1. 1.png

0.719465478842

0.627575757576

0.623673469388

self.conv1 = nn.Conv2d(3, 6, 5)

self.pool = nn.MaxPool2d(2,2)

self.conv2 = nn.Conv2d(6, 16, 5)

self.fc1 = nn.Linear(16\*5\*5, 120)

self.fc2 = nn.Linear(120, 84)

self.fc3 = nn.Linear(84, 10)

1. 1+xavier //accu好一点

0.788752783964

0.636767676768

0.638979591837

self.conv1 = nn.Conv2d(3, 6, 5)

nn.init.xavier\_uniform(self.conv1.weight)

nn.init.constant(self.conv1.bias, 0.1)

self.pool = nn.MaxPool2d(2,2)

self.conv2 = nn.Conv2d(6, 16, 5)

nn.init.xavier\_uniform(self.conv2.weight)

nn.init.constant(self.conv2.bias, 0.1)

self.fc1 = nn.Linear(16\*5\*5, 120)

self.fc2 = nn.Linear(120, 84)

self.fc3 = nn.Linear(84, 10)

1. xavier+BN//快一点 accu并没有提高

0.733429844098

0.625353535354

0.631428571429

1. xavier+BN+Adam Optimizer//更快 accu有提高

0.794721603563

0.65

0.651020408163

1. Kernel变成3

kernel = 3,增加了filter number

self.conv1 = nn.Conv2d(3, 18, 3)

self.conv2 = nn.Conv2d(18, 54, 3)

1. conv3fc3.png //accu提高 stop:10epoch

kernel = 3, conv3,fc3

self.conv1 = nn.Conv2d(3, 32, 3)

self.conv2 = nn.Conv2d(32, 64, 3)

self.conv3 = nn.Conv2d(64,128,3)

0.923340757238

0.740505050505

0.746326530612

1. conv5fc3.png

0.927906458797

0.78

0.783265306122

//

class Net(nn.Module):

def \_\_init\_\_(self):

super(Net, self).\_\_init\_\_()

self.conv1 = nn.Conv2d(3, 32, 3)

nn.init.xavier\_uniform(self.conv1.weight)

nn.init.constant(self.conv1.bias, 0.1)

self.conv1\_bn = nn.BatchNorm2d(32)

self.pool = nn.MaxPool2d(2,2)

self.avgpool=nn.AvgPool2d(2,2)

self.conv2 = nn.Conv2d(32, 64, 3)

nn.init.xavier\_uniform(self.conv2.weight)

nn.init.constant(self.conv2.bias, 0.1)

self.conv2\_bn = nn.BatchNorm2d(64)

self.conv3 = nn.Conv2d(64,128,3)

nn.init.xavier\_uniform(self.conv3.weight)

nn.init.constant(self.conv3.bias, 0.1)

self.conv4 = nn.Conv2d(128,256,3)

nn.init.xavier\_uniform(self.conv4.weight)

nn.init.constant(self.conv4.bias, 0.1)

self.conv5 = nn.Conv2d(256,512,3)

nn.init.xavier\_uniform(self.conv5.weight)

nn.init.constant(self.conv5.bias, 0.1)

self.fc1 = nn.Linear(512\*3\*3, 120)

self.fc1\_bn = nn.BatchNorm1d(120)

self.fc2 = nn.Linear(120, 84)

self.fc2\_bn = nn.BatchNorm1d(84)

self.fc3 = nn.Linear(84, 10)

#self.fc4 = nn.Linear(128, 10)

def forward(self, x):

x = self.pool(F.relu(self.conv1\_bn(self.conv1(x))))

x = F.relu(self.conv2\_bn(self.conv2(x)))

x = F.relu(self.conv3(x))

x = F.relu(self.conv4(x))

x = self.pool(F.relu(self.conv5(x)))

#x = self.avgpool(F.relu(self.conv2\_bn(self.conv2(x))))

#x = self.avgpool(F.relu(self.conv3\_bn(self.conv3(x))))

#x = self.pool(F.relu(self.conv3(x)))

x = x.view(-1, 512\*3\*3)

x = F.relu(self.fc1\_bn(self.fc1(x)))

x = F.relu(self.fc2\_bn(self.fc2(x)))

#x = F.relu(self.fc3(x))

x = self.fc3(x)

return x

1. conv5fc4.png

0.927639198218

0.775050505051

0.781428571429

class Net(nn.Module):

def \_\_init\_\_(self):

super(Net, self).\_\_init\_\_()

self.conv1 = nn.Conv2d(3, 32, 3)

nn.init.xavier\_uniform(self.conv1.weight)

nn.init.constant(self.conv1.bias, 0.1)

self.conv1\_bn = nn.BatchNorm2d(32)

self.pool = nn.MaxPool2d(2,2)

self.avgpool=nn.AvgPool2d(2,2)

self.conv2 = nn.Conv2d(32, 64, 3)

nn.init.xavier\_uniform(self.conv2.weight)

nn.init.constant(self.conv2.bias, 0.1)

self.conv2\_bn = nn.BatchNorm2d(64)

self.conv3 = nn.Conv2d(64,128,3)

nn.init.xavier\_uniform(self.conv3.weight)

nn.init.constant(self.conv3.bias, 0.1)

self.conv4 = nn.Conv2d(128,256,3)

nn.init.xavier\_uniform(self.conv4.weight)

nn.init.constant(self.conv4.bias, 0.1)

self.conv5 = nn.Conv2d(256,512,3)

nn.init.xavier\_uniform(self.conv5.weight)

nn.init.constant(self.conv5.bias, 0.1)

self.fc1 = nn.Linear(512\*3\*3, 480)

self.fc1\_bn = nn.BatchNorm1d(480)

self.fc2 = nn.Linear(480, 120)

self.fc2\_bn = nn.BatchNorm1d(120)

self.fc3 = nn.Linear(120, 84)

self.fc3\_bn = nn.BatchNorm1d(84)

self.fc4 = nn.Linear(84, 10)

def forward(self, x):

x = self.pool(F.relu(self.conv1\_bn(self.conv1(x))))

x = F.relu(self.conv2\_bn(self.conv2(x)))

x = F.relu(self.conv3(x))

x = F.relu(self.conv4(x))

x = self.pool(F.relu(self.conv5(x)))

#x = self.avgpool(F.relu(self.conv2\_bn(self.conv2(x))))

#x = self.avgpool(F.relu(self.conv3\_bn(self.conv3(x))))

#x = self.pool(F.relu(self.conv3(x)))

x = x.view(-1, 512\*3\*3)

x = F.relu(self.fc1\_bn(self.fc1(x)))

x = F.relu(self.fc2\_bn(self.fc2(x)))

x = F.relu(self.fc3\_bn(self.fc3(x)))

#x = F.relu(self.fc3(x))

x = self.fc4(x)

return x

1. con5fc3+2avgpool5.png //把第二个pool换成average pool，accu有提高

0.903541202673

0.785050505051

0.80306122449

class Net(nn.Module):

def \_\_init\_\_(self):

super(Net, self).\_\_init\_\_()

self.conv1 = nn.Conv2d(3, 32, 3)

nn.init.xavier\_uniform(self.conv1.weight)

nn.init.constant(self.conv1.bias, 0.1)

self.conv1\_bn = nn.BatchNorm2d(32)

self.pool = nn.MaxPool2d(2,2)

self.avgpool=nn.AvgPool2d(2,2)

self.conv2 = nn.Conv2d(32, 64, 3)

nn.init.xavier\_uniform(self.conv2.weight)

nn.init.constant(self.conv2.bias, 0.1)

self.conv2\_bn = nn.BatchNorm2d(64)

self.conv3 = nn.Conv2d(64,128,3)

nn.init.xavier\_uniform(self.conv3.weight)

nn.init.constant(self.conv3.bias, 0.1)

self.conv4 = nn.Conv2d(128,256,3)

nn.init.xavier\_uniform(self.conv4.weight)

nn.init.constant(self.conv4.bias, 0.1)

self.conv5 = nn.Conv2d(256,512,3)

nn.init.xavier\_uniform(self.conv5.weight)

nn.init.constant(self.conv5.bias, 0.1)

self.fc1 = nn.Linear(512\*3\*3, 120)

self.fc1\_bn = nn.BatchNorm1d(120)

self.fc2 = nn.Linear(120, 84)

self.fc2\_bn = nn.BatchNorm1d(84)

self.fc3 = nn.Linear(84, 10)

#self.fc4 = nn.Linear(128, 10)

def forward(self, x):

x = self.pool(F.relu(self.conv1\_bn(self.conv1(x))))

x = F.relu(self.conv2\_bn(self.conv2(x)))

x = F.relu(self.conv3(x))

x = F.relu(self.conv4(x))

x = self.avgpool(F.relu(self.conv5(x)))

#x = self.avgpool(F.relu(self.conv2\_bn(self.conv2(x))))

#x = self.avgpool(F.relu(self.conv3\_bn(self.conv3(x))))

#x = self.pool(F.relu(self.conv3(x)))

x = x.view(-1, 512\*3\*3)

x = F.relu(self.fc1\_bn(self.fc1(x)))

x = F.relu(self.fc2\_bn(self.fc2(x)))

#x = F.relu(self.fc3(x))

x = self.fc3(x)

return x

1. con5fc3(2).jpg

0.959643652561

0.81

0.817551020408

class Net(nn.Module):

def \_\_init\_\_(self):

super(Net, self).\_\_init\_\_()

self.conv1 = nn.Conv2d(3, 32, 3)

nn.init.xavier\_uniform(self.conv1.weight)

nn.init.constant(self.conv1.bias, 0.1)

self.conv1\_bn = nn.BatchNorm2d(32)

self.pool = nn.MaxPool2d(2,2)

self.avgpool=nn.AvgPool2d(2,2)

self.conv2 = nn.Conv2d(32, 64, 3)

nn.init.xavier\_uniform(self.conv2.weight)

nn.init.constant(self.conv2.bias, 0.1)

self.conv2\_bn = nn.BatchNorm2d(64)

self.conv3 = nn.Conv2d(64,128,3)

nn.init.xavier\_uniform(self.conv3.weight)

nn.init.constant(self.conv3.bias, 0.1)

self.conv4 = nn.Conv2d(128,256,3)

nn.init.xavier\_uniform(self.conv4.weight)

nn.init.constant(self.conv4.bias, 0.1)

self.conv5 = nn.Conv2d(256,512,3)

nn.init.xavier\_uniform(self.conv5.weight)

nn.init.constant(self.conv5.bias, 0.1)

self.fc1 = nn.Linear(512\*4\*4, 120)

self.fc1\_bn = nn.BatchNorm1d(120)

self.fc2 = nn.Linear(120, 84)

self.fc2\_bn = nn.BatchNorm1d(84)

self.fc3 = nn.Linear(84, 10)

#self.fc4 = nn.Linear(128, 10)

def forward(self, x):

x = F.relu(self.conv1\_bn(self.conv1(x)))

x = self.pool(F.relu(self.conv2\_bn(self.conv2(x))))

x = F.relu(self.conv3(x))

x = F.relu(self.conv4(x))

x = self.pool(F.relu(self.conv5(x)))

#x = self.avgpool(F.relu(self.conv2\_bn(self.conv2(x))))

#x = self.avgpool(F.relu(self.conv3\_bn(self.conv3(x))))

#x = self.pool(F.relu(self.conv3(x)))

x = x.view(-1, 512\*4\*4)

x = F.relu(self.fc1\_bn(self.fc1(x)))

x = F.relu(self.fc2\_bn(self.fc2(x)))

#x = F.relu(self.fc3(x))

x = self.fc3(x)

return x

1. conv5fc3 5x5x5

0.922316258352

0.774141414141

0.780816326531

class Net(nn.Module):

def \_\_init\_\_(self):

super(Net, self).\_\_init\_\_()

self.conv1 = nn.Conv2d(3, 32, 5)

nn.init.xavier\_uniform(self.conv1.weight)

nn.init.constant(self.conv1.bias, 0.1)

self.conv1\_bn = nn.BatchNorm2d(32)

self.pool = nn.MaxPool2d(2,2)

self.avgpool=nn.AvgPool2d(2,2)

self.conv2 = nn.Conv2d(32, 64, 5)

nn.init.xavier\_uniform(self.conv2.weight)

nn.init.constant(self.conv2.bias, 0.1)

self.conv2\_bn = nn.BatchNorm2d(64)

self.conv3 = nn.Conv2d(64,128,5)

nn.init.xavier\_uniform(self.conv3.weight)

nn.init.constant(self.conv3.bias, 0.1)

self.conv4 = nn.Conv2d(128,256,3)

nn.init.xavier\_uniform(self.conv4.weight)

nn.init.constant(self.conv4.bias, 0.1)

self.conv5 = nn.Conv2d(256,512,3)

nn.init.xavier\_uniform(self.conv5.weight)

nn.init.constant(self.conv5.bias, 0.1)

self.fc1 = nn.Linear(512\*2\*2, 120)

self.fc1\_bn = nn.BatchNorm1d(120)

self.fc2 = nn.Linear(120, 84)

self.fc2\_bn = nn.BatchNorm1d(84)

self.fc3 = nn.Linear(84, 10)

#self.fc4 = nn.Linear(128, 10)

def forward(self, x):

x = F.relu(self.conv1\_bn(self.conv1(x)))

x = self.pool(F.relu(self.conv2\_bn(self.conv2(x))))

x = F.relu(self.conv3(x))

x = F.relu(self.conv4(x))

x = self.avgpool(F.relu(self.conv5(x)))

#x = self.avgpool(F.relu(self.conv2\_bn(self.conv2(x))))

#x = self.avgpool(F.relu(self.conv3\_bn(self.conv3(x))))

#x = self.pool(F.relu(self.conv3(x)))

x = x.view(-1, 512\*2\*2)

x = F.relu(self.fc1\_bn(self.fc1(x)))

x = F.relu(self.fc2\_bn(self.fc2(x)))

#x = F.relu(self.fc3(x))

x = self.fc3(x)

return x

1. conv5fc3(3)

0.950556792873

0.824545454545

0.835918367347

class Net(nn.Module):

def \_\_init\_\_(self):

super(Net, self).\_\_init\_\_()

self.conv1 = nn.Conv2d(3, 64, 3)

nn.init.xavier\_uniform(self.conv1.weight)

nn.init.constant(self.conv1.bias, 0.1)

self.conv1\_bn = nn.BatchNorm2d(64)

self.pool = nn.MaxPool2d(2,2)

self.avgpool=nn.AvgPool2d(2,2)

self.conv2 = nn.Conv2d(64, 256, 3)

nn.init.xavier\_uniform(self.conv2.weight)

nn.init.constant(self.conv2.bias, 0.1)

self.conv2\_bn = nn.BatchNorm2d(256)

self.conv3 = nn.Conv2d(256,512,3)

nn.init.xavier\_uniform(self.conv3.weight)

nn.init.constant(self.conv3.bias, 0.1)

self.conv4 = nn.Conv2d(512,1024,3)

nn.init.xavier\_uniform(self.conv4.weight)

nn.init.constant(self.conv4.bias, 0.1)

self.conv5 = nn.Conv2d(1024,2048,3)

nn.init.xavier\_uniform(self.conv5.weight)

nn.init.constant(self.conv5.bias, 0.1)

self.fc1 = nn.Linear(2048\*4\*4, 120)

self.fc1\_bn = nn.BatchNorm1d(120)

self.fc2 = nn.Linear(120, 84)

self.fc2\_bn = nn.BatchNorm1d(84)

self.fc3 = nn.Linear(84, 10)

#self.fc4 = nn.Linear(128, 10)

def forward(self, x):

x = F.relu(self.conv1\_bn(self.conv1(x)))

x = self.pool(F.relu(self.conv2\_bn(self.conv2(x))))

x = F.relu(self.conv3(x))

x = F.relu(self.conv4(x))

x = self.avgpool(F.relu(self.conv5(x)))

#x = self.avgpool(F.relu(self.conv2\_bn(self.conv2(x))))

#x = self.avgpool(F.relu(self.conv3\_bn(self.conv3(x))))

#x = self.pool(F.relu(self.conv3(x)))

x = x.view(-1, 2048\*4\*4)

x = F.relu(self.fc1\_bn(self.fc1(x)))

x = F.relu(self.fc2\_bn(self.fc2(x)))

#x = F.relu(self.fc3(x))

x = self.fc3(x)

return x

1. conv6fc3.jpg

0.942360801782

0.811212121212

0.823673469388

class Net(nn.Module):

def \_\_init\_\_(self):

super(Net, self).\_\_init\_\_()

self.conv1 = nn.Conv2d(3, 32, 3)

nn.init.xavier\_uniform(self.conv1.weight)

nn.init.constant(self.conv1.bias, 0.1)

self.conv1\_bn = nn.BatchNorm2d(32)

self.pool = nn.MaxPool2d(2,2)

self.avgpool=nn.AvgPool2d(2,2)

self.conv2 = nn.Conv2d(32, 64, 3)

nn.init.xavier\_uniform(self.conv2.weight)

nn.init.constant(self.conv2.bias, 0.1)

self.conv2\_bn = nn.BatchNorm2d(64)

self.conv3 = nn.Conv2d(64,128,3)

nn.init.xavier\_uniform(self.conv3.weight)

nn.init.constant(self.conv3.bias, 0.1)

self.conv4 = nn.Conv2d(128,256,3)

nn.init.xavier\_uniform(self.conv4.weight)

nn.init.constant(self.conv4.bias, 0.1)

self.conv5 = nn.Conv2d(256,512,3)

nn.init.xavier\_uniform(self.conv5.weight)

nn.init.constant(self.conv5.bias, 0.1)

self.conv6 = nn.Conv2d(512,2048,3)

nn.init.xavier\_uniform(self.conv6.weight)

nn.init.constant(self.conv6.bias, 0.1)

self.fc1 = nn.Linear(2048\*3\*3, 120)

self.fc1\_bn = nn.BatchNorm1d(120)

self.fc2 = nn.Linear(120, 84)

self.fc2\_bn = nn.BatchNorm1d(84)

self.fc3 = nn.Linear(84, 10)

#self.fc4 = nn.Linear(128, 10)

def forward(self, x):

x = F.relu(self.conv1\_bn(self.conv1(x)))

x = self.pool(F.relu(self.conv2\_bn(self.conv2(x))))

x = F.relu(self.conv3(x))

x = F.relu(self.conv4(x))

x = F.relu(self.conv5(x))

#x = self.avgpool(F.relu(self.conv2\_bn(self.conv2(x))))

#x = self.avgpool(F.relu(self.conv3\_bn(self.conv3(x))))

x = self.pool(F.relu(self.conv6(x)))

x = x.view(-1, 2048\*3\*3)

x = F.relu(self.fc1\_bn(self.fc1(x)))

x = F.relu(self.fc2\_bn(self.fc2(x)))

#x = F.relu(self.fc3(x))

x = self.fc3(x)

return x