Assignment 1 CNN for image recognition (20p)

1901213121

黄芝琪

Environment: Ubuntu, 1080 Titan, Pytorch, tensorboardX

Set up: EPOCH = 40, Batch Size = 64,

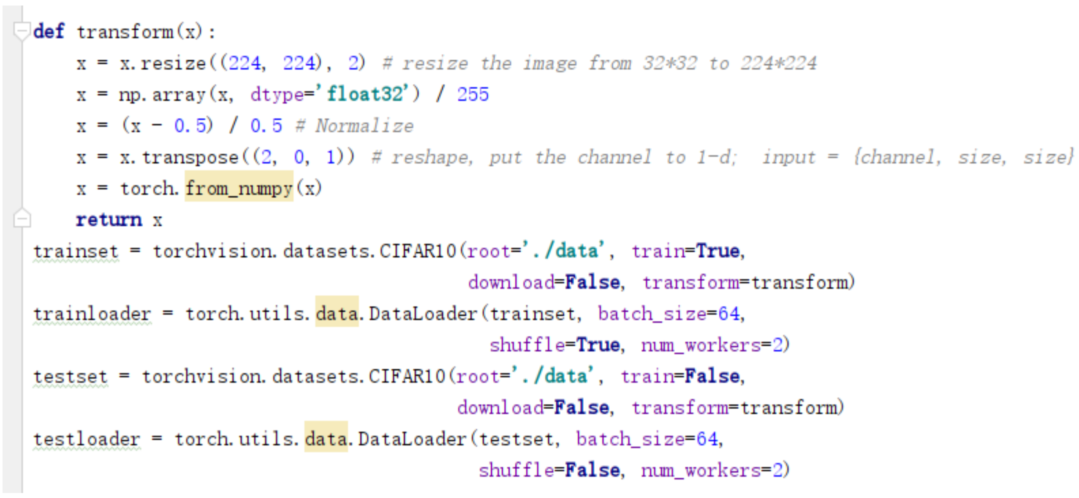
Learning Rate = 0.01, momentum=0.9, weight_decay=5e-4

Step1. Dataset download

Load and resize the data with torchvision and transform.

Remember to resize the image from 32*32 to 224*224, for the sake of not only align the network with the output size in homework description, but also improving the training accuracy.

Train: Test = 50000: 10000



```
def transform(x):
    x = x.resize((224, 224), 2) # resize the image from 32*32 to 224*224
    x = np.array(x, dtype='float32') / 255
    x = (x - 0.5) / 0.5 # Normalize
    x = x.transpose((2, 0, 1)) # reshape, put the channel to 1-d;  input = {channel, size, size}
    x = torch.from_numpy(x)
    return x
trainset = torchvision.datasets.CIFAR10(root='./data', train=True,
                                        download=False, transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=64,
                                          shuffle=True, num_workers=2)
testset = torchvision.datasets.CIFAR10(root='./data', train=False,
                                       download=False, transform=transform)
testloader = torch.utils.data.DataLoader(testset, batch_size=64,
                                         shuffle=False, num_workers=2)
```

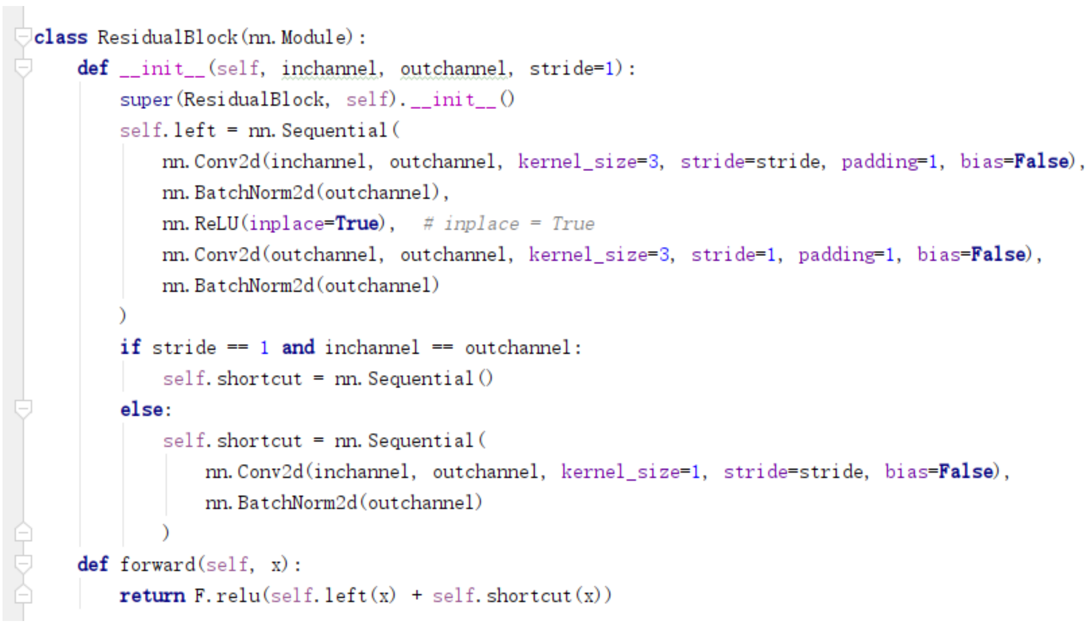
Fig 1. Data loading

Step2. Create ResNet18 shown in Table 1 (19p)+

layer name	output size	18-layer
conv1	112×112	
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$
	1×1	
FLOPs		1.8×10^9

Table 1. Architectures for ResNet18

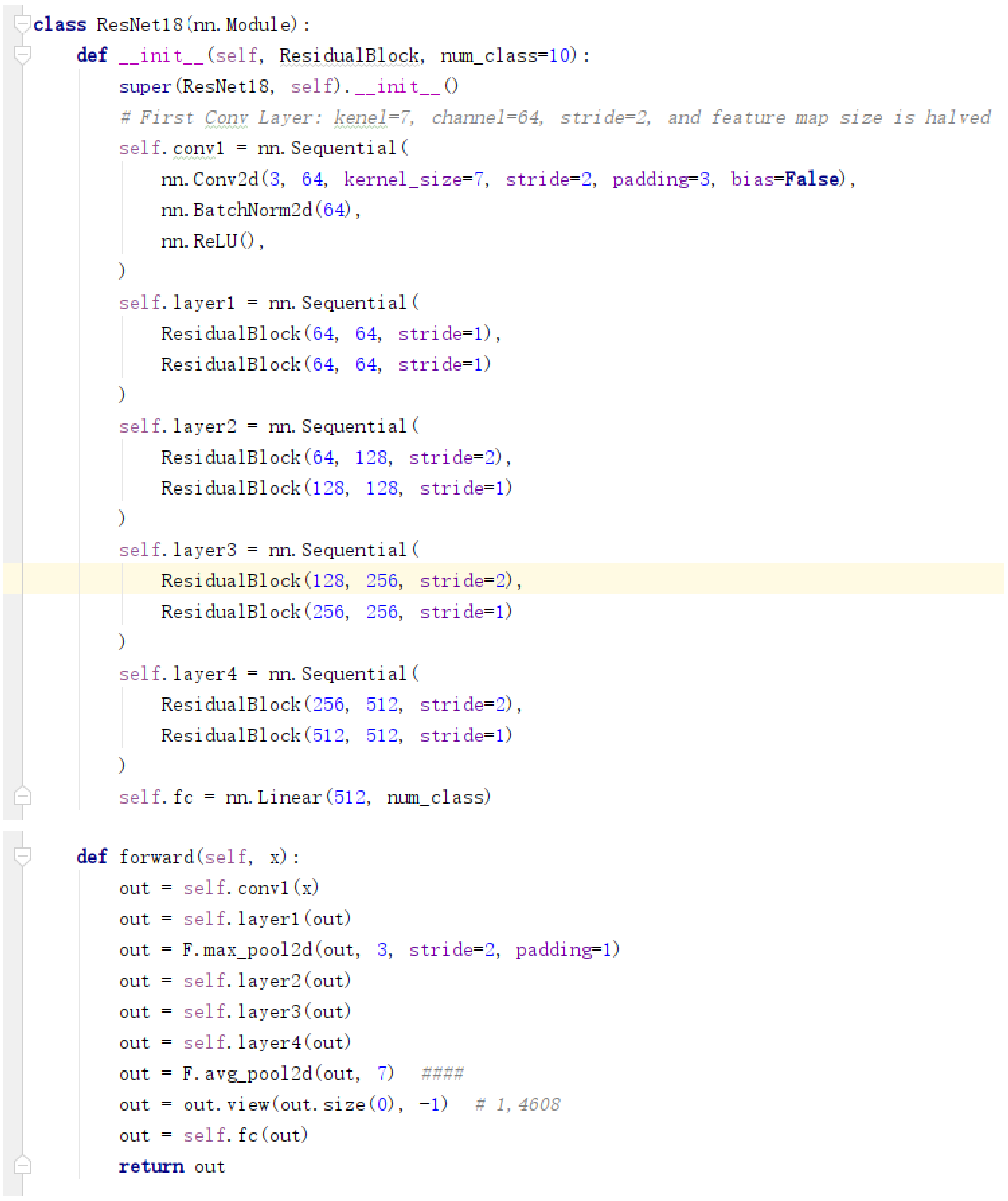
First, define a ResidualBlock according to the ResNet paper[1].



```python
class ResidualBlock(nn.Module):
    def __init__(self, inchannel, outchannel, stride=1):
        super(ResidualBlock, self).__init__()
        self.left = nn.Sequential(
            nn.Conv2d(inchannel, outchannel, kernel_size=3, stride=stride, padding=1, bias=False),
            nn.BatchNorm2d(outchannel),
            nn.ReLU(inplace=True),  # inplace = True
            nn.Conv2d(outchannel, outchannel, kernel_size=3, stride=1, padding=1, bias=False),
            nn.BatchNorm2d(outchannel)
        )
        if stride == 1 and inchannel == outchannel:
            self.shortcut = nn.Sequential()
        else:
            self.shortcut = nn.Sequential(
                nn.Conv2d(inchannel, outchannel, kernel_size=1, stride=stride, bias=False),
                nn.BatchNorm2d(outchannel)
            )
    def forward(self, x):
        return F.relu(self.left(x) + self.shortcut(x))
```

Fig 2. ResidualBlock with shortcut

Then define ResNet18 with above ResidualBlock.



```python
class ResNet18(nn.Module):
    def __init__(self, ResidualBlock, num_class=10):
        super(ResNet18, self).__init__()
        # First Conv Layer: kenel=7, channel=64, stride=2, and feature map size is halved
        self.conv1 = nn.Sequential(
            nn.Conv2d(3, 64, kernel_size=7, stride=2, padding=3, bias=False),
            nn.BatchNorm2d(64),
            nn.ReLU(),
        )
        self.layer1 = nn.Sequential(
            ResidualBlock(64, 64, stride=1),
            ResidualBlock(64, 64, stride=1)
        )
        self.layer2 = nn.Sequential(
            ResidualBlock(64, 128, stride=2),
            ResidualBlock(128, 128, stride=1)
        )
        self.layer3 = nn.Sequential(
            ResidualBlock(128, 256, stride=2),
            ResidualBlock(256, 256, stride=1)
        )
        self.layer4 = nn.Sequential(
            ResidualBlock(256, 512, stride=2),
            ResidualBlock(512, 512, stride=1)
        )
        self.fc = nn.Linear(512, num_class)

    def forward(self, x):
        out = self.conv1(x)
        out = self.layer1(out)
        out = F.max_pool2d(out, 3, stride=2, padding=1)
        out = self.layer2(out)
        out = self.layer3(out)
        out = self.layer4(out)
        out = F.avg_pool2d(out, 7)  ####
        out = out.view(out.size(0), -1)  # 1, 4608
        out = self.fc(out)
        return out
```

Fig 3. Architecture for ResNet18

```
 (conv1): Sequential(
   (0): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
   (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
   (2): ReLU()
 )
 (layer1): Sequential(
   (0): ResidualBlock(
     (left): Sequential(
       (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
       (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
       (2): ReLU(inplace)
       (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
       (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
     )
     (shortcut): Sequential()
   )
   (1): ResidualBlock(
     (left): Sequential(
       (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
       (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
       (2): ReLU(inplace)
       (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
       (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
     )
     (shortcut): Sequential()
   )
 )
```

Fig 4. Layer of ‘Conv1’ and ‘Layer1’

In order to ensure the computation complexity in each layer is the same, while the feature map size halved, channel size will double.

Step3. Final accuracy of training and testing for the CIFAR-10

In the training, I set the batch size 64, thus one epoch finish with 784 iterations gradient update. And 15,680 steps were executed after I train the model for 20 epochs, and I use this model for testing the CIFAR-10 test set.

The final training accuracy: 99.0%

And the final test result is: 88.0%

Step4. Visualize the learning progress (tensorboardX)

Then, I use tensorboardX to visualize the train loss train accuracy, which are showed on the following picture:

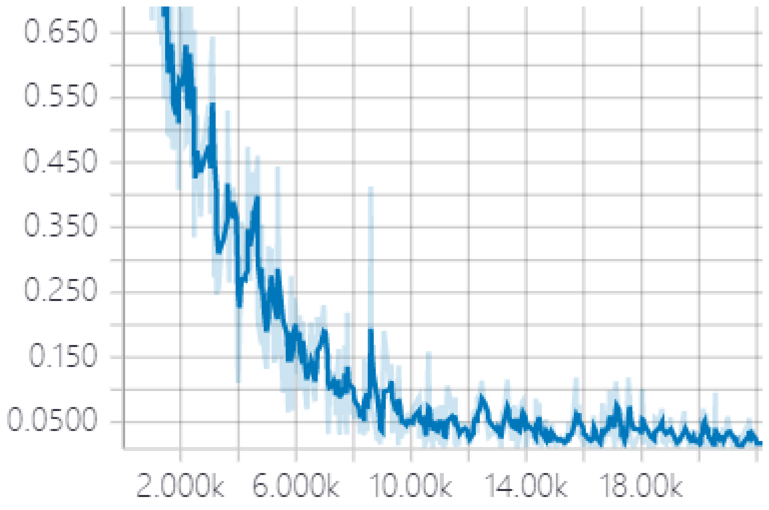


Fig 4. Variations of training loss(cross entropy)

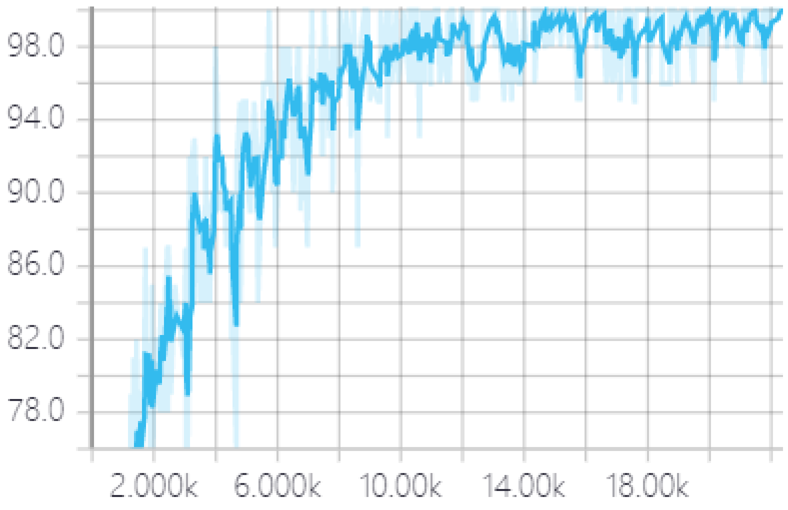


Fig 5. Accuracy for training dataset

Step5. Filter Visualization

Visualization is carried out according to a GitHub Repo: pytorch-cnn-visualizations[3].

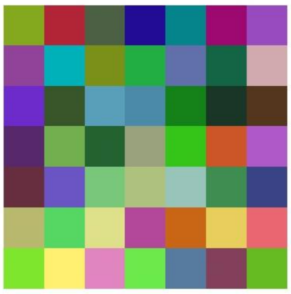


Fig 6. One of the filter in the layer Conv1

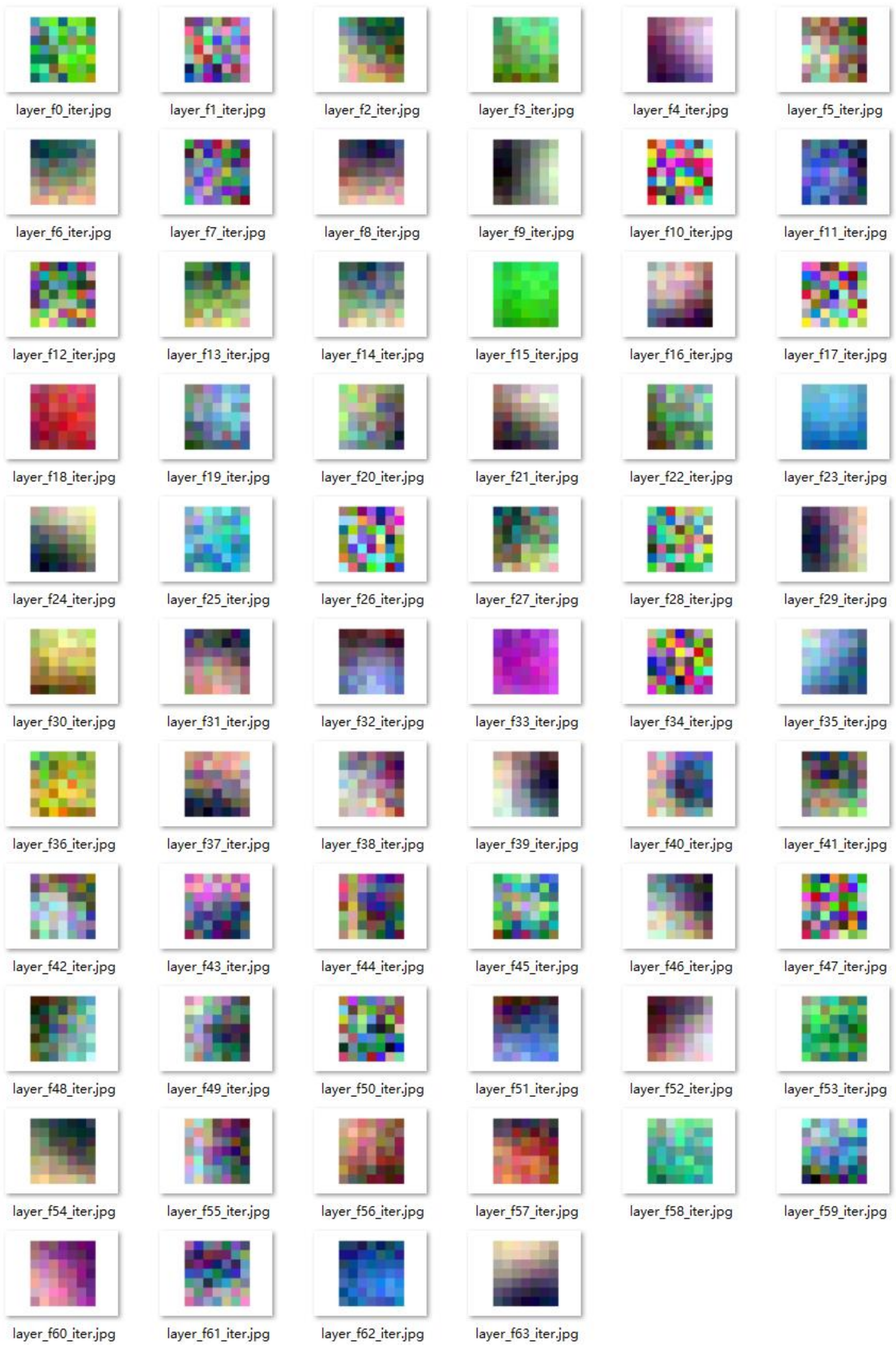


Fig 7. Visualization the 64 filters in the layer Conv1 (7*7*64)

Step6. Feature mapping Visualization

Visualize the feature maps of the layer Conv1 and also the last layer in the block Conv5_x in Table 1.

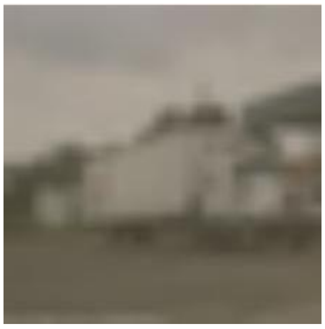


Fig 8. The Original Image (224*224*3 after resize)

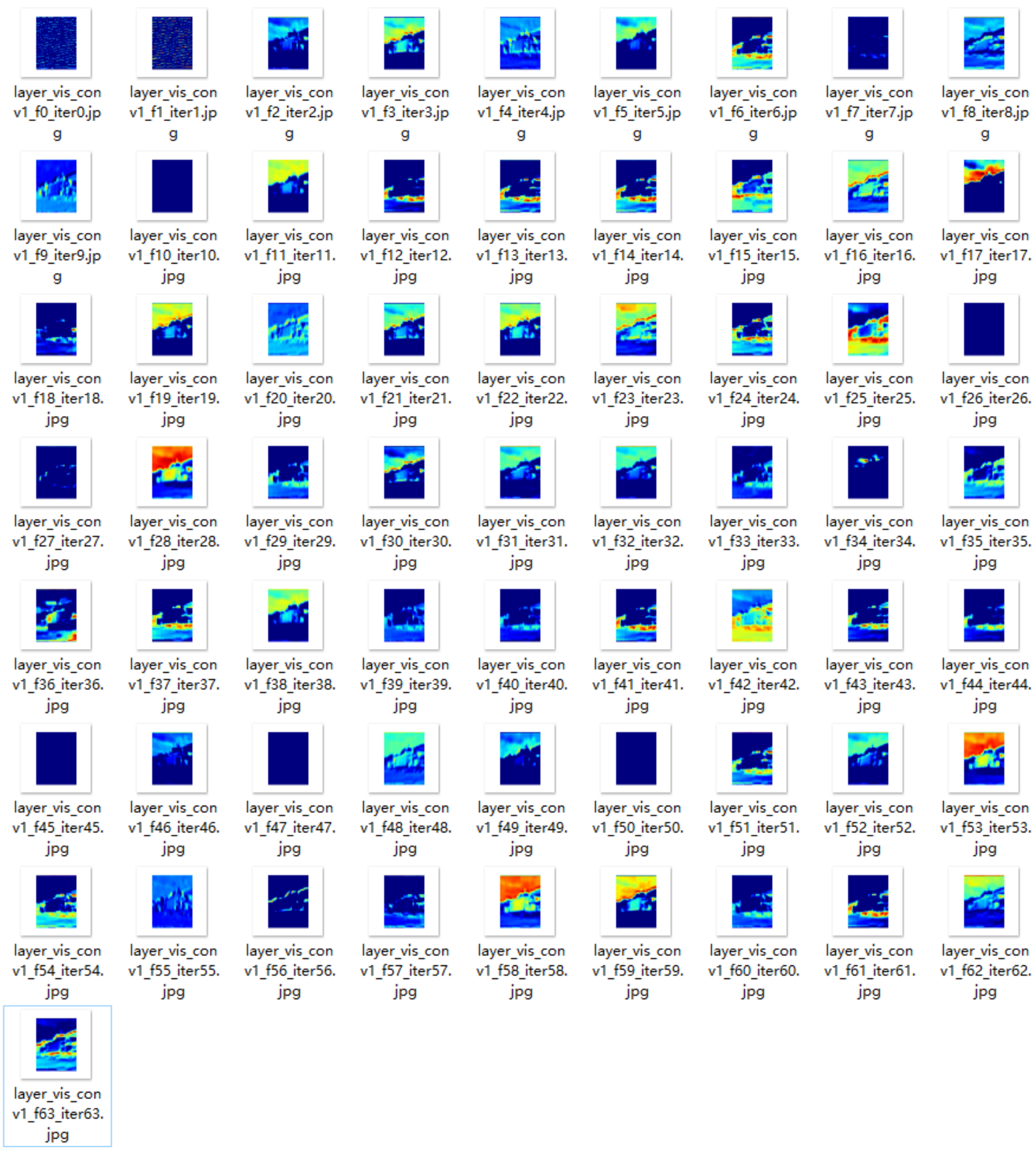

layer_vis_conv1_f0_iter0.jpg
layer_vis_conv1_f1_iter1.jpg
layer_vis_conv1_f2_iter2.jpg
layer_vis_conv1_f3_iter3.jpg
layer_vis_conv1_f4_iter4.jpg
layer_vis_conv1_f5_iter5.jpg
layer_vis_conv1_f6_iter6.jpg
layer_vis_conv1_f7_iter7.jpg
layer_vis_conv1_f8_iter8.jpg
layer_vis_conv1_f9_iter9.jpg
layer_vis_conv1_f10_iter10.jpg
layer_vis_conv1_f11_iter11.jpg
layer_vis_conv1_f12_iter12.jpg
layer_vis_conv1_f13_iter13.jpg
layer_vis_conv1_f14_iter14.jpg
layer_vis_conv1_f15_iter15.jpg
layer_vis_conv1_f16_iter16.jpg
layer_vis_conv1_f17_iter17.jpg
layer_vis_conv1_f18_iter18.jpg
layer_vis_conv1_f19_iter19.jpg
layer_vis_conv1_f20_iter20.jpg
layer_vis_conv1_f21_iter21.jpg
layer_vis_conv1_f22_iter22.jpg
layer_vis_conv1_f23_iter23.jpg
layer_vis_conv1_f24_iter24.jpg
layer_vis_conv1_f25_iter25.jpg
layer_vis_conv1_f26_iter26.jpg
layer_vis_conv1_f27_iter27.jpg
layer_vis_conv1_f28_iter28.jpg
layer_vis_conv1_f29_iter29.jpg
layer_vis_conv1_f30_iter30.jpg
layer_vis_conv1_f31_iter31.jpg
layer_vis_conv1_f32_iter32.jpg
layer_vis_conv1_f33_iter33.jpg
layer_vis_conv1_f34_iter34.jpg
layer_vis_conv1_f35_iter35.jpg
layer_vis_conv1_f36_iter36.jpg
layer_vis_conv1_f37_iter37.jpg
layer_vis_conv1_f38_iter38.jpg
layer_vis_conv1_f39_iter39.jpg
layer_vis_conv1_f40_iter40.jpg
layer_vis_conv1_f41_iter41.jpg
layer_vis_conv1_f42_iter42.jpg
layer_vis_conv1_f43_iter43.jpg
layer_vis_conv1_f44_iter44.jpg
layer_vis_conv1_f45_iter45.jpg
layer_vis_conv1_f46_iter46.jpg
layer_vis_conv1_f47_iter47.jpg
layer_vis_conv1_f48_iter48.jpg
layer_vis_conv1_f49_iter49.jpg
layer_vis_conv1_f50_iter50.jpg
layer_vis_conv1_f51_iter51.jpg
layer_vis_conv1_f52_iter52.jpg
layer_vis_conv1_f53_iter53.jpg
layer_vis_conv1_f54_iter54.jpg
layer_vis_conv1_f55_iter55.jpg
layer_vis_conv1_f56_iter56.jpg
layer_vis_conv1_f57_iter57.jpg
layer_vis_conv1_f58_iter58.jpg
layer_vis_conv1_f59_iter59.jpg
layer_vis_conv1_f60_iter60.jpg
layer_vis_conv1_f61_iter61.jpg
layer_vis_conv1_f62_iter62.jpg
layer_vis_conv1_f63_iter63.jpg

Fig 9. Feature maps of layer Conv1 (112*112*64)

Note that because of the space limitation, there are 512 channel in feature maps of layer Conv5_x and I just show 220 of them.

If it is essential to submit all the pictures, please contact with me through 1901213121@pku.edu.cn.

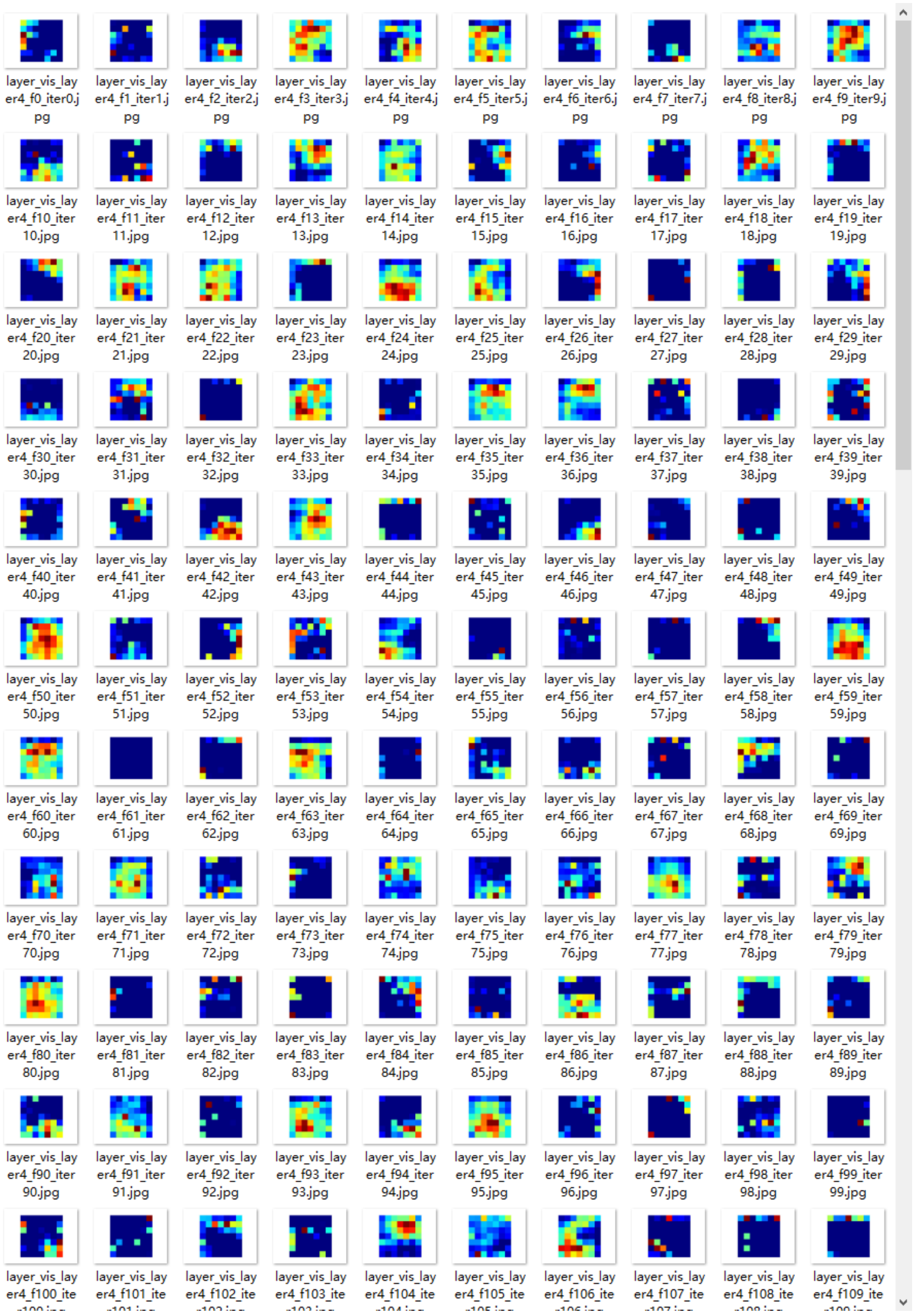

layer_vis_layer4_f0_iter0.jpg
layer_vis_layer4_f1_iter1.jpg
layer_vis_layer4_f2_iter2.jpg
layer_vis_layer4_f3_iter3.jpg
layer_vis_layer4_f4_iter4.jpg
layer_vis_layer4_f5_iter5.jpg
layer_vis_layer4_f6_iter6.jpg
layer_vis_layer4_f7_iter7.jpg
layer_vis_layer4_f8_iter8.jpg
layer_vis_layer4_f9_iter9.jpg
layer_vis_layer4_f10_iter10.jpg
layer_vis_layer4_f11_iter11.jpg
layer_vis_layer4_f12_iter12.jpg
layer_vis_layer4_f13_iter13.jpg
layer_vis_layer4_f14_iter14.jpg
layer_vis_layer4_f15_iter15.jpg
layer_vis_layer4_f16_iter16.jpg
layer_vis_layer4_f17_iter17.jpg
layer_vis_layer4_f18_iter18.jpg
layer_vis_layer4_f19_iter19.jpg
layer_vis_layer4_f20_iter20.jpg
layer_vis_layer4_f21_iter21.jpg
layer_vis_layer4_f22_iter22.jpg
layer_vis_layer4_f23_iter23.jpg
layer_vis_layer4_f24_iter24.jpg
layer_vis_layer4_f25_iter25.jpg
layer_vis_layer4_f26_iter26.jpg
layer_vis_layer4_f27_iter27.jpg
layer_vis_layer4_f28_iter28.jpg
layer_vis_layer4_f29_iter29.jpg
layer_vis_layer4_f30_iter30.jpg
layer_vis_layer4_f31_iter31.jpg
layer_vis_layer4_f32_iter32.jpg
layer_vis_layer4_f33_iter33.jpg
layer_vis_layer4_f34_iter34.jpg
layer_vis_layer4_f35_iter35.jpg
layer_vis_layer4_f36_iter36.jpg
layer_vis_layer4_f37_iter37.jpg
layer_vis_layer4_f38_iter38.jpg
layer_vis_layer4_f39_iter39.jpg
layer_vis_layer4_f40_iter40.jpg
layer_vis_layer4_f41_iter41.jpg
layer_vis_layer4_f42_iter42.jpg
layer_vis_layer4_f43_iter43.jpg
layer_vis_layer4_f44_iter44.jpg
layer_vis_layer4_f45_iter45.jpg
layer_vis_layer4_f46_iter46.jpg
layer_vis_layer4_f47_iter47.jpg
layer_vis_layer4_f48_iter48.jpg
layer_vis_layer4_f49_iter49.jpg
layer_vis_layer4_f50_iter50.jpg
layer_vis_layer4_f51_iter51.jpg
layer_vis_layer4_f52_iter52.jpg
layer_vis_layer4_f53_iter53.jpg
layer_vis_layer4_f54_iter54.jpg
layer_vis_layer4_f55_iter55.jpg
layer_vis_layer4_f56_iter56.jpg
layer_vis_layer4_f57_iter57.jpg
layer_vis_layer4_f58_iter58.jpg
layer_vis_layer4_f59_iter59.jpg
layer_vis_layer4_f60_iter60.jpg
layer_vis_layer4_f61_iter61.jpg
layer_vis_layer4_f62_iter62.jpg
layer_vis_layer4_f63_iter63.jpg
layer_vis_layer4_f64_iter64.jpg
layer_vis_layer4_f65_iter65.jpg
layer_vis_layer4_f66_iter66.jpg
layer_vis_layer4_f67_iter67.jpg
layer_vis_layer4_f68_iter68.jpg
layer_vis_layer4_f69_iter69.jpg
layer_vis_layer4_f70_iter70.jpg
layer_vis_layer4_f71_iter71.jpg
layer_vis_layer4_f72_iter72.jpg
layer_vis_layer4_f73_iter73.jpg
layer_vis_layer4_f74_iter74.jpg
layer_vis_layer4_f75_iter75.jpg
layer_vis_layer4_f76_iter76.jpg
layer_vis_layer4_f77_iter77.jpg
layer_vis_layer4_f78_iter78.jpg
layer_vis_layer4_f79_iter79.jpg
layer_vis_layer4_f80_iter80.jpg
layer_vis_layer4_f81_iter81.jpg
layer_vis_layer4_f82_iter82.jpg
layer_vis_layer4_f83_iter83.jpg
layer_vis_layer4_f84_iter84.jpg
layer_vis_layer4_f85_iter85.jpg
layer_vis_layer4_f86_iter86.jpg
layer_vis_layer4_f87_iter87.jpg
layer_vis_layer4_f88_iter88.jpg
layer_vis_layer4_f89_iter89.jpg
layer_vis_layer4_f90_iter90.jpg
layer_vis_layer4_f91_iter91.jpg
layer_vis_layer4_f92_iter92.jpg
layer_vis_layer4_f93_iter93.jpg
layer_vis_layer4_f94_iter94.jpg
layer_vis_layer4_f95_iter95.jpg
layer_vis_layer4_f96_iter96.jpg
layer_vis_layer4_f97_iter97.jpg
layer_vis_layer4_f98_iter98.jpg
layer_vis_layer4_f99_iter99.jpg
layer_vis_layer4_f100_ite
layer_vis_layer4_f101_ite
layer_vis_layer4_f102_ite
layer_vis_layer4_f103_ite
layer_vis_layer4_f104_ite
layer_vis_layer4_f105_ite
layer_vis_layer4_f106_ite
layer_vis_layer4_f107_ite
layer_vis_layer4_f108_ite
layer_vis_layer4_f109_ite

Fig 10(1). Feature maps of layer Conv5_x (7*7*512)

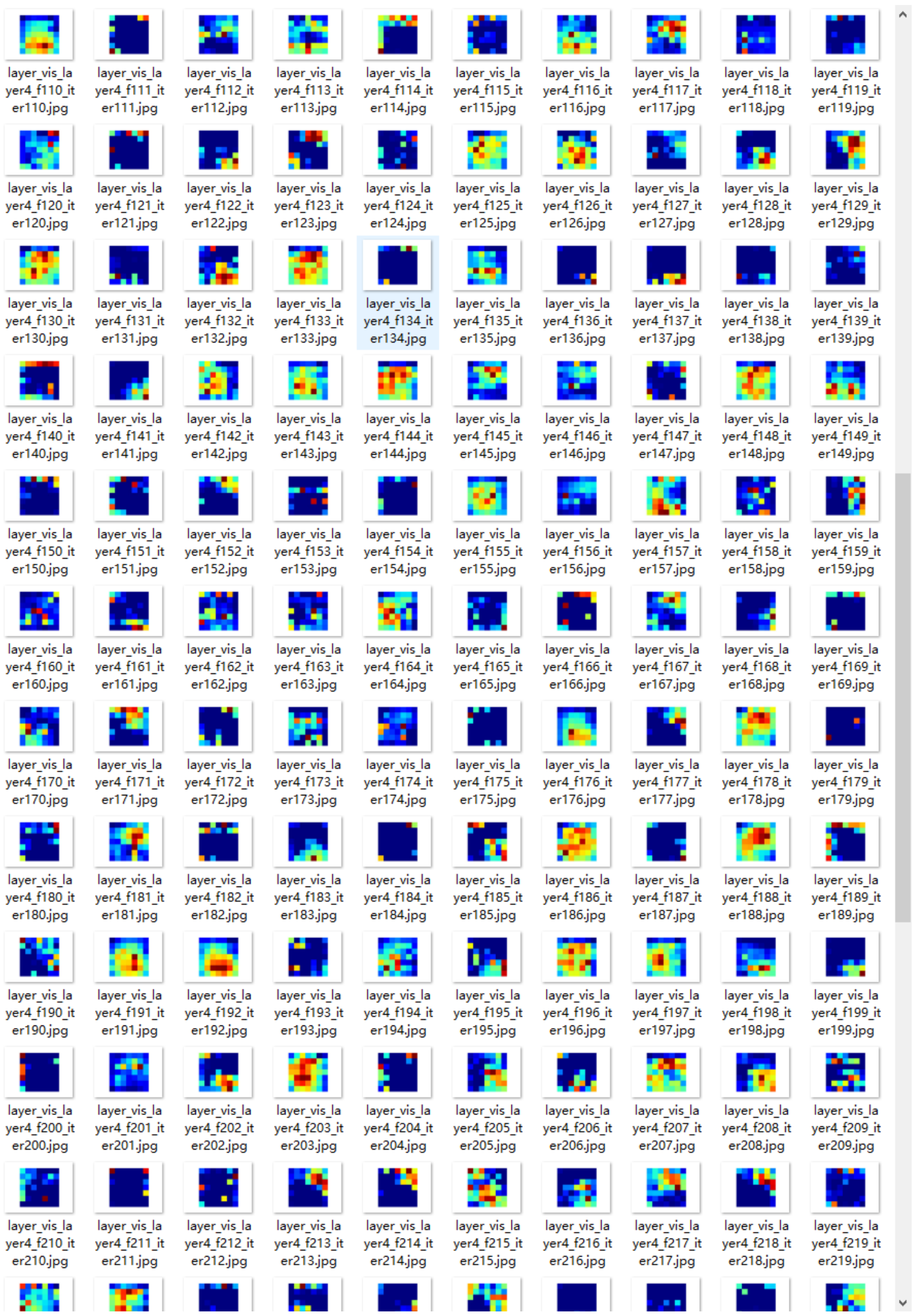

layer_vis_layer4_f110_iter110.jpg
layer_vis_layer4_f111_iter111.jpg
layer_vis_layer4_f112_iter112.jpg
layer_vis_layer4_f113_iter113.jpg
layer_vis_layer4_f114_iter114.jpg
layer_vis_layer4_f115_iter115.jpg
layer_vis_layer4_f116_iter116.jpg
layer_vis_layer4_f117_iter117.jpg
layer_vis_layer4_f118_iter118.jpg
layer_vis_layer4_f119_iter119.jpg
layer_vis_layer4_f120_iter120.jpg
layer_vis_layer4_f121_iter121.jpg
layer_vis_layer4_f122_iter122.jpg
layer_vis_layer4_f123_iter123.jpg
layer_vis_layer4_f124_iter124.jpg
layer_vis_layer4_f125_iter125.jpg
layer_vis_layer4_f126_iter126.jpg
layer_vis_layer4_f127_iter127.jpg
layer_vis_layer4_f128_iter128.jpg
layer_vis_layer4_f129_iter129.jpg
layer_vis_layer4_f130_iter130.jpg
layer_vis_layer4_f131_iter131.jpg
layer_vis_layer4_f132_iter132.jpg
layer_vis_layer4_f133_iter133.jpg
layer_vis_layer4_f134_iter134.jpg
layer_vis_layer4_f135_iter135.jpg
layer_vis_layer4_f136_iter136.jpg
layer_vis_layer4_f137_iter137.jpg
layer_vis_layer4_f138_iter138.jpg
layer_vis_layer4_f139_iter139.jpg
layer_vis_layer4_f140_iter140.jpg
layer_vis_layer4_f141_iter141.jpg
layer_vis_layer4_f142_iter142.jpg
layer_vis_layer4_f143_iter143.jpg
layer_vis_layer4_f144_iter144.jpg
layer_vis_layer4_f145_iter145.jpg
layer_vis_layer4_f146_iter146.jpg
layer_vis_layer4_f147_iter147.jpg
layer_vis_layer4_f148_iter148.jpg
layer_vis_layer4_f149_iter149.jpg
layer_vis_layer4_f150_iter150.jpg
layer_vis_layer4_f151_iter151.jpg
layer_vis_layer4_f152_iter152.jpg
layer_vis_layer4_f153_iter153.jpg
layer_vis_layer4_f154_iter154.jpg
layer_vis_layer4_f155_iter155.jpg
layer_vis_layer4_f156_iter156.jpg
layer_vis_layer4_f157_iter157.jpg
layer_vis_layer4_f158_iter158.jpg
layer_vis_layer4_f159_iter159.jpg
layer_vis_layer4_f160_iter160.jpg
layer_vis_layer4_f161_iter161.jpg
layer_vis_layer4_f162_iter162.jpg
layer_vis_layer4_f163_iter163.jpg
layer_vis_layer4_f164_iter164.jpg
layer_vis_layer4_f165_iter165.jpg
layer_vis_layer4_f166_iter166.jpg
layer_vis_layer4_f167_iter167.jpg
layer_vis_layer4_f168_iter168.jpg
layer_vis_layer4_f169_iter169.jpg
layer_vis_layer4_f170_iter170.jpg
layer_vis_layer4_f171_iter171.jpg
layer_vis_layer4_f172_iter172.jpg
layer_vis_layer4_f173_iter173.jpg
layer_vis_layer4_f174_iter174.jpg
layer_vis_layer4_f175_iter175.jpg
layer_vis_layer4_f176_iter176.jpg
layer_vis_layer4_f177_iter177.jpg
layer_vis_layer4_f178_iter178.jpg
layer_vis_layer4_f179_iter179.jpg
layer_vis_layer4_f180_iter180.jpg
layer_vis_layer4_f181_iter181.jpg
layer_vis_layer4_f182_iter182.jpg
layer_vis_layer4_f183_iter183.jpg
layer_vis_layer4_f184_iter184.jpg
layer_vis_layer4_f185_iter185.jpg
layer_vis_layer4_f186_iter186.jpg
layer_vis_layer4_f187_iter187.jpg
layer_vis_layer4_f188_iter188.jpg
layer_vis_layer4_f189_iter189.jpg
layer_vis_layer4_f190_iter190.jpg
layer_vis_layer4_f191_iter191.jpg
layer_vis_layer4_f192_iter192.jpg
layer_vis_layer4_f193_iter193.jpg
layer_vis_layer4_f194_iter194.jpg
layer_vis_layer4_f195_iter195.jpg
layer_vis_layer4_f196_iter196.jpg
layer_vis_layer4_f197_iter197.jpg
layer_vis_layer4_f198_iter198.jpg
layer_vis_layer4_f199_iter199.jpg
layer_vis_layer4_f200_iter200.jpg
layer_vis_layer4_f201_iter201.jpg
layer_vis_layer4_f202_iter202.jpg
layer_vis_layer4_f203_iter203.jpg
layer_vis_layer4_f204_iter204.jpg
layer_vis_layer4_f205_iter205.jpg
layer_vis_layer4_f206_iter206.jpg
layer_vis_layer4_f207_iter207.jpg
layer_vis_layer4_f208_iter208.jpg
layer_vis_layer4_f209_iter209.jpg
layer_vis_layer4_f210_iter210.jpg
layer_vis_layer4_f211_iter211.jpg
layer_vis_layer4_f212_iter212.jpg
layer_vis_layer4_f213_iter213.jpg
layer_vis_layer4_f214_iter214.jpg
layer_vis_layer4_f215_iter215.jpg
layer_vis_layer4_f216_iter216.jpg
layer_vis_layer4_f217_iter217.jpg
layer_vis_layer4_f218_iter218.jpg
layer_vis_layer4_f219_iter219.jpg

Fig 10(2). Feature maps of layer Conv5_x (7*7*512)

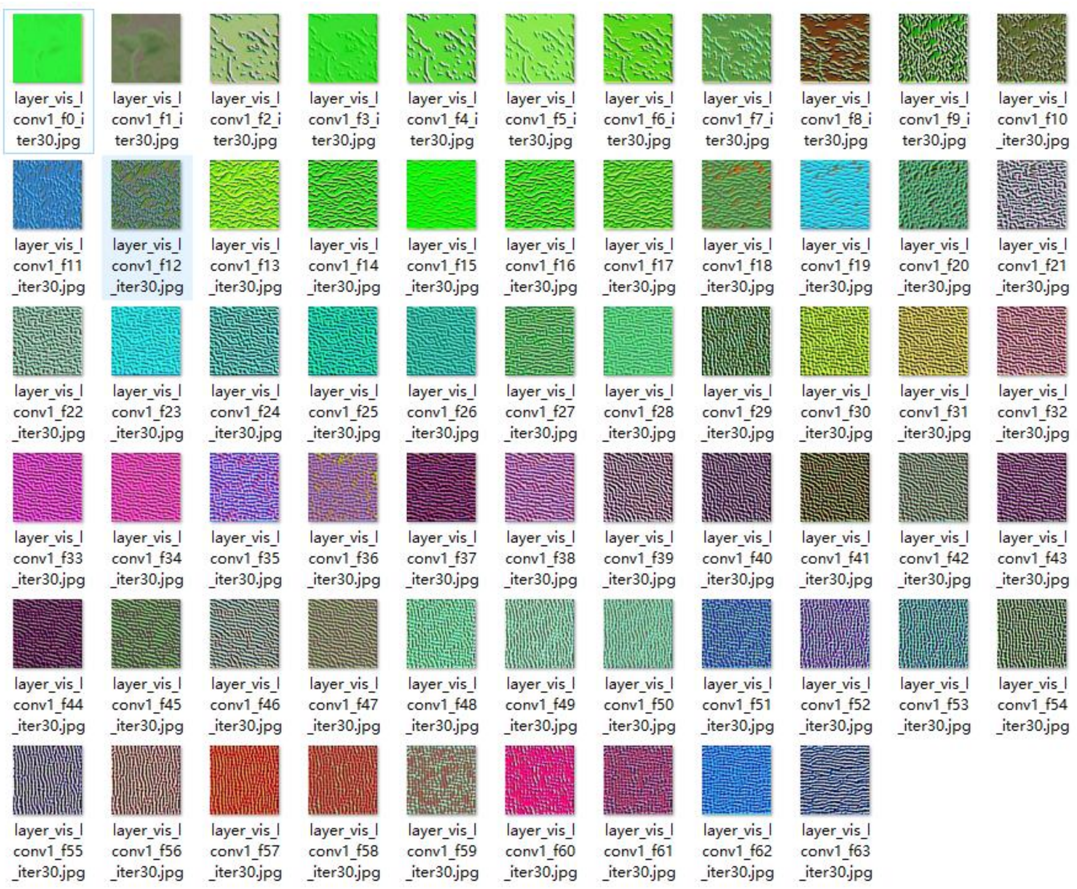

layer_vis_lconv1_f0_iter30.jpg
layer_vis_lconv1_f1_iter30.jpg
layer_vis_lconv1_f2_iter30.jpg
layer_vis_lconv1_f3_iter30.jpg
layer_vis_lconv1_f4_iter30.jpg
layer_vis_lconv1_f5_iter30.jpg
layer_vis_lconv1_f6_iter30.jpg
layer_vis_lconv1_f7_iter30.jpg
layer_vis_lconv1_f8_iter30.jpg
layer_vis_lconv1_f9_iter30.jpg
layer_vis_lconv1_f10_iter30.jpg
layer_vis_lconv1_f11_iter30.jpg
layer_vis_lconv1_f12_iter30.jpg
layer_vis_lconv1_f13_iter30.jpg
layer_vis_lconv1_f14_iter30.jpg
layer_vis_lconv1_f15_iter30.jpg
layer_vis_lconv1_f16_iter30.jpg
layer_vis_lconv1_f17_iter30.jpg
layer_vis_lconv1_f18_iter30.jpg
layer_vis_lconv1_f19_iter30.jpg
layer_vis_lconv1_f20_iter30.jpg
layer_vis_lconv1_f21_iter30.jpg
layer_vis_lconv1_f22_iter30.jpg
layer_vis_lconv1_f23_iter30.jpg
layer_vis_lconv1_f24_iter30.jpg
layer_vis_lconv1_f25_iter30.jpg
layer_vis_lconv1_f26_iter30.jpg
layer_vis_lconv1_f27_iter30.jpg
layer_vis_lconv1_f28_iter30.jpg
layer_vis_lconv1_f29_iter30.jpg
layer_vis_lconv1_f30_iter30.jpg
layer_vis_lconv1_f31_iter30.jpg
layer_vis_lconv1_f32_iter30.jpg
layer_vis_lconv1_f33_iter30.jpg
layer_vis_lconv1_f34_iter30.jpg
layer_vis_lconv1_f35_iter30.jpg
layer_vis_lconv1_f36_iter30.jpg
layer_vis_lconv1_f37_iter30.jpg
layer_vis_lconv1_f38_iter30.jpg
layer_vis_lconv1_f39_iter30.jpg
layer_vis_lconv1_f40_iter30.jpg
layer_vis_lconv1_f41_iter30.jpg
layer_vis_lconv1_f42_iter30.jpg
layer_vis_lconv1_f43_iter30.jpg
layer_vis_lconv1_f44_iter30.jpg
layer_vis_lconv1_f45_iter30.jpg
layer_vis_lconv1_f46_iter30.jpg
layer_vis_lconv1_f47_iter30.jpg
layer_vis_lconv1_f48_iter30.jpg
layer_vis_lconv1_f49_iter30.jpg
layer_vis_lconv1_f50_iter30.jpg
layer_vis_lconv1_f51_iter30.jpg
layer_vis_lconv1_f52_iter30.jpg
layer_vis_lconv1_f53_iter30.jpg
layer_vis_lconv1_f54_iter30.jpg
layer_vis_lconv1_f55_iter30.jpg
layer_vis_lconv1_f56_iter30.jpg
layer_vis_lconv1_f57_iter30.jpg
layer_vis_lconv1_f58_iter30.jpg
layer_vis_lconv1_f59_iter30.jpg
layer_vis_lconv1_f60_iter30.jpg
layer_vis_lconv1_f61_iter30.jpg
layer_vis_lconv1_f62_iter30.jpg
layer_vis_lconv1_f63_iter30.jpg

Fig 11. The reconstructed patterns from the validation set

Feedback (1p)

1) Time you spend for this assignment, i.e., how many hours?

   Half of a month, nearly 30 hours.

2) Comments for this course?

   Useful, meaningful, and interesting.

   However, it is more like a computer vision class. I think Artificial Intelligence should not only contain CV. It has many other fields such as NLP, Robotics, Autonomous Vehicles. It will be better if we can learning things about such filed from this course.

3) Comments for this assignment?

   Helpful. I learn some useful skills from this assignment that will do good help to my further research work.

And the most important is it gives us a chance to combat, not only stay in the book, we can learn how to build a model from scratch, how to adjust the parameter, how to use tensorboardX to get an accurate picture, how to visualize the feature map.

All this experience is meaningful.

4) Suggestion for the following lectures?

I think this lecture focus too much on the computer vision field. As we all know, artificial intelligence is not only for CV, it has many subfields. And for me, I major in Natural Language Processing, so I would like to know more about NLP pioneer, such as Transformer, XL-Net, etc.

To sum up, I think it should be an Introduction Class for us to enter the field of Artificial Intelligence, containing a wide and comprehensive field of AI, just like lecture on Computer Introduction for a computer newbie.

Reference:

[1] https://arxiv.org/abs/1512.03385

[2] https://blog.csdn.net/sunqiande88/article/details/80100891

[3] https://github.com/utkuozbulak/pytorch-cnn-visualizations

[4] https://github.com/kvfrans/feature-visualization

[5] http://kvfrans.com/visualizing-features-from-a-convolutional-neural-network/