Week 3 Report

Weitian Li weitian.li@rutgers.edu

1. Another version of XNOR-Net (With CUDA code in C++)

完成情况：对该 github 项目进行 debug 并跑通在 Cifar10 上的结果，相关结果将在第三个任务里展示。

问题：

1. Pytorch 版本问题。
2. Python 版本问题。
3. binop 编译问题。
4. util 包 pytorch 语句更新问题，191，192 行。

```
self.target_modules[index].data.sign().mul(
                    self.alpha_to_save[index].data.expand(s),
out=self.target_modules[index].data)
```

5. __init__问题，已经在 issue 反映。

2. Backpropagation of quantization neural network

PDF

XNOR-Net: ImageNet Classification Using Binary
Convolutional Neural Networks

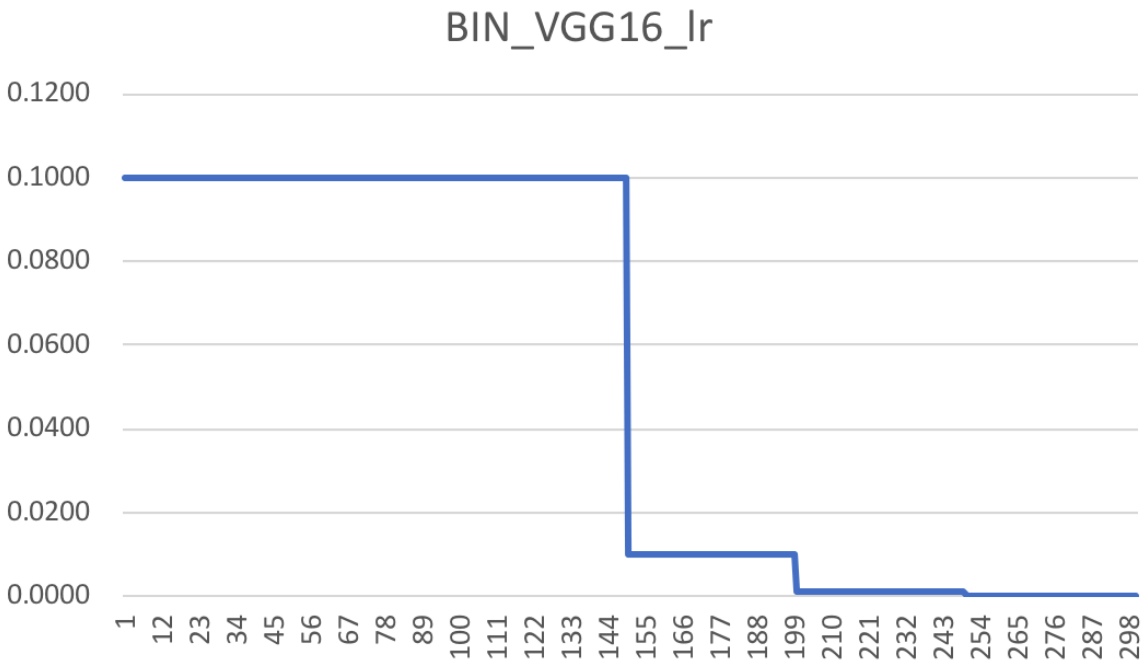
Binarized Neural Networks: Training Neural Networks with Weights and
Activations Constrained to +1 or −1

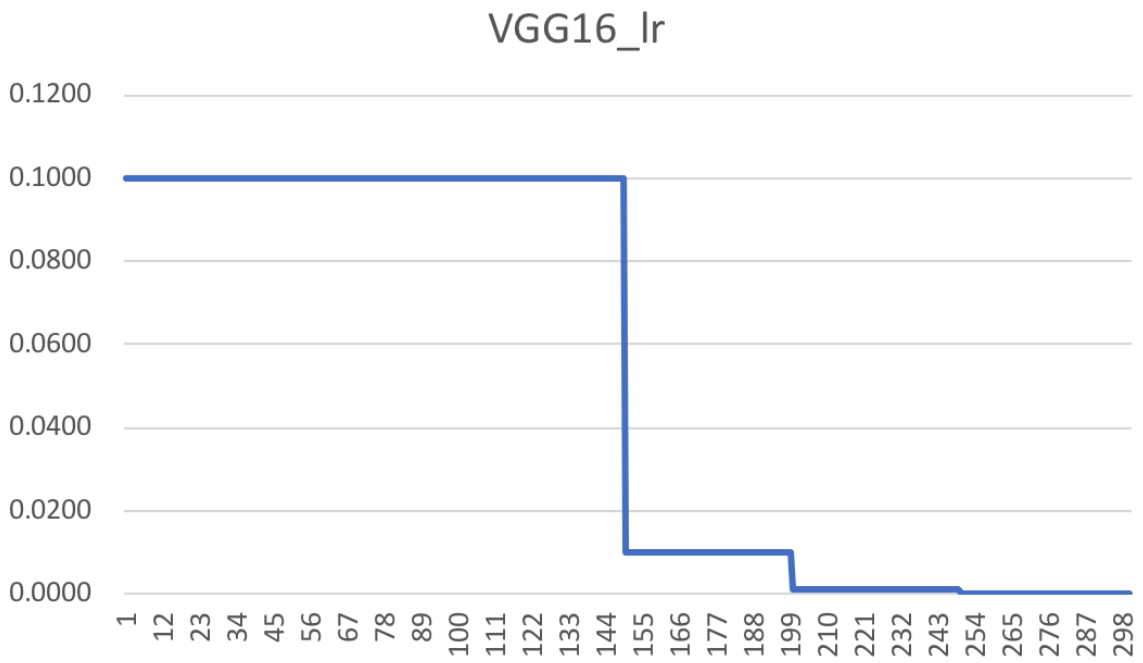
完成情况：大概理解数学的公式和回传过程，pytorch 代码理解中。

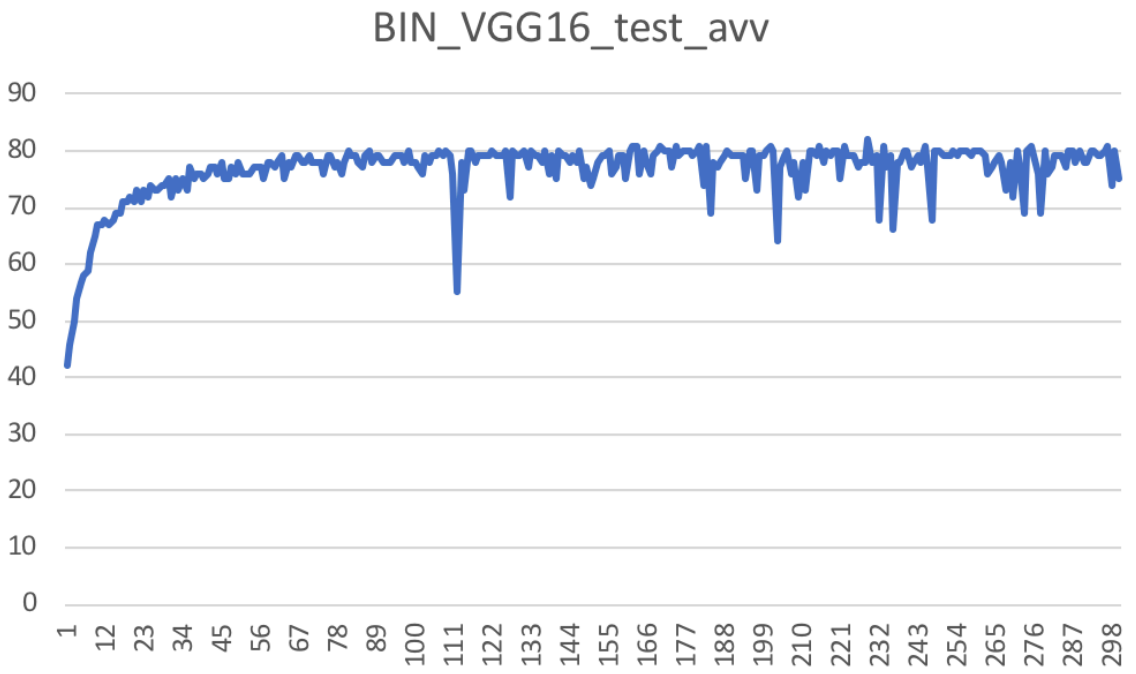
问题：部分 pytorch 代码理解不够明确。

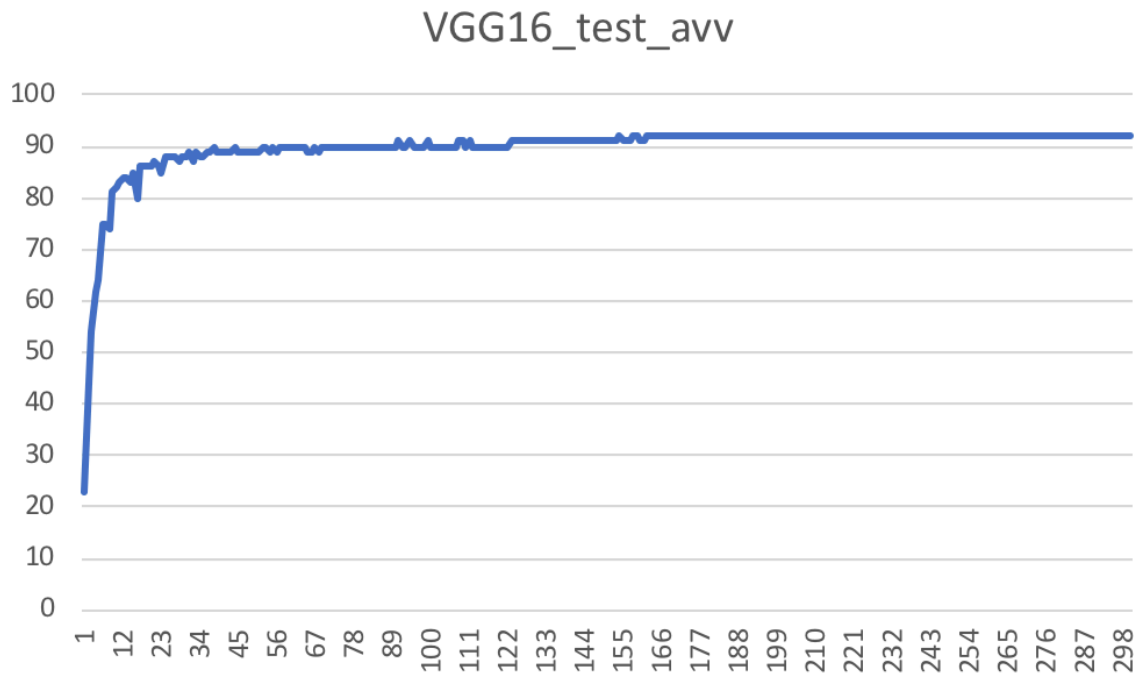
3. Report learning curve of CIFAR-10 of XNOR-Net

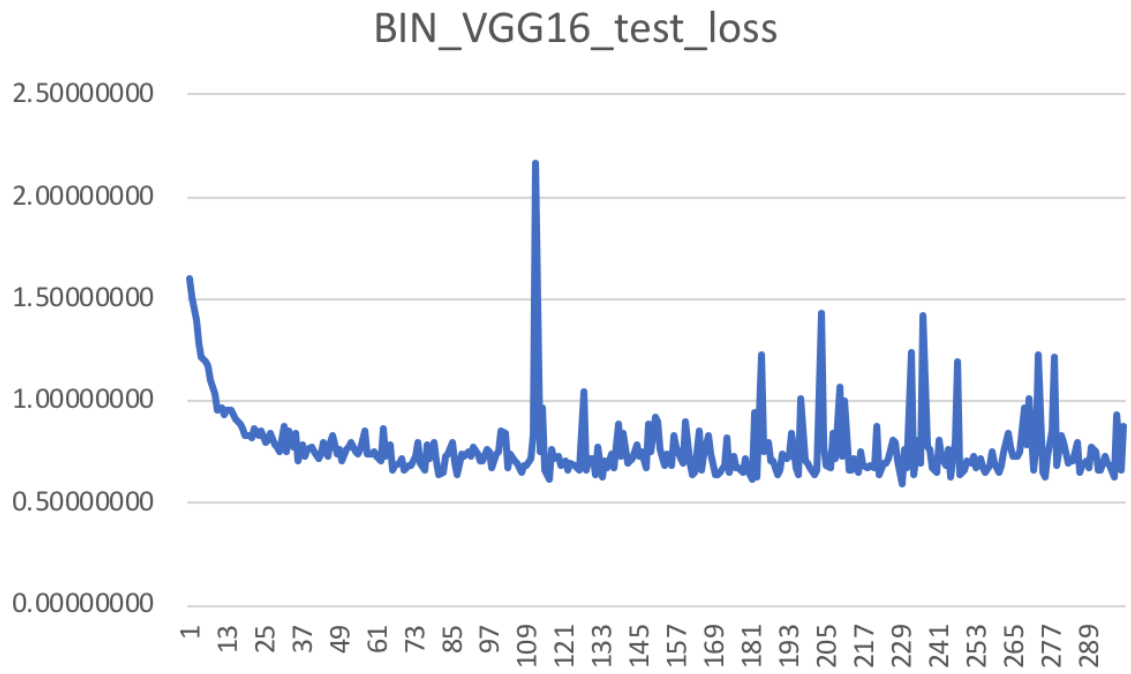
完成情况：完成第一个任务的结果，BIN_VGG16 和 VGG16 的运行和 evaluate。

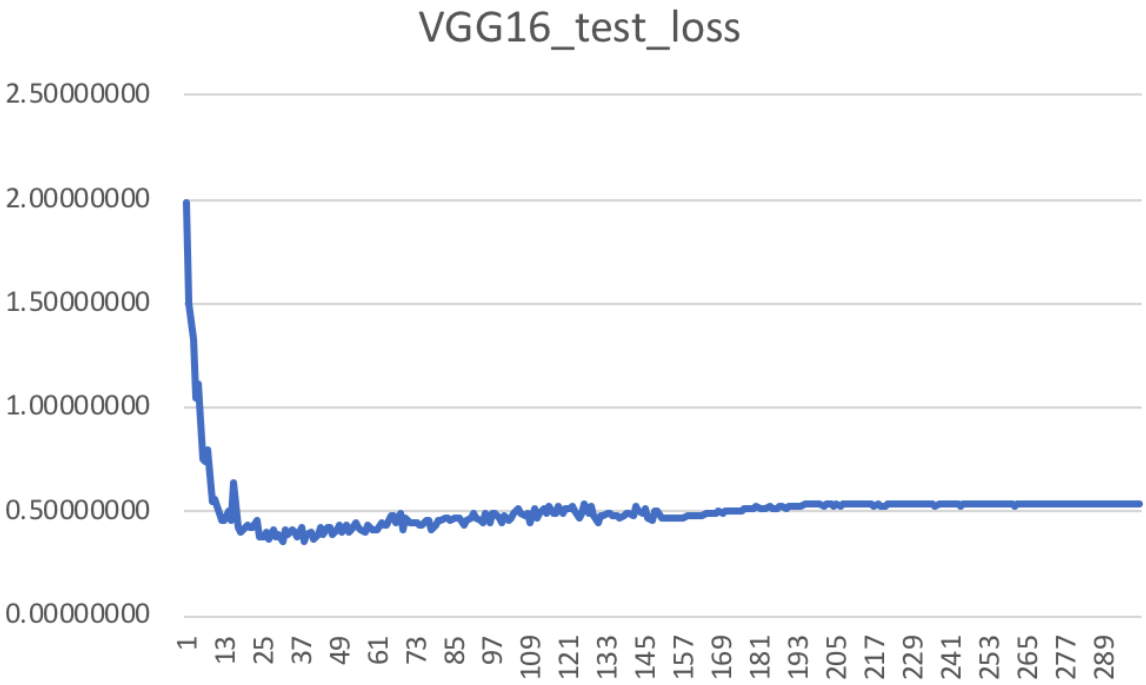

BIN_VGG16_lr
0.1200
0.1000
0.0800
0.0600
0.0400
0.0200
0.0000
1
12
23
34
45
56
67
78
89
100
111
122
133
144
155
166
177
188
199
210
221
232
243
254
265
276
287
298

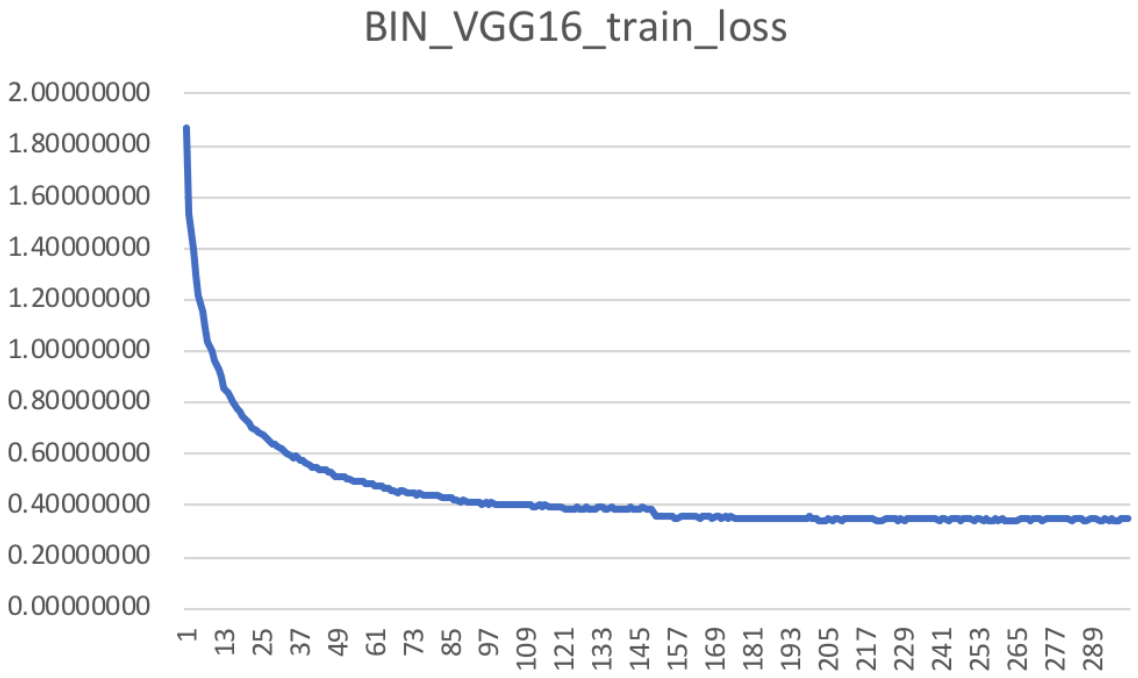

VGG16_lr
0.1200
0.1000
0.0800
0.0600
0.0400
0.0200
0.0000
1
12
23
34
45
56
67
78
89
100
111
122
133
144
155
166
177
188
199
210
221
232
243
254
265
276
287
298

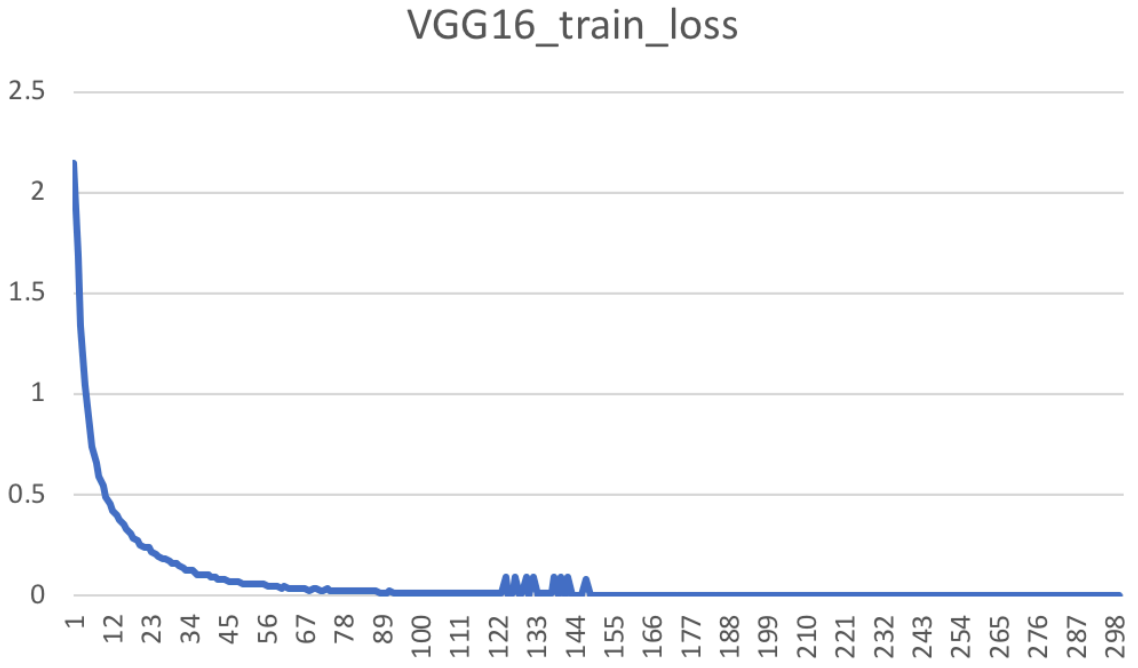

BIN_VGG16_test_avv
90
80
70
60
50
40
30
20
10
0
1
12
23
34
45
56
67
78
89
100
111
122
133
144
155
166
177
188
199
210
221
232
243
254
265
276
287
298


VGG16_test_avv
100
90
80
70
60
50
40
30
20
10
0
1
12
23
34
45
56
67
78
89
100
111
122
133
144
155
166
177
188
199
210
221
232
243
254
265
276
287
298


BIN_VGG16_test_loss
2.50000000
2.00000000
1.50000000
1.00000000
0.50000000
0.00000000
1
13
25
37
49
61
73
85
97
109
121
133
145
157
169
181
193
205
217
229
241
253
265
277
289


VGG16_test_loss
2.50000000
2.00000000
1.50000000
1.00000000
0.50000000
0.00000000
1
13
25
37
49
61
73
85
97
109
121
133
145
157
169
181
193
205
217
229
241
253
265
277
289


BIN_VGG16_train_loss
2.00000000
1.80000000
1.60000000
1.40000000
1.20000000
1.00000000
0.80000000
0.60000000
0.40000000
0.20000000
0.00000000
1
13
25
37
49
61
73
85
97
109
121
133
145
157
169
181
193
205
217
229
241
253
265
277
289


VGG16_train_loss
2.5
2
1.5
1
0.5
0
1
12
23
34
45
56
67
78
89
100
111
122
133
144
155
166
177
188
199
210
221
232
243
254
265
276
287
298

pretrain_VGG16:

Test set: Average loss: 0.4672, Accuracy: 9206/10000 (92.00%)

```
Confusion matrix
[[946   5  14   6   2   0   0   2  14  11]
 [  6 965   0   1   0   0   0   0   3  25]
 [ 26   1 885  20  25  13  19   5   6   0]
 [ 11   2  21 827  21  70  24  13   6   5]
 [  5   2  15  16 926  11   6  17   1   1]
 [  3   0  14  76  17 871   7  10   2   0]
 [  8   0  17  21   7   4 939   1   3   0]
 [  7   0   8   9  11  19   0 943   1   2]
 [ 26   6   3   1   0   1   1   1 956   5]
 [  8  27   2   3   0   0   1   1  10 948]]

Normalized confusion matrix
[[0.946 0.005 0.014 0.006 0.002 0.    0.    0.002 0.014 0.011]
 [0.006 0.965 0.    0.001 0.    0.    0.    0.    0.003 0.025]
 [0.026 0.001 0.885 0.02  0.025 0.013 0.019 0.005 0.006 0.   ]
 [0.011 0.002 0.021 0.827 0.021 0.07  0.024 0.013 0.006 0.005]
 [0.005 0.002 0.015 0.016 0.926 0.011 0.006 0.017 0.001 0.001]
 [0.003 0.    0.014 0.076 0.017 0.871 0.007 0.01  0.002 0.   ]
 [0.008 0.    0.017 0.021 0.007 0.004 0.939 0.001 0.003 0.   ]
 [0.007 0.    0.008 0.009 0.011 0.019 0.    0.943 0.001 0.002]
 [0.026 0.006 0.003 0.001 0.    0.001 0.001 0.001 0.956 0.005]
 [0.008 0.027 0.002 0.003 0.    0.    0.001 0.001 0.01  0.948]]
             precision    recall  f1-score   support

      plane   0.904398  0.946000  0.924731      1000
        car   0.957341  0.965000  0.961155      1000
       bird   0.903984  0.885000  0.894391      1000
        cat   0.843878  0.827000  0.835354      1000
       deer   0.917740  0.926000  0.921852      1000
        dog   0.880688  0.871000  0.875817      1000
       frog   0.941825  0.939000  0.940411      1000
      horse   0.949648  0.943000  0.946312      1000
       ship   0.954092  0.956000  0.955045      1000
      truck   0.950853  0.948000  0.949424      1000

avg / total   0.920445  0.920600  0.920449     10000
```

pretrainBin_VGG16:

Test set: Average loss: 0.5929, Accuracy: 8203/10000 (82.00%)

```
Confusion matrix
[[778  26  76   6   6   2  13  13  50  30]
 [  8 965   2   2   0   0   1   1   6  15]
 [ 24   7 812  21  20  17  69  14   4  12]
 [ 10   7  74 612  24 131  99  22   7  14]
 [  8   3  94  28 722  21  69  44   8   3]
 [  4   5  48 108  16 739  31  45   0   4]
 [  3   6  24  29   6   8 921   1   2   0]
 [  6   7  17  18  12  28   9 893   2   8]
 [ 34  33  10   4   0   2   5   2 899  11]
 [  4 107   2   2   0   1   3   4  15 862]]
```

```
Normalized confusion matrix
[[0.778 0.026 0.076 0.006 0.006 0.002 0.013 0.013 0.05  0.03 ]
 [0.008 0.965 0.002 0.002 0.    0.    0.001 0.001 0.006 0.015]
 [0.024 0.007 0.812 0.021 0.02  0.017 0.069 0.014 0.004 0.012]
 [0.01  0.007 0.074 0.612 0.024 0.131 0.099 0.022 0.007 0.014]
 [0.008 0.003 0.094 0.028 0.722 0.021 0.069 0.044 0.008 0.003]
 [0.004 0.005 0.048 0.108 0.016 0.739 0.031 0.045 0.    0.004]
 [0.003 0.006 0.024 0.029 0.006 0.008 0.921 0.001 0.002 0.   ]
 [0.006 0.007 0.017 0.018 0.012 0.028 0.009 0.893 0.002 0.008]
 [0.034 0.033 0.01  0.004 0.    0.002 0.005 0.002 0.899 0.011]
 [0.004 0.107 0.002 0.002 0.    0.001 0.003 0.004 0.015 0.862]]
```

	precision	recall	f1-score	support
plane	0.885097	0.778000	0.828100	1000
car	0.827616	0.965000	0.891043	1000
bird	0.700604	0.812000	0.752200	1000
cat	0.737349	0.612000	0.668852	1000
deer	0.895782	0.722000	0.799557	1000
dog	0.778714	0.739000	0.758338	1000
frog	0.754918	0.921000	0.829730	1000
horse	0.859480	0.893000	0.875920	1000
ship	0.905337	0.899000	0.902158	1000
truck	0.898853	0.862000	0.880041	1000
avg / total	0.824375	0.820300	0.818594	10000

问题：准确率达不到 github 所说的情况，代码和版本要求比较高，bug 不少。

BP — XNOR-NET

Training:

BNN (-1, 1)

conv: Input * Weight (conv kernel)

Input *(B(conv kernel) ⊕ I) α (scaled function)

优化: W≈αB, 尽量达到.

使用优化函数 $J(B,\alpha)=\|W-\alpha B\|^2$

$B_{(BEST)}=\arg\max\{W^TB\}$, $B\in\{+1,-1\}^n$

$\alpha_{(BEST)}=\frac{1}{n}\|W_{l1}\|$ (Mean Center).

Forward: Actinam (Sigmoid, ReLU)

Prop.

① $\widetilde{Weight}=Sign(Weight)$, $Input=\sum\widetilde{W}x_i$

1) $a_{ik}=f(Input)$

2) $q_k=\widehat{Batch\ norm}(a_{ik})$

3) $\widetilde{a_k}=Sign(q_k)$

Back:

Prop.

$q=Sign(r)$

Gradient for bincy weight

$g_r=g_q|_{|r|\le 1}$

for $|r|\le 1$, $H\tanh(x)=Clip(x,-1,1)=\max(-1,\min(1,x))$