

# DeepProbLog

May 5, 2021

## 1 DeepProbLog

DeepProbLog is an extension of ProbLog that integrates Probabilistic Logic Programming with Deep Learning.

The git repo of the project can be downloaded by:

git clone <https://bitbucket.org/problog/deepproblog.git>

### 1.1 Example: MNIST Digit Addition

In this experiment, the task is to classify the sum of two lists of MNIST digits representing multi-digit numbers.

First, we create a ProbLog file containing the logic part of the program. The file will be saved as *tutorial/multi\_digit.pl*.

```
nn(mnist_net,[X],Y,[0,1,2,3,4,5,6,7,8,9]) :: digit(X,Y).

number([],Result,Result).
number([H|T],Acc,Result) :- digit(H,Nr),
                             Acc2 is Nr+10*Acc,
                             number(T,Acc2,Result).
number(X,Y) :- number(X,0,Y).

addition(X,Y,Z) :- number(X,X2), number(Y,Y2), Z is X2+Y2.
```

Then, we load the MNIST dataset, the queries and the ProbLog file.

```
[1]: from torchvision.datasets import MNIST
import torchvision.transforms as transforms
from data_loader import load

transform = transforms.Compose([transforms.ToTensor(), transforms.Normalize((0.
    ↪5, ), (0.5, ))])
mnist_train_data = MNIST(root='data/MNIST', train=True, ↵
    ↪download=True, transform=transform)
mnist_test_data = MNIST(root='data/MNIST', train=False, ↵
    ↪download=True, transform=transform)
```

```

train_queries = load('tutorial/train.txt')
test_queries = load('tutorial/test.txt')[:100]

with open('tutorial/multi_digit.pl') as f:
    problog_string = f.read()

```

Train and test queries look like this:

```

addition([train(2764)], [train(8527)], 9).
addition([train(27012)], [train(56713)], 10).
...

addition([test(9271),test(5812),test(9788)], [test(4522),test(8572),test(3555)], 1575).
addition([test(4052),test(7966),test(5512)], [test(4884),test(5655),test(133)], 1554).
...

```

We can now define a python class implementing a standard CNN for MNIST images, and a neural predicate connecting the image id (as found in the query) to the corresponding image and the sending it to the neural net.

```

[2]: import torch
import torch.nn as nn
from torch.autograd import Variable

class MNIST_Net(nn.Module):
    def __init__(self, N=10):
        super(MNIST_Net, self).__init__()
        self.encoder = nn.Sequential(
            nn.Conv2d(1, 6, 5),
            nn.MaxPool2d(2, 2), # 6 24 24 -> 6 12 12
            nn.ReLU(True),
            nn.Conv2d(6, 16, 5), # 6 12 12 -> 16 8 8
            nn.MaxPool2d(2, 2), # 16 8 8 -> 16 4 4
            nn.ReLU(True)
        )
        self.classifier = nn.Sequential(
            nn.Linear(16 * 4 * 4, 120),
            nn.ReLU(),
            nn.Linear(120, 84),
            nn.ReLU(),
            nn.Linear(84, N),
            nn.Softmax(1)
        )

    def forward(self, x):
        x = self.encoder(x)
        x = x.view(-1, 16 * 4 * 4)

```

```

        x = self.classifier(x)
        return x

def neural_predicate(network, i):
    # i is something like train(2764) or test(4052)
    dataset = str(i.functor)
    i = int(i.args[0])
    if dataset == 'train':
        d, l = mnist_train_data[i]
    elif dataset == 'test':
        d, l = mnist_test_data[i]
    d = Variable(d.unsqueeze(0))
    output = network.net(d)
    return output.squeeze(0)

```

Finally, we can create the network and the DeepProbLog model with the network as neural predicate, and train it as a standard torch model.

```

[3]: from train import train_model
    from network import Network
    from model import Model
    from optimizer import Optimizer

    def test(model):
        acc = model.accuracy(test_queries, test=True)
        print('Accuracy: ', acc)
        return [('accuracy', acc)]

    network = MNIST_Net()
    net = Network(network, 'mnist_net', neural_predicate)
    net.optimizer = torch.optim.Adam(network.parameters(), lr=0.001)
    model = Model(problog_string, [net], caching=False)
    optimizer = Optimizer(model, 2)

    train_model(model, train_queries, 1, optimizer, test_iter=1000, test=test,
        ↪ snapshot_iter=10000)

```

Training for 1 epochs (30000 iterations).

/anaconda/lib/python3.6/site-packages/torch/nn/modules/module.py:795:

UserWarning: Using a non-full backward hook when the forward contains multiple autograd Nodes is deprecated and will be removed in future versions. This hook will be missing some grad\_input. Please use register\_full\_backward\_hook to get the documented behavior.

warnings.warn("Using a non-full backward hook when the forward contains multiple autograd Nodes ")

Accuracy 0.0

Accuracy: [('Accuracy', 0.0)]

Epoch 1

Iteration:	100	Average Loss:	2.792309776149683
Iteration:	200	Average Loss:	2.794529172274774
Iteration:	300	Average Loss:	2.7226911304399373
Iteration:	400	Average Loss:	2.6777938620027495
Iteration:	500	Average Loss:	2.4212049471079284
Iteration:	600	Average Loss:	2.1814432368345726
Iteration:	700	Average Loss:	2.34649068013998
Iteration:	800	Average Loss:	2.070823437229237
Iteration:	900	Average Loss:	2.0759107402896295
Iteration:	1000	Average Loss:	1.9284236394877732

Accuracy 0.02

Accuracy: [('Accuracy', 0.02)]

Iteration:	1100	Average Loss:	1.6618416189795162
Iteration:	1200	Average Loss:	1.48477843574703
Iteration:	1300	Average Loss:	1.3135947585705268
Iteration:	1400	Average Loss:	1.1531996091992525
Iteration:	1500	Average Loss:	0.8075905448534274
Iteration:	1600	Average Loss:	0.8614069331663536
Iteration:	1700	Average Loss:	0.6954511971524876
Iteration:	1800	Average Loss:	0.5237280982990555
Iteration:	1900	Average Loss:	0.821969847210108
Iteration:	2000	Average Loss:	0.6616524171030629

Accuracy 0.58

Accuracy: [('Accuracy', 0.58)]

Iteration:	2100	Average Loss:	0.4877591337200873
Iteration:	2200	Average Loss:	0.6078675338812183
Iteration:	2300	Average Loss:	0.9050086610832453
Iteration:	2400	Average Loss:	0.3722563267002236
Iteration:	2500	Average Loss:	0.3099381339045193
Iteration:	2600	Average Loss:	0.4569035952716875
Iteration:	2700	Average Loss:	0.3865900036989023
Iteration:	2800	Average Loss:	0.4097022564223662
Iteration:	2900	Average Loss:	0.4327400090030068
Iteration:	3000	Average Loss:	0.3245186722270845

Accuracy 0.65

Accuracy: [('Accuracy', 0.65)]

Iteration:	3100	Average Loss:	0.27101991493155486
Iteration:	3200	Average Loss:	0.29050574552002023
Iteration:	3300	Average Loss:	0.49906900021059497
Iteration:	3400	Average Loss:	0.5097507986470833
Iteration:	3500	Average Loss:	0.49093142113174737
Iteration:	3600	Average Loss:	0.3938806028452349
Iteration:	3700	Average Loss:	0.3744497721601713
Iteration:	3800	Average Loss:	0.3099438781994838
Iteration:	3900	Average Loss:	0.3679988305076168
Iteration:	4000	Average Loss:	0.2793371284339611

Accuracy 0.67

```

Accuracy: [('Accuracy', 0.67)]
Iteration: 4100      Average Loss: 0.5442097922412641
Iteration: 4200      Average Loss: 0.44123931032597047
Iteration: 4300      Average Loss: 0.35903992639743393
Iteration: 4400      Average Loss: 0.3842048669367069
Iteration: 4500      Average Loss: 0.20033821824474476
Iteration: 4600      Average Loss: 0.561075089441936
Iteration: 4700      Average Loss: 0.23276036867191988
Iteration: 4800      Average Loss: 0.23075578528784269
Iteration: 4900      Average Loss: 0.21539172391205114
Iteration: 5000      Average Loss: 0.23355511917334063
Accuracy 0.71
Accuracy: [('Accuracy', 0.71)]
Iteration: 5100      Average Loss: 0.45895553853722043
Iteration: 5200      Average Loss: 0.305295529071662
Iteration: 5300      Average Loss: 0.23493660878174688
Iteration: 5400      Average Loss: 0.24031581394442475
Iteration: 5500      Average Loss: 0.13677654412048887
Iteration: 5600      Average Loss: 0.22199253078989417
Iteration: 5700      Average Loss: 0.17501996109703755
Iteration: 5800      Average Loss: 0.21355945980017865
Iteration: 5900      Average Loss: 0.3968203104373756
Iteration: 6000      Average Loss: 0.4259022608059283
Accuracy 0.75
Accuracy: [('Accuracy', 0.75)]
Iteration: 6100      Average Loss: 0.24187494270563584
Iteration: 6200      Average Loss: 0.16887874279327889
Iteration: 6300      Average Loss: 0.15338433782648755
Iteration: 6400      Average Loss: 0.1743943822873106
Iteration: 6500      Average Loss: 0.19940258851616968
Iteration: 6600      Average Loss: 0.23047905197942298
Iteration: 6700      Average Loss: 0.22558434555640916
Iteration: 6800      Average Loss: 0.2634678309336145
Iteration: 6900      Average Loss: 0.12072512142179548
Iteration: 7000      Average Loss: 0.28975729237286485
Accuracy 0.74
Accuracy: [('Accuracy', 0.74)]
Iteration: 7100      Average Loss: 0.10399908078788445
Iteration: 7200      Average Loss: 0.20089343073984103
Iteration: 7300      Average Loss: 0.25399636924625935
Iteration: 7400      Average Loss: 0.3009463680785037
Iteration: 7500      Average Loss: 0.25675238039784815
Iteration: 7600      Average Loss: 0.20212818109445282
Iteration: 7700      Average Loss: 0.27918971824645256
Iteration: 7800      Average Loss: 0.350630829195142
Iteration: 7900      Average Loss: 0.16635465335817276
Iteration: 8000      Average Loss: 0.36888368196513655
Accuracy 0.81

```

```

Accuracy: [('Accuracy', 0.81)]
Iteration: 8100      Average Loss: 0.2830018876663354
Iteration: 8200      Average Loss: 0.31087280535061695
Iteration: 8300      Average Loss: 0.17822080836535228
Iteration: 8400      Average Loss: 0.177209786469524
Iteration: 8500      Average Loss: 0.21517888037187258
Iteration: 8600      Average Loss: 0.13037360769420464
Iteration: 8700      Average Loss: 0.2997790336769616
Iteration: 8800      Average Loss: 0.15018880464867534
Iteration: 8900      Average Loss: 0.13847980665643958
Iteration: 9000      Average Loss: 0.22914536832093213
Accuracy 0.72
Accuracy: [('Accuracy', 0.72)]
Iteration: 9100      Average Loss: 0.49864801475234066
Iteration: 9200      Average Loss: 0.3489727446720882
Iteration: 9300      Average Loss: 0.23072606262655326
Iteration: 9400      Average Loss: 0.32609516994006765
Iteration: 9500      Average Loss: 0.35267949139934324
Iteration: 9600      Average Loss: 0.22050194704803275
Iteration: 9700      Average Loss: 0.2005762506726679
Iteration: 9800      Average Loss: 0.27810890555705337
Iteration: 9900      Average Loss: 0.13212129675237302
Writing snapshot to model_iter_10000.mdl
Iteration: 10000     Average Loss: 0.1101929027724212
Accuracy 0.84
Accuracy: [('Accuracy', 0.84)]
Iteration: 10100     Average Loss: 0.18855726506280124
Iteration: 10200     Average Loss: 0.3637059169334043
Iteration: 10300     Average Loss: 0.16912870848969516
Iteration: 10400     Average Loss: 0.32256183637413643
Iteration: 10500     Average Loss: 0.22344494006137355
Iteration: 10600     Average Loss: 0.27796729405098725
Iteration: 10700     Average Loss: 0.368838967177586
Iteration: 10800     Average Loss: 0.19038791284974987
Iteration: 10900     Average Loss: 0.12780590417605034
Iteration: 11000     Average Loss: 0.17126897351451872
Accuracy 0.83
Accuracy: [('Accuracy', 0.83)]
Iteration: 11100     Average Loss: 0.29658769425261516
Iteration: 11200     Average Loss: 0.21608636487892258
Iteration: 11300     Average Loss: 0.07603814173859907
Iteration: 11400     Average Loss: 0.1526365050077243
Iteration: 11500     Average Loss: 0.29962201407151096
Iteration: 11600     Average Loss: 0.1444309853041516
Iteration: 11700     Average Loss: 0.3565173842342454
Iteration: 11800     Average Loss: 0.10750003459380544
Iteration: 11900     Average Loss: 0.2588887488563058
Iteration: 12000     Average Loss: 0.1548269002971158

```

Accuracy 0.87

Accuracy: [('Accuracy', 0.87)]

Iteration:	12100	Average Loss:	0.18050276808270893
Iteration:	12200	Average Loss:	0.15614154808485467
Iteration:	12300	Average Loss:	0.14875207718157518
Iteration:	12400	Average Loss:	0.10790540717273008
Iteration:	12500	Average Loss:	0.315453188946049
Iteration:	12600	Average Loss:	0.21130407032190643
Iteration:	12700	Average Loss:	0.06469882635289345
Iteration:	12800	Average Loss:	0.13829876475450564
Iteration:	12900	Average Loss:	0.13362351232384756
Iteration:	13000	Average Loss:	0.250083425753315

Accuracy 0.83

Accuracy: [('Accuracy', 0.83)]

Iteration:	13100	Average Loss:	0.17974605195721266
Iteration:	13200	Average Loss:	0.21718874018651754
Iteration:	13300	Average Loss:	0.20975998157781428
Iteration:	13400	Average Loss:	0.195560313118142
Iteration:	13500	Average Loss:	0.2304627580344018
Iteration:	13600	Average Loss:	0.26872804374052994
Iteration:	13700	Average Loss:	0.2478820332569064
Iteration:	13800	Average Loss:	0.244254075671261
Iteration:	13900	Average Loss:	0.2233945421707102
Iteration:	14000	Average Loss:	0.1859459717007629

Accuracy 0.86

Accuracy: [('Accuracy', 0.86)]

Iteration:	14100	Average Loss:	0.19912916902719816
Iteration:	14200	Average Loss:	0.060020521338157405
Iteration:	14300	Average Loss:	0.26602334901277147
Iteration:	14400	Average Loss:	0.1220623273129908
Iteration:	14500	Average Loss:	0.10465937143231091
Iteration:	14600	Average Loss:	0.1481839173751317
Iteration:	14700	Average Loss:	0.21388832420664436
Iteration:	14800	Average Loss:	0.1104962706545833
Iteration:	14900	Average Loss:	0.22459130092816865
Iteration:	15000	Average Loss:	0.18214141349062196

Accuracy 0.83

Accuracy: [('Accuracy', 0.83)]

Iteration:	15100	Average Loss:	0.1871358213128478
Iteration:	15200	Average Loss:	0.35690733824413096
Iteration:	15300	Average Loss:	0.04174833139761721
Iteration:	15400	Average Loss:	0.19727947596500744
Iteration:	15500	Average Loss:	0.04823643787857873
Iteration:	15600	Average Loss:	0.22600652576822014
Iteration:	15700	Average Loss:	0.12991806360599725
Iteration:	15800	Average Loss:	0.26572445425944957
Iteration:	15900	Average Loss:	0.08741766321940934
Iteration:	16000	Average Loss:	0.06170435390822125

Accuracy 0.81

Accuracy: [('Accuracy', 0.81)]

Iteration:	16100	Average Loss:	0.13364141581511452
Iteration:	16200	Average Loss:	0.32951216755654966
Iteration:	16300	Average Loss:	0.4521692604880683
Iteration:	16400	Average Loss:	0.1238315776573363
Iteration:	16500	Average Loss:	0.16988146281527847
Iteration:	16600	Average Loss:	0.23458714307430595
Iteration:	16700	Average Loss:	0.24102218334773653
Iteration:	16800	Average Loss:	0.1646592615555333
Iteration:	16900	Average Loss:	0.17130136727723014
Iteration:	17000	Average Loss:	0.19653355677317116

Accuracy 0.85

Accuracy: [('Accuracy', 0.85)]

Iteration:	17100	Average Loss:	0.1986736865042129
Iteration:	17200	Average Loss:	0.19522068253788696
Iteration:	17300	Average Loss:	0.14325218400273262
Iteration:	17400	Average Loss:	0.18318719216193546
Iteration:	17500	Average Loss:	0.10102564543996034
Iteration:	17600	Average Loss:	0.13642386126607156
Iteration:	17700	Average Loss:	0.15482608935787162
Iteration:	17800	Average Loss:	0.0636872419676467
Iteration:	17900	Average Loss:	0.06952205777461204
Iteration:	18000	Average Loss:	0.06556551914824965

Accuracy 0.86

Accuracy: [('Accuracy', 0.86)]

Iteration:	18100	Average Loss:	0.28443239358081657
Iteration:	18200	Average Loss:	0.1420605256675179
Iteration:	18300	Average Loss:	0.19017446824402895
Iteration:	18400	Average Loss:	0.119873642194792
Iteration:	18500	Average Loss:	0.18503009167296894
Iteration:	18600	Average Loss:	0.2062310322886002
Iteration:	18700	Average Loss:	0.12182318809254752
Iteration:	18800	Average Loss:	0.15759335118361856
Iteration:	18900	Average Loss:	0.28836898378127607
Iteration:	19000	Average Loss:	0.17873767188025283

Accuracy 0.86

Accuracy: [('Accuracy', 0.86)]

Iteration:	19100	Average Loss:	0.05313786162711513
Iteration:	19200	Average Loss:	0.15693677172999643
Iteration:	19300	Average Loss:	0.36028146782293713
Iteration:	19400	Average Loss:	0.13989691103646862
Iteration:	19500	Average Loss:	0.2629229302086418
Iteration:	19600	Average Loss:	0.13605091596459098
Iteration:	19700	Average Loss:	0.1638555775631413
Iteration:	19800	Average Loss:	0.1761741385136119
Iteration:	19900	Average Loss:	0.10216504212999389

Writing snapshot to model\_iter\_20000.mdl



Iteration: 20000      Average Loss: 0.08815498326884992  
Accuracy 0.85  
Accuracy: [('Accuracy', 0.85)]

Iteration: 20100      Average Loss: 0.1501731259997029  
Iteration: 20200      Average Loss: 0.07994057334342344  
Iteration: 20300      Average Loss: 0.3011325756211247  
Iteration: 20400      Average Loss: 0.2361803727376902  
Iteration: 20500      Average Loss: 0.3433850690875873  
Iteration: 20600      Average Loss: 0.14127775283490676  
Iteration: 20700      Average Loss: 0.16844244716974605  
Iteration: 20800      Average Loss: 0.13656010745411026  
Iteration: 20900      Average Loss: 0.07328913959484  
Iteration: 21000      Average Loss: 0.05936463592969809  
Accuracy 0.85  
Accuracy: [('Accuracy', 0.85)]

Iteration: 21100      Average Loss: 0.1410562166466517  
Iteration: 21200      Average Loss: 0.2459133145377473  
Iteration: 21300      Average Loss: 0.22872364597021216  
Iteration: 21400      Average Loss: 0.22678056256117443  
Iteration: 21500      Average Loss: 0.07128727839938627  
Iteration: 21600      Average Loss: 0.4007098118339161  
Iteration: 21700      Average Loss: 0.13521862887001063  
Iteration: 21800      Average Loss: 0.04532249153991105  
Iteration: 21900      Average Loss: 0.04153429093874996  
Iteration: 22000      Average Loss: 0.35716422806874154  
Accuracy 0.82  
Accuracy: [('Accuracy', 0.82)]

Iteration: 22100      Average Loss: 0.14399104337702037  
Iteration: 22200      Average Loss: 0.06220280460214771  
Iteration: 22300      Average Loss: 0.19642637556266887  
Iteration: 22400      Average Loss: 0.2846994361620914  
Iteration: 22500      Average Loss: 0.09362669029871551  
Iteration: 22600      Average Loss: 0.09869870561149259  
Iteration: 22700      Average Loss: 0.1478284725514119  
Iteration: 22800      Average Loss: 0.1737527904439888  
Iteration: 22900      Average Loss: 0.18545520343159244  
Iteration: 23000      Average Loss: 0.13425320307596517  
Accuracy 0.9  
Accuracy: [('Accuracy', 0.9)]

Iteration: 23100      Average Loss: 0.29955638782027444  
Iteration: 23200      Average Loss: 0.20835539731898906  
Iteration: 23300      Average Loss: 0.11787348743617608  
Iteration: 23400      Average Loss: 0.09203501590838377  
Iteration: 23500      Average Loss: 0.2950658216775856  
Iteration: 23600      Average Loss: 0.0776327429472613  
Iteration: 23700      Average Loss: 0.09803862780407097  
Iteration: 23800      Average Loss: 0.11829571150761285  
Iteration: 23900      Average Loss: 0.10415134066786798

Iteration: 24000      Average Loss: 0.03444078940989721  
Accuracy 0.87  
Accuracy: [('Accuracy', 0.87)]

Iteration: 24100      Average Loss: 0.19694593231183408  
Iteration: 24200      Average Loss: 0.26057716874174974  
Iteration: 24300      Average Loss: 0.22687086344818608  
Iteration: 24400      Average Loss: 0.07545864768070235  
Iteration: 24500      Average Loss: 0.3235859398640678  
Iteration: 24600      Average Loss: 0.28394523842672104  
Iteration: 24700      Average Loss: 0.05015293536826416  
Iteration: 24800      Average Loss: 0.20062945222087236  
Iteration: 24900      Average Loss: 0.3011304018640948  
Iteration: 25000      Average Loss: 0.12229440973007188  
Accuracy 0.85  
Accuracy: [('Accuracy', 0.85)]

Iteration: 25100      Average Loss: 0.24169506889236916  
Iteration: 25200      Average Loss: 0.20518786199998085  
Iteration: 25300      Average Loss: 0.150049556231767  
Iteration: 25400      Average Loss: 0.3495068599732552  
Iteration: 25500      Average Loss: 0.15067074588555868  
Iteration: 25600      Average Loss: 0.25647651356653683  
Iteration: 25700      Average Loss: 0.20826050784031302  
Iteration: 25800      Average Loss: 0.04871066254392478  
Iteration: 25900      Average Loss: 0.07162535181368736  
Iteration: 26000      Average Loss: 0.18689846599785323  
Accuracy 0.87  
Accuracy: [('Accuracy', 0.87)]

Iteration: 26100      Average Loss: 0.11819776438482045  
Iteration: 26200      Average Loss: 0.06168726133555815  
Iteration: 26300      Average Loss: 0.06394185513363959  
Iteration: 26400      Average Loss: 0.26177896079814855  
Iteration: 26500      Average Loss: 0.13410847696652145  
Iteration: 26600      Average Loss: 0.1453440105437357  
Iteration: 26700      Average Loss: 0.1478654342466147  
Iteration: 26800      Average Loss: 0.17542430658449695  
Iteration: 26900      Average Loss: 0.13034148420179625  
Iteration: 27000      Average Loss: 0.1011424781891197  
Accuracy 0.88  
Accuracy: [('Accuracy', 0.88)]

Iteration: 27100      Average Loss: 0.264385389836211  
Iteration: 27200      Average Loss: 0.1797835312620503  
Iteration: 27300      Average Loss: 0.12450619114676136  
Iteration: 27400      Average Loss: 0.04910959119162541  
Iteration: 27500      Average Loss: 0.09813641028506685  
Iteration: 27600      Average Loss: 0.20209759978224007  
Iteration: 27700      Average Loss: 0.19180123935367882  
Iteration: 27800      Average Loss: 0.17065431654111657  
Iteration: 27900      Average Loss: 0.06088158472967379

```
Iteration: 28000      Average Loss: 0.13802649927462543
Accuracy 0.86
Accuracy: [('Accuracy', 0.86)]
Iteration: 28100      Average Loss: 0.2685439694554405
Iteration: 28200      Average Loss: 0.1243713723373973
Iteration: 28300      Average Loss: 0.15584777911549666
Iteration: 28400      Average Loss: 0.13159421331148694
Iteration: 28500      Average Loss: 0.24912571543092288
Iteration: 28600      Average Loss: 0.21351914973236805
Iteration: 28700      Average Loss: 0.048721407504111684
Iteration: 28800      Average Loss: 0.06674696413944373
Iteration: 28900      Average Loss: 0.1244021672658857
Iteration: 29000      Average Loss: 0.08707496375229079
Accuracy 0.88
Accuracy: [('Accuracy', 0.88)]
Iteration: 29100      Average Loss: 0.1568919748974675
Iteration: 29200      Average Loss: 0.19004072375786382
Iteration: 29300      Average Loss: 0.17106205520813547
Iteration: 29400      Average Loss: 0.15944537665776498
Iteration: 29500      Average Loss: 0.12034811635071187
Iteration: 29600      Average Loss: 0.1834175517091623
Iteration: 29700      Average Loss: 0.10318922853475122
Iteration: 29800      Average Loss: 0.10304018717736269
Iteration: 29900      Average Loss: 0.14917122270575692
Writing snapshot to model_iter_30000.mdl
Iteration: 30000      Average Loss: 0.13562046811145825
Accuracy 0.85
Accuracy: [('Accuracy', 0.85)]
Epoch time: 3228.0903418064117
```

[3]: <logger.Logger at 0x1300bb080>

[ ]: