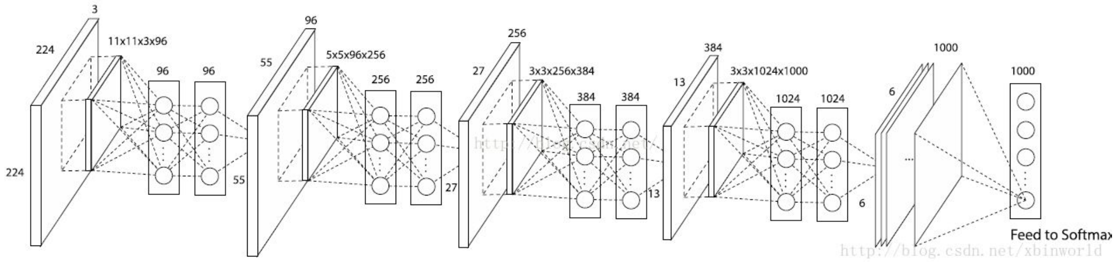
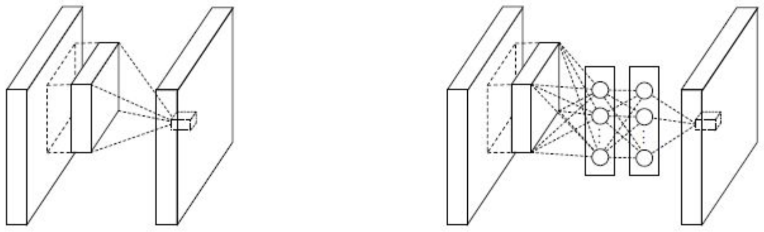
1.NIN network



2.Mlpconv layer

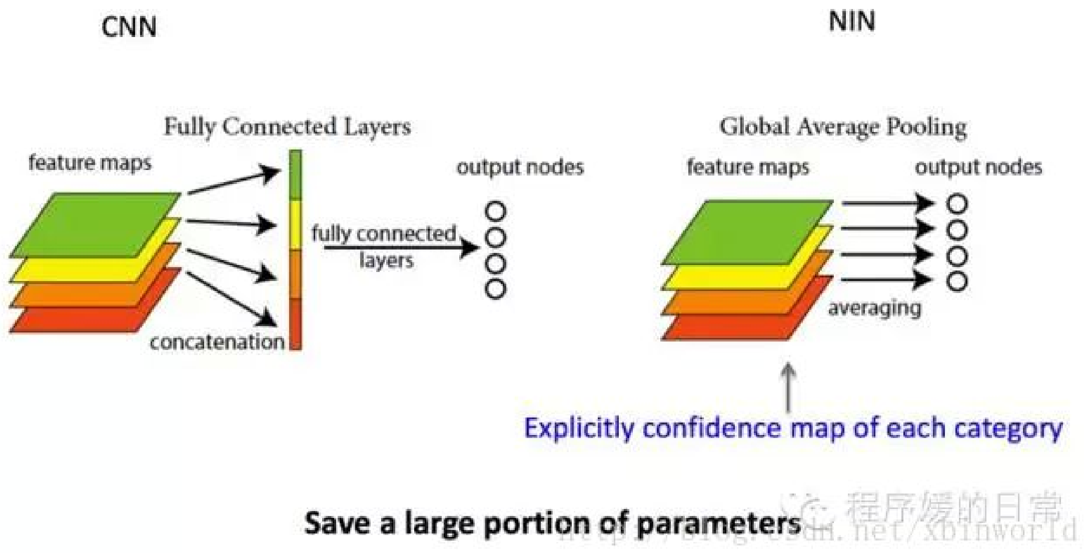


(a) Linear convolution layer

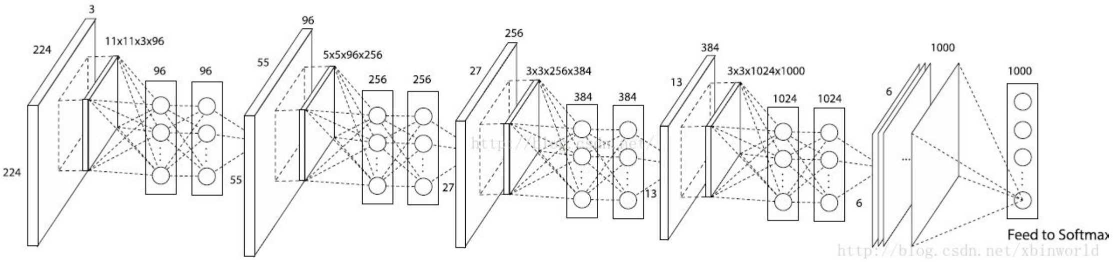
(b) Mlpconv layer

Figure 1: Comparison of linear convolution layer and mlpconv layer. The linear convolution layer includes a linear filter while the mlpconv layer includes a micro network (we choose the multilayer perceptron in this paper). Both layers map the local receptive field to a confidence value of the latent concept.

3.CNN vs NIN



4.NIN network code review



class torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True)

Parameters:

in_channels (int) – Number of channels in the input image
out_channels (int) – Number of channels produced by the convolution
kernel_size (intortuple) – Size of the convolving kernel
stride (intortuple,optional) – Stride of the convolution. Default: 1
padding (intortuple,optional) – Zero-padding added to both sides of the input. Default: 0
dilation (intortuple,optional) – Spacing between kernel elements. Default: 1
groups (int,optional) – Number of blocked connections from input channels to output channels. Default: 1
bias (bool,optional) – If True, adds a learnable bias to the output. Default: True

```
//############################################################
        nn.Conv2d(3,96,(11, 11),(4, 4)),
            //去掉輸入小於0的部分
            nn.ReLU(inplace=True),

            //1*1 convolution
            nn.Conv2d(96,96,(1, 1)),
            nn.ReLU(inplace=True),

            nn.Conv2d(96,96,(1, 1)),
            nn.ReLU(inplace=True),
          pool2d,
//############################################################
            nn.Conv2d(96,256,(5, 5),(1, 1),(2, 2)),
            nn.ReLU(inplace=True),
            nn.Conv2d(256,256,(1, 1)),
            nn.ReLU(inplace=True),
            nn.Conv2d(256,256,(1, 1)),
            nn.ReLU(inplace=True),
          pool2d,
//############################################################
            nn.Conv2d(256,384,(3, 3),(1, 1),(1, 1)),
            nn.ReLU(inplace=True),
            nn.Conv2d(384,384,(1, 1)),
            nn.ReLU(inplace=True),
            nn.Conv2d(384,384,(1, 1)),
            nn.ReLU(inplace=True),
        pool2d,
//##########################################################
            //Global Average Pooling

            //減少過擬和
            nn.Dropout(0.5),
            nn.Conv2d(384,1024,(3, 3),(1, 1),(1, 1)),
            nn.ReLU(inplace=True),
            nn.Conv2d(1024,1024,(1, 1)),
            nn.ReLU(inplace=True),
            nn.Conv2d(1024,1000,(1, 1)),
            nn.ReLU(inplace=True),
            nn.AvgPool2d((6, 6),(1, 1),(0, 0),ceil_mode=True),
            //選取機率最大的節點
            nn.Softmax()
```

在pytorch-nin執行python NIN.py

5. 請到下面網址下載並且將資料放到pytorch-nin-cifar10的data資料夾
download link

在pytorch-nin-cifar10執行python original.py, 部份結果在record.txt

```
Train Epoch: 1 [0/50000 (0%)]   Loss: 2.302401  LR: 0.2
Train Epoch: 1 [12800/50000 (26%)]      Loss: 2.278484  LR: 0.2
Train Epoch: 1 [25600/50000 (51%)]      Loss: 2.232099  LR: 0.2
Train Epoch: 1 [38400/50000 (77%)]      Loss: 2.237220  LR: 0.2

Test set: Average loss: 2.7008, Accuracy: 2161/10000 (21.00%)


Train Epoch: 2 [0/50000 (0%)]   Loss: 2.149667  LR: 0.2
Train Epoch: 2 [12800/50000 (26%)]      Loss: 2.292702  LR: 0.2
Train Epoch: 2 [25600/50000 (51%)]      Loss: 2.314060  LR: 0.2
Train Epoch: 2 [38400/50000 (77%)]      Loss: 2.294102  LR: 0.2

Test set: Average loss: 2.8744, Accuracy: 1390/10000 (13.00%)

Train Epoch: 3 [0/50000 (0%)]   Loss: 2.260334  LR: 0.2
.
```

```
.
.
Train Epoch: 130 [0/50000 (0%)] Loss: 0.026027  LR: 0.020000000000000004
Train Epoch: 130 [12800/50000 (26%)]    Loss: 0.072957  LR:
0.020000000000000004
Train Epoch: 130 [25600/50000 (51%)]    Loss: 0.040447  LR:
0.020000000000000004
Train Epoch: 130 [38400/50000 (77%)]    Loss: 0.046105  LR:
0.020000000000000004


Train Epoch: 131 [0/50000 (0%)] Loss: 0.093213  LR: 0.020000000000000004
Train Epoch: 131 [12800/50000 (26%)]    Loss: 0.011945  LR:
0.020000000000000004
Train Epoch: 131 [25600/50000 (51%)]    Loss: 0.020603  LR:
0.020000000000000004
```