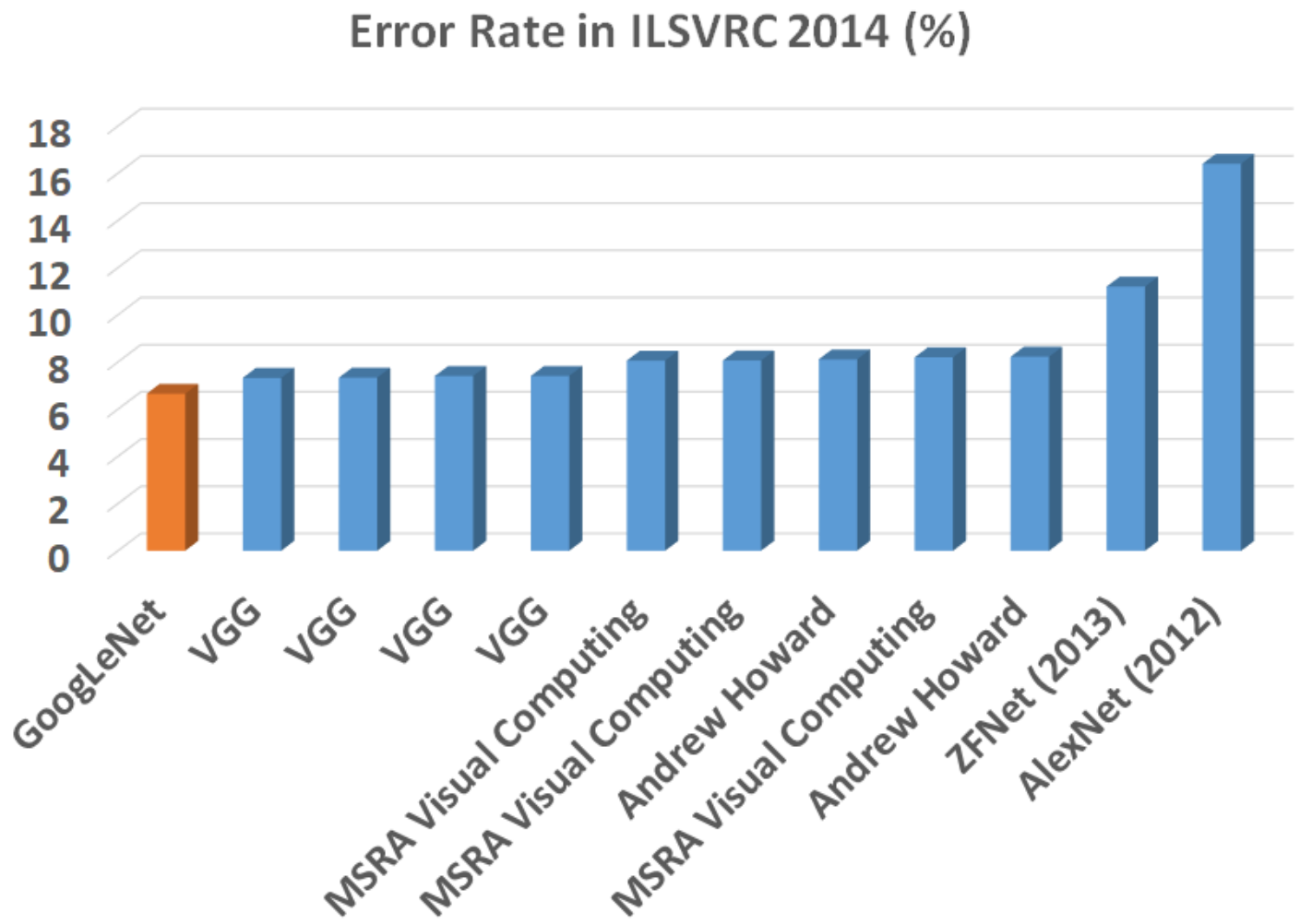
GoogLeNet

Kyeong Hwan Moon

01 GoogleNet

Introduction



01 Dataset

Inspection


n02097047 (196)


n01682714 (40)


n03134739 (522)


n04254777 (806)


n02859443 (449)


n02096177 (192)


n02107683 (239)

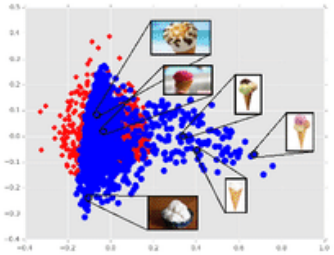

n01443537 (1)


n02264363 (318)


Icecream - ILSVRC12 (IN)


Icecream - WIN

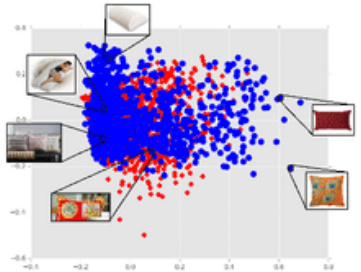

Icecream - WINC


ILSVR red, WINC blue


Pillow - ILSVRC12 (IN)

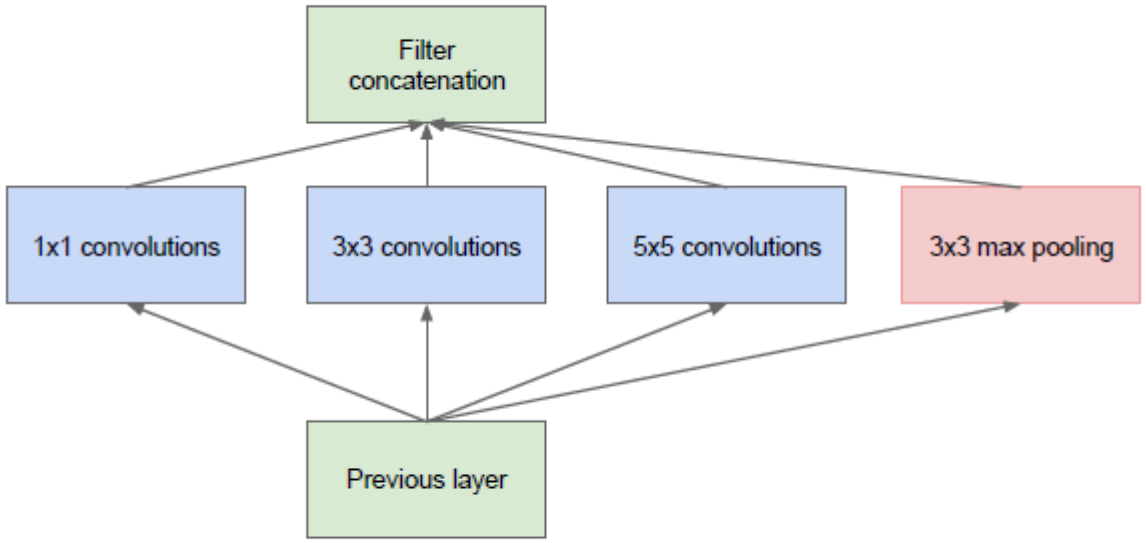

Pillow - WIN


Pillow - WINC


ILSVR red, WINC blue

01 Inception Module

Naïve version

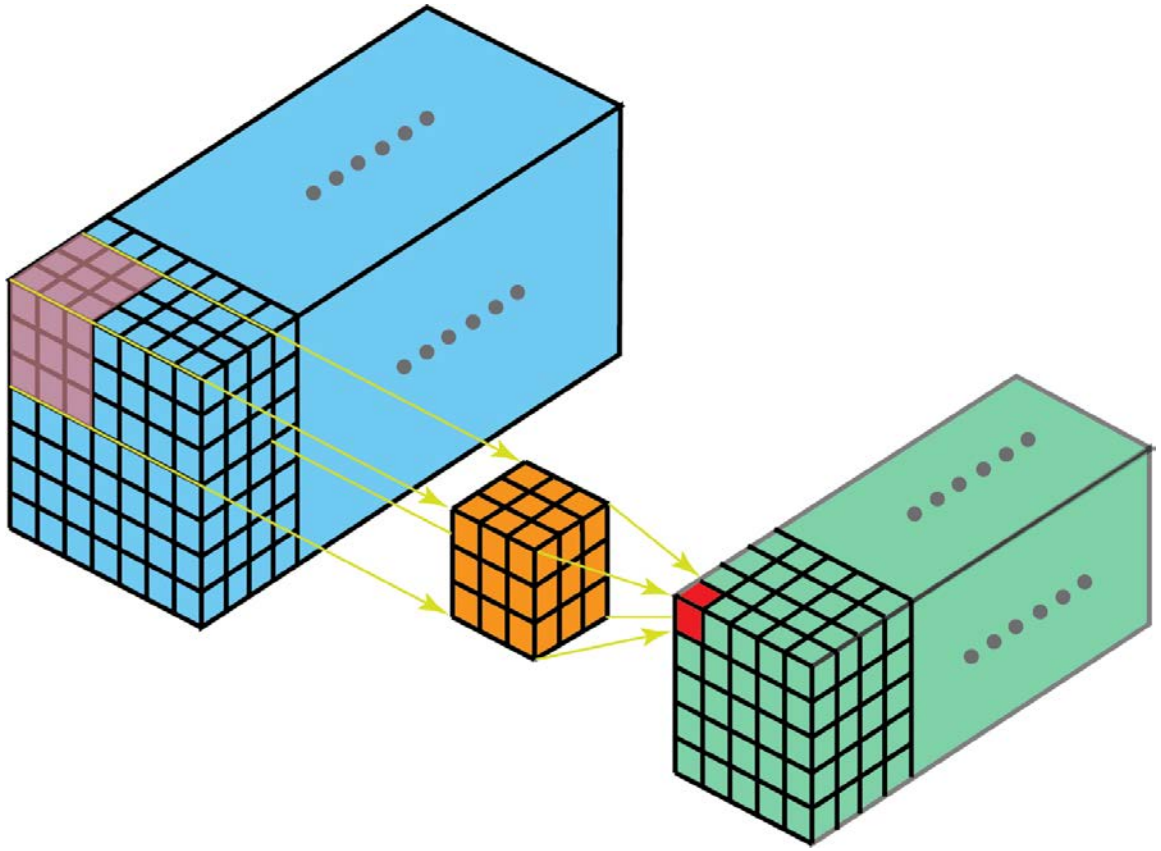


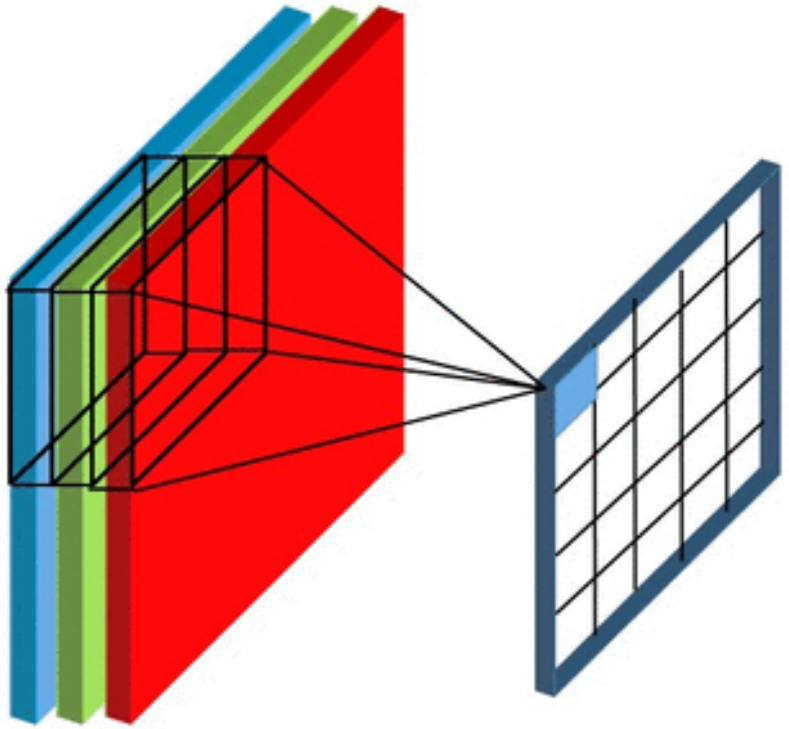
(a) Inception module, naïve version

- Increase in the number of outputs from stage to stage
- Calculation could be expensive on the top of convolutional layer

01 Inception Module

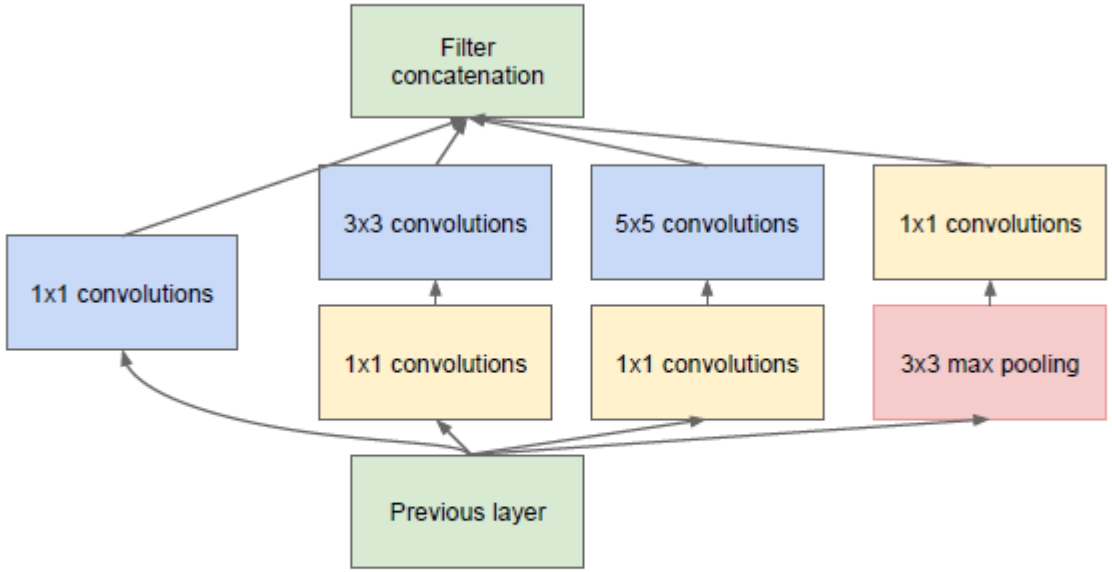
Naïve version





01 Inception Module

Dimension reduction



(b) Inception module with dimension reductions

- Used 1x1 convolutional filter
- Calculation could be cheaper than naïve version
- Used Linear activation(But ReLU could be better)

01 GoogLeNet

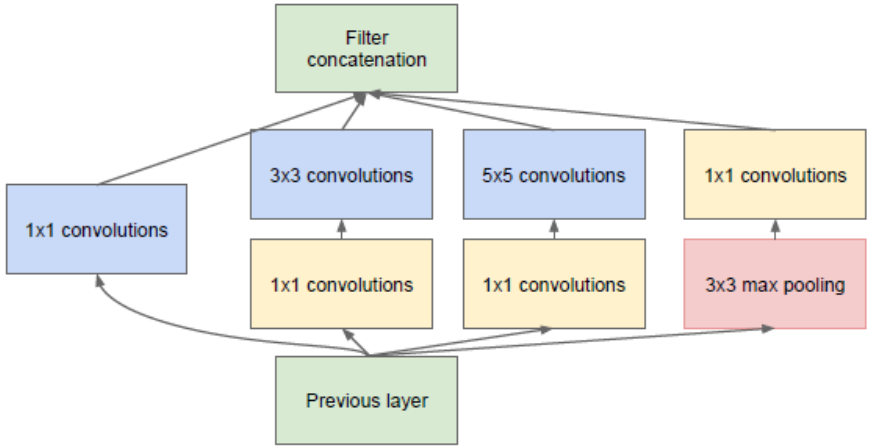
Table

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

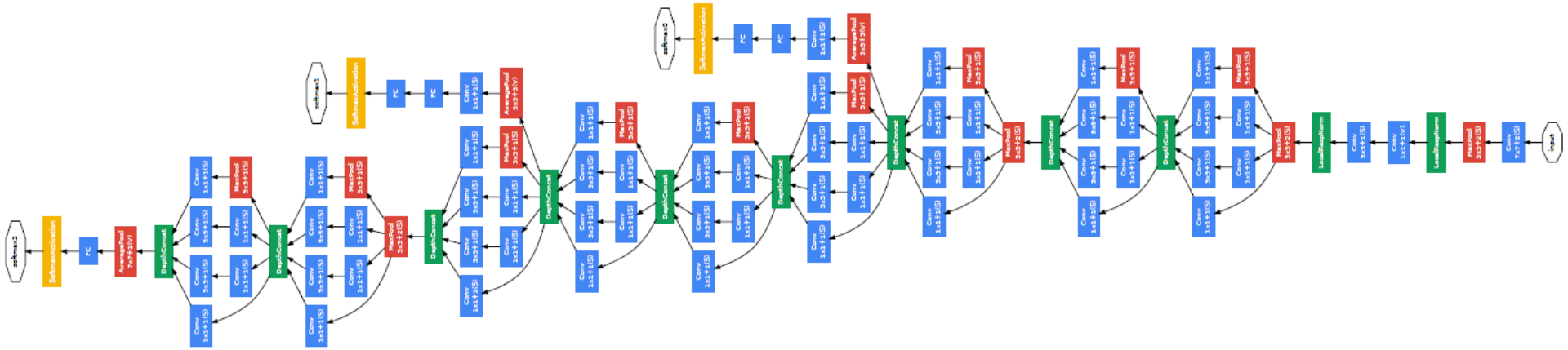
Table 1: GoogLeNet incarnation of the Inception architecture

01 GoogLeNet

GoogleNet



(b) Inception module with dimension reductions



- Large depth of the network
- Auxiliary classifiers are calculated while training
- Their loss gets added to the total loss of the network with a discount weight

01 Code

```
class Inception(nn.Module):
  def __init__(self, in_planes, kernel_1_out, kernel_3_in, kernel_3_out,
               kernel_5_in, kernel_5_out, pool_planes):
    super(Inception, self).__init__()

    # 1x1 conv bolck
    self.block1 = nn.Sequential(
        nn.Conv2d(in_planes, kernel_1_out, kernel_size=1),
        nn.ReLU(inplace=True),
    )

    # 1x1 conv block -> 3x3 conv block
    self.block2 = nn.Sequential(
        nn.Conv2d(in_planes, kernel_3_in, kernel_size=1),
        nn.ReLU(inplace=True),
        nn.Conv2d(kernel_3_in, kernel_3_out, kernel_size=3, padding=1),
        nn.ReLU(inplace=True),
    )

    # 1x1 conv block -> 5x5 conv block
    self.block3 = nn.Sequential(
        nn.Conv2d(in_planes, kernel_5_in, kernel_size=1),
        nn.ReLU(inplace=True),
        nn.Conv2d(kernel_5_in, kernel_5_out, kernel_size=5, padding=2),
        nn.ReLU(inplace=True),
    )
```

```
    # 3x3 pool block -> 1x1 conv block
    self.block4 = nn.Sequential(
        nn.MaxPool2d(kernel_size=3, stride=1, padding=1),
        nn.Conv2d(in_planes, pool_planes, kernel_size=1),
        nn.ReLU(inplace=True),
    )

  def forward(self, x):
    x1 = self.block1(x)
    x2 = self.block2(x)
    x3 = self.block3(x)
    x4 = self.block4(x)
    return torch.cat([x1, x2, x3, x4], 1)
```

01 Code

```python
class BasicConv2d(torch.nn.Module):
    def __init__(self, in_channels, out_channels, **kwargs):
        super(BasicConv2d, self).__init__()
        self.conv = torch.nn.Conv2d(in_channels, out_channels, **kwargs)
        self.batchnorm = torch.nn.BatchNorm2d(out_channels)
        self.relu = torch.nn.ReLU(inplace=True)

    def forward(self, x):
        x = self.conv(x)
        x = self.batchnorm(x)
        x = self.relu(x)
        return x
```

```python
class Aux(torch.nn.Module):
    def __init__(self, in_channels, num_classes):
        super(Aux, self).__init__()
        self.avgpool = torch.nn.AvgPool2d(kernel_size=2, stride=2)
        self.conv = BasicConv2d(in_channels, 128, kernel_size=1)
        self.fc1 = torch.nn.Sequential(torch.nn.Linear(2 * 2 * 128, 256))
        self.fc2 = torch.nn.Linear(256, num_classes)

    def forward(self, x):
        out = self.avgpool(x)
        out = self.conv(out)
        out = out.view(out.size(0), -1)
        out = torch.nn.functional.dropout(out, 0.5, training=self.training)
        out = torch.nn.functional.relu(self.fc1(out), inplace=True)
        out = torch.nn.functional.dropout(out, 0.5, training=self.training)
        out = self.fc2(out)
        return out
```

01 Code

```
class GoogLeNet(nn.Module):
  def __init__(self, base_dim, num_classes=2, batch_size=50, aux_logits=True)
    super(GoogLeNet, self).__init__()
    self.num_classes = num_classes
    self.batch_size = batch_size

    self.post_inception = nn.Sequential(
        nn.Conv2d(3, base_dim, kernel_size=7, padding=1),
        nn.MaxPool2d(kernel_size=3, stride=2, padding=1),
        nn.Conv2d(base_dim,base_dim*3,3,1,1),
        nn.MaxPool2d(3,2,1),
    )

    self.inception_layer1 = nn.Sequential(
       Inception(base_dim*3, 64, 96, 128, 16, 32, 32),
       Inception(base_dim*4, 128, 128, 192, 32, 96, 64),
       nn.MaxPool2d(3, 2, 1),
       Inception(480, 192, 96, 208, 16, 48, 64),
    )

    self.inception_layer2 = nn.Sequential(
        Inception(512 ,160 ,112, 224, 24, 64, 64),
        Inception(512, 128, 128, 256, 24, 64, 64),
        Inception(512, 112, 144, 288, 32, 64, 64),

    )
```

```
    if self.aux_logits:
      self.aux1 = Aux(512, num_classes)
      self.aux2 = Aux(528, num_classes)

    self.avgpool = torch.nn.AdaptiveAvgPool2d((1, 1))
    self.dropout = torch.nn.Dropout(0.4)
    self.fc = torch.nn.Linear(1024, num_classes)

  def forward(self, x):
    out = self.post_inception(x)
    out = self.inception_layer1(out)

    if self.training and self.aux_logits:
      aux1 = self.aux1(out)

    out = self.inception_layer2(out)

    if self.training and self.aux_logits:
      aux2 = self.aux2(out)

    out = self.inception_layer3(out)
    out = self.dropout(out)
    out = out.view(out.size(0),-1)
    out = self.fc(out)
    return out, aux1, aux2
```

Thank You