Model Description

In [1]:

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

from torch.autograd import Variable

def _weights_init(m):
    classname = m.__class__.__name__
    #print(classname)
    if isinstance(m, nn.Linear) or isinstance(m, nn.Conv2d):
        init.kaiming_normal_(m.weight)

class LambdaLayer(nn.Module):
    def __init__(self, lambd):
        super(LambdaLayer, self).__init__()
        self.lambd = lambd

    def forward(self, x):
        return self.lambd(x)


class BasicBlock(nn.Module):
    expansion = 1

    def __init__(self, in_planes, planes, stride=1, option='A'):
        super(BasicBlock, self).__init__()
        self.conv1 = nn.Conv2d(in_planes, planes, kernel_size=3, stride=stride, padding=
1, bias=False)
        self.bn1 = nn.BatchNorm2d(planes)
        self.conv2 = nn.Conv2d(planes, planes, kernel_size=3, stride=1, padding=1, bias=
False)
        self.bn2 = nn.BatchNorm2d(planes)

        self.shortcut = nn.Sequential()
        if stride != 1 or in_planes != planes:
            if option == 'A':
                """
                For CIFAR10 ResNet paper uses option A.
                """
                self.shortcut = LambdaLayer(lambda x:
                                            F.pad(x[:, :, ::2, ::2], (0, 0, 0, 0, plane
s//4, planes//4), "constant", 0))
            elif option == 'B':
                self.shortcut = nn.Sequential(
                     nn.Conv2d(in_planes, self.expansion * planes, kernel_size=1, stride
=stride, bias=False),
                     nn.BatchNorm2d(self.expansion * planes)
                )

    def forward(self, x):
        out = F.relu(self.bn1(self.conv1(x)))
        out = self.bn2(self.conv2(out))
        out += self.shortcut(x)
        out = F.relu(out)
        return out


class ResNet(nn.Module):
    def __init__(self, block, num_blocks, num_classes=100):
        super(ResNet, self).__init__()
        self.in_planes = 16
        self.conv1 = nn.Conv2d(3, 16, kernel_size=3, stride=1, padding=1, bias=False)
        self.bn1 = nn.BatchNorm2d(16)
        self.layer1 = self._make_layer(block, 16, num_blocks[0], stride=1)
        self.layer2 = self._make_layer(block, 32, num_blocks[1], stride=2)
        self.layer3 = self._make_layer(block, 64, num_blocks[2], stride=2)
        self.linear = nn.Linear(64, num_classes)

        self.apply(_weights_init)

    def _make_layer(self, block, planes, num_blocks, stride):
        strides = [stride] + [1]*(num_blocks-1)
        layers = []
        for stride in strides:
            layers.append(block(self.in_planes, planes, stride))
            self.in_planes = planes * block.expansion

        return nn.Sequential(*layers)

    def forward(self, x):
        out = F.relu(self.bn1(self.conv1(x)))
        out = self.layer1(out)
        out = self.layer2(out)
        out = self.layer3(out)
        out = F.avg_pool2d(out, out.size()[3])
        out = out.view(out.size(0), -1)
        out = self.linear(out)
        return out


def resnet20(num_classes=100):
    return ResNet(BasicBlock, [3, 3, 3], num_classes=num_classes)
```

In [2]:

```
model = resnet20(num_classes=100)
```

In [3]:

```
print(model)
```

```
ResNet(
  (conv1): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (layer1): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fal
se)
      (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
      (conv2): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fal
se)
      (bn2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
      (shortcut): Sequential()
    )
    (1): BasicBlock(
      (conv1): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fal
se)
      (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
      (conv2): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fal
se)
      (bn2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
      (shortcut): Sequential()
    )
    (2): BasicBlock(
      (conv1): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fal
se)
      (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
      (conv2): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fal
se)
      (bn2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
      (shortcut): Sequential()
    )
  )
  (layer2): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(16, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=Fal
se)
      (bn1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
      (conv2): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fal
se)
      (bn2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
      (shortcut): LambdaLayer()
    )
    (1): BasicBlock(
      (conv1): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fal
se)
      (bn1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
      (conv2): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fal
se)
      (bn2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
      (shortcut): Sequential()
    )
    (2): BasicBlock(
      (conv1): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fal
se)
      (bn1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
      (conv2): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fal
se)
      (bn2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
      (shortcut): Sequential()
    )
  )
  (layer3): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=Fal
se)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fal
se)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
      (shortcut): LambdaLayer()
    )
    (1): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fal
se)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fal
se)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
      (shortcut): Sequential()
    )
    (2): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fal
se)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fal
se)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
      (shortcut): Sequential()
    )
  )
  (linear): Linear(in_features=64, out_features=100, bias=True)
)
```

In [6]:

```
from torchsummary import summary
```

In [10]:

```
summary(model,(3,224,224))
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 16, 224, 224]             432
       BatchNorm2d-2         [-1, 16, 224, 224]              32
            Conv2d-3         [-1, 16, 224, 224]           2,304
       BatchNorm2d-4         [-1, 16, 224, 224]              32
            Conv2d-5         [-1, 16, 224, 224]           2,304
       BatchNorm2d-6         [-1, 16, 224, 224]              32
        BasicBlock-7         [-1, 16, 224, 224]               0
            Conv2d-8         [-1, 16, 224, 224]           2,304
       BatchNorm2d-9         [-1, 16, 224, 224]              32
           Conv2d-10         [-1, 16, 224, 224]           2,304
      BatchNorm2d-11         [-1, 16, 224, 224]              32
       BasicBlock-12         [-1, 16, 224, 224]               0
           Conv2d-13         [-1, 16, 224, 224]           2,304
      BatchNorm2d-14         [-1, 16, 224, 224]              32
           Conv2d-15         [-1, 16, 224, 224]           2,304
      BatchNorm2d-16         [-1, 16, 224, 224]              32
       BasicBlock-17         [-1, 16, 224, 224]               0
           Conv2d-18         [-1, 32, 112, 112]           4,608
      BatchNorm2d-19         [-1, 32, 112, 112]              64
           Conv2d-20         [-1, 32, 112, 112]           9,216
      BatchNorm2d-21         [-1, 32, 112, 112]              64
      LambdaLayer-22         [-1, 32, 112, 112]               0
       BasicBlock-23         [-1, 32, 112, 112]               0
           Conv2d-24         [-1, 32, 112, 112]           9,216
      BatchNorm2d-25         [-1, 32, 112, 112]              64
           Conv2d-26         [-1, 32, 112, 112]           9,216
      BatchNorm2d-27         [-1, 32, 112, 112]              64
       BasicBlock-28         [-1, 32, 112, 112]               0
           Conv2d-29         [-1, 32, 112, 112]           9,216
      BatchNorm2d-30         [-1, 32, 112, 112]              64
           Conv2d-31         [-1, 32, 112, 112]           9,216
      BatchNorm2d-32         [-1, 32, 112, 112]              64
       BasicBlock-33         [-1, 32, 112, 112]               0
           Conv2d-34           [-1, 64, 56, 56]          18,432
      BatchNorm2d-35           [-1, 64, 56, 56]             128
           Conv2d-36           [-1, 64, 56, 56]          36,864
      BatchNorm2d-37           [-1, 64, 56, 56]             128
      LambdaLayer-38           [-1, 64, 56, 56]               0
       BasicBlock-39           [-1, 64, 56, 56]               0
           Conv2d-40           [-1, 64, 56, 56]          36,864
      BatchNorm2d-41           [-1, 64, 56, 56]             128
           Conv2d-42           [-1, 64, 56, 56]          36,864
      BatchNorm2d-43           [-1, 64, 56, 56]             128
       BasicBlock-44           [-1, 64, 56, 56]               0
           Conv2d-45           [-1, 64, 56, 56]          36,864
      BatchNorm2d-46           [-1, 64, 56, 56]             128
           Conv2d-47           [-1, 64, 56, 56]          36,864
      BatchNorm2d-48           [-1, 64, 56, 56]             128
       BasicBlock-49           [-1, 64, 56, 56]               0
           Linear-50                  [-1, 100]           6,500
================================================================
Total params: 275,572
Trainable params: 275,572
Non-trainable params: 0
```

```
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 177.63
Params size (MB): 1.05
Estimated Total Size (MB): 179.25
----------------------------------------------------------------
```

In [ ]: