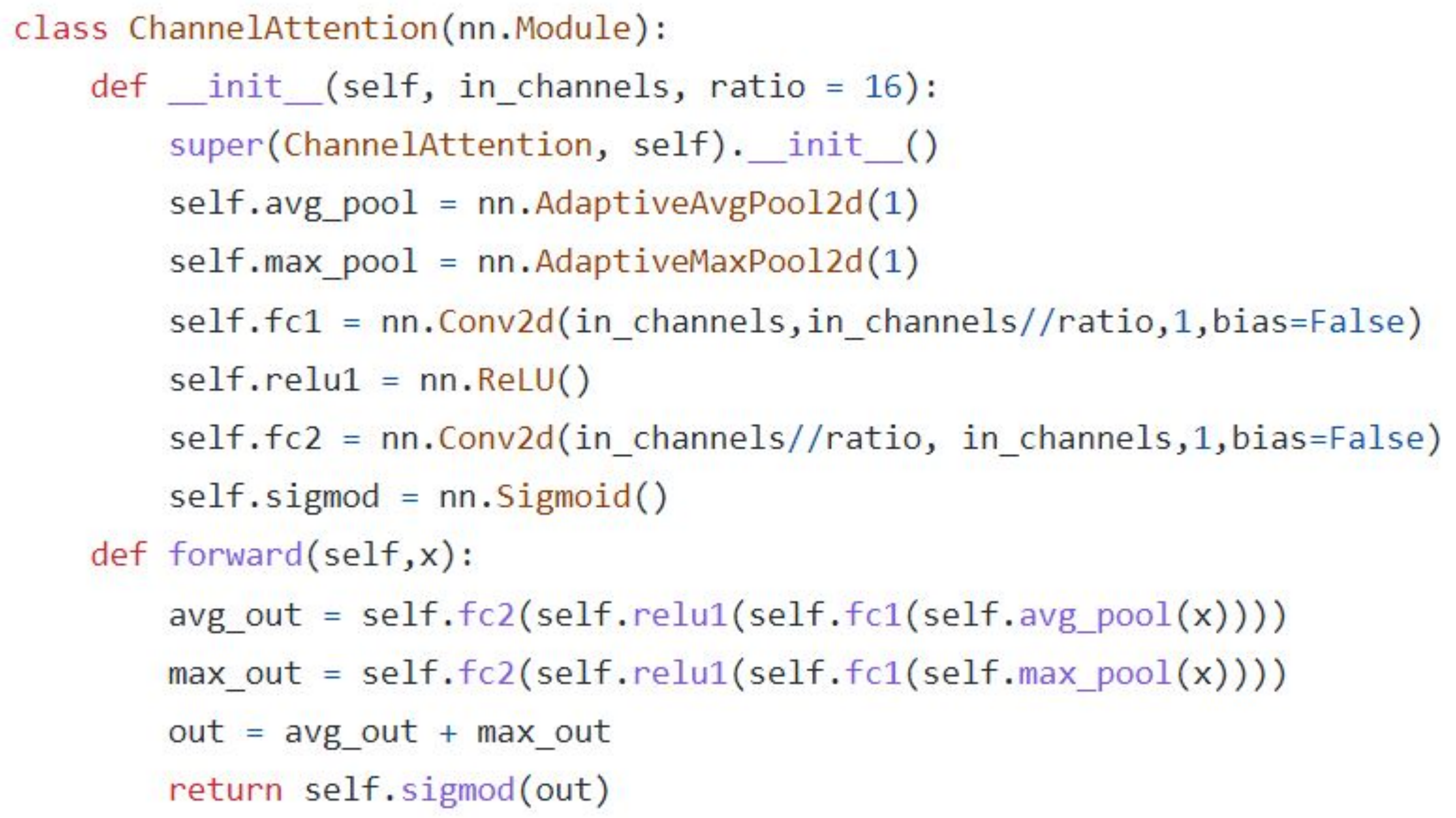


```python
class SELayer(nn.Module):
    def __init__(self, channel, reduction=16):
        super(SELayer, self).__init__()
        self.avg_pool = nn.AdaptiveAvgPool2d(1)
        self.fc = nn.Sequential(
            nn.Linear(channel, channel // reduction, bias=False),
            nn.ReLU(inplace=True),
            nn.Linear(channel // reduction, channel, bias=False),
            nn.Sigmoid()
        )

    def forward(self, x):
        b, c, _, _ = x.size()
        y = self.avg_pool(x).view(b, c)
        y = self.fc(y).view(b, c, 1, 1)
        return x * y.expand_as(x)
```

https://github.com/moskomule/senet.pytorch/blob/master/senet/se_module.py#L4

Channel Attention (Alternate Implementation)



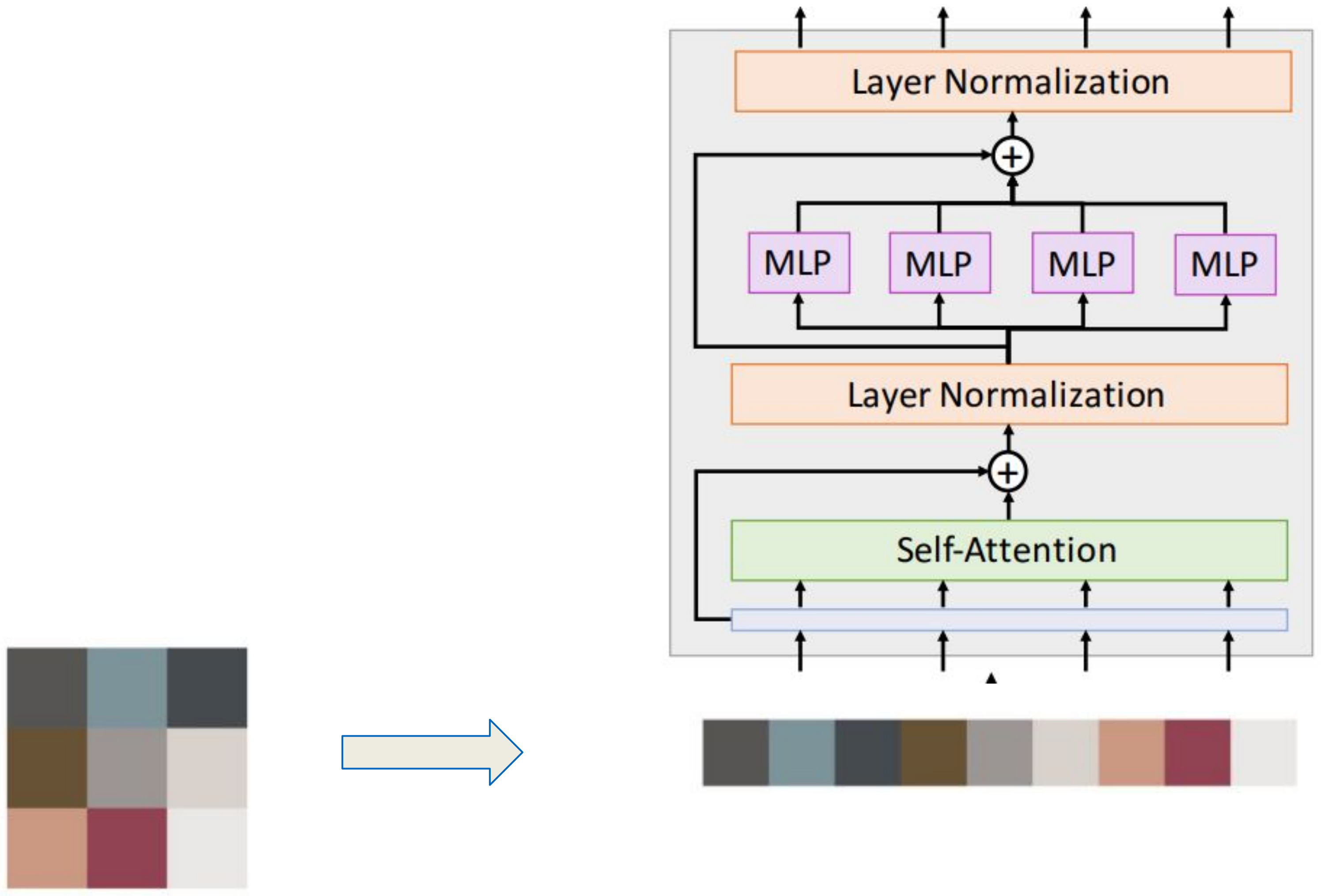
```
class ChannelAttention(nn.Module):
    def __init__(self, in_channels, ratio = 16):
        super(ChannelAttention, self).__init__()
        self.avg_pool = nn.AdaptiveAvgPool2d(1)
        self.max_pool = nn.AdaptiveMaxPool2d(1)
        self.fc1 = nn.Conv2d(in_channels,in_channels//ratio,1,bias=False)
        self.relu1 = nn.ReLU()
        self.fc2 = nn.Conv2d(in_channels//ratio, in_channels,1,bias=False)
        self.sigmod = nn.Sigmoid()
    def forward(self,x):
        avg_out = self.fc2(self.relu1(self.fc1(self.avg_pool(x))))
        max_out = self.fc2(self.relu1(self.fc1(self.max_pool(x))))
        out = avg_out + max_out
        return self.sigmod(out)
```

https://github.com/likyoo/Siam-NestedUNet/blob/master/models/Models.py

Transformer: Global Context

- Initially introduced in NLP
- Sentences are processed as a whole, instead of word by word
- Since sentence is processed as a whole, it does not “forget”
- Parallelism

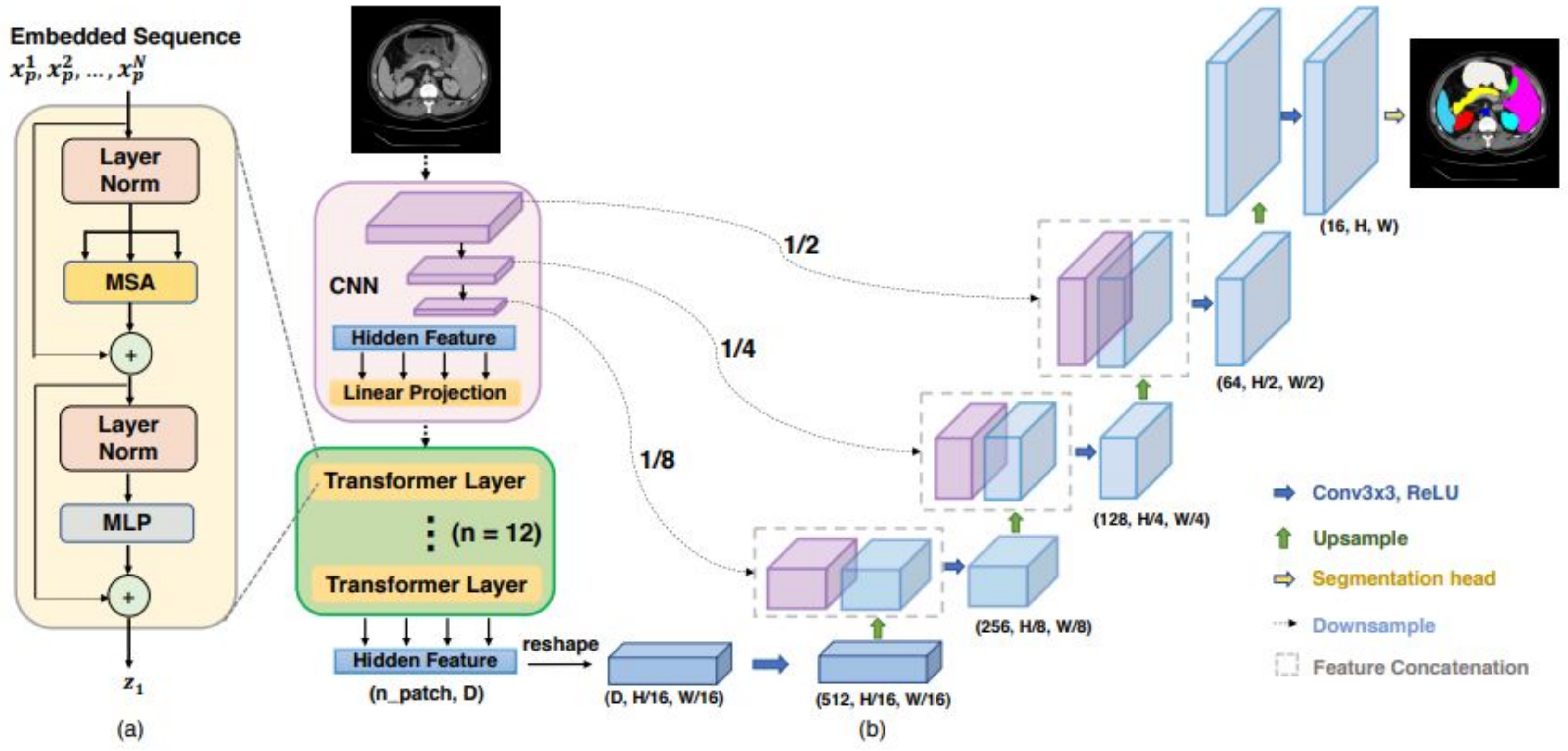
Vision Transformer



AI4EO

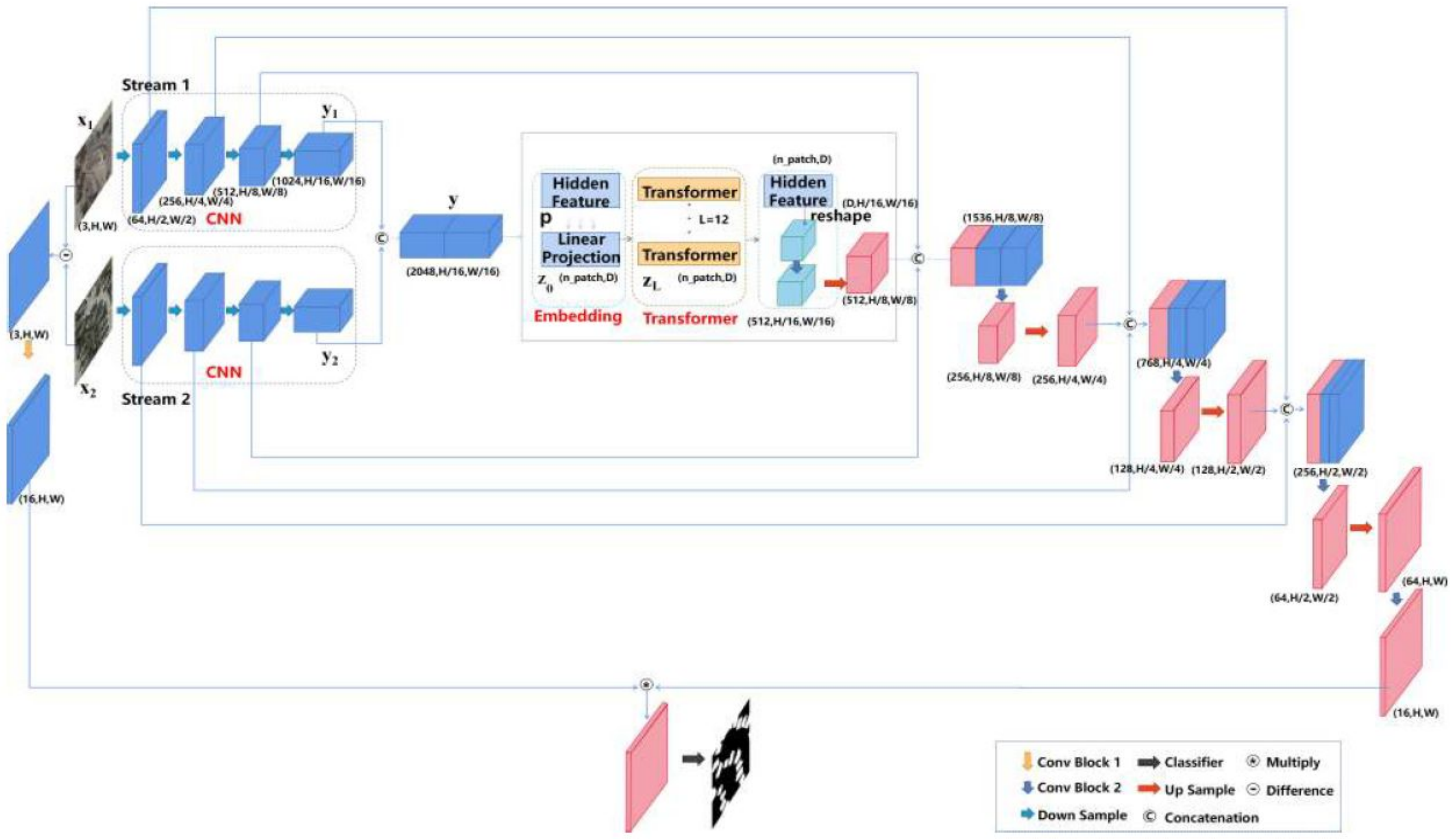
Transformer-Based Segmentation or Change Detection

TransUNet



AI4EO

TransUNet for CD



TransUNetCD: A Hybrid Transformer Network for Change Detection in Optical Remote-Sensing Images