

FullyConnectedNetsContinued

February 16, 2024

```
[1]: # This mounts your Google Drive to the Colab VM.
from google.colab import drive
drive.mount('/content/drive')

# TODO: Enter the foldername in your Drive where you have saved the unzipped
# assignment folder, e.g. 'cse493g1/assignments/assignment3/'
FOLDERNAME = 'cse493g1/assignments/assignment3/'
assert FOLDERNAME is not None, "[!] Enter the foldername."

# Now that we've mounted your Drive, this ensures that
# the Python interpreter of the Colab VM can load
# python files from within it.
import sys
sys.path.append('/content/drive/My Drive/{}'.format(FOLDERNAME))

# This downloads the COCO dataset to your Drive
# if it doesn't already exist.
%cd /content/drive/My\ Drive/$FOLDERNAME/cse493g1/datasets/
!bash get_datasets.sh
%cd /content/drive/My\ Drive/$FOLDERNAME
```

```
Mounted at /content/drive
/content/drive/My Drive/cse493g1/assignments/assignment3/cse493g1/datasets
/content/drive/My Drive/cse493g1/assignments/assignment3
```

1 Multi-Layer Fully Connected Network Part 2

In this exercise, you will extend your fully connected network from Assignment 2 with Dropout and Normalization Layers. First, you will copy and paste all the necessary parts from Assignment 2. Then you will re-train your model from A2 as a baseline. Next, you will complete the batchnorm and dropout notebook, and then return to this notebook and create an improved model using dropout and normalization.

```
[4]: # Setup cell.
import time
import numpy as np
import matplotlib.pyplot as plt
```

```

from cse493g1.classifiers.fc_net import *
from cse493g1.data_utils import get_CIFAR10_data
from cse493g1.gradient_check import eval_numerical_gradient, \
    eval_numerical_gradient_array
from cse493g1.solver import Solver

%matplotlib inline
plt.rcParams["figure.figsize"] = (10.0, 8.0) # Set default size of plots.
plt.rcParams["image.interpolation"] = "nearest"
plt.rcParams["image.cmap"] = "gray"

%load_ext autoreload
%autoreload 2

def rel_error(x, y):
    """Returns relative error."""
    return np.max(np.abs(x - y) / (np.maximum(1e-8, np.abs(x) + np.abs(y))))

```

```

[5]: # Load the (preprocessed) CIFAR-10 data.
data = get_CIFAR10_data()
for k, v in list(data.items()):
    print(f"{k}: {v.shape}")

```

```

X_train: (49000, 3, 32, 32)
y_train: (49000,)
X_val: (1000, 3, 32, 32)
y_val: (1000,)
X_test: (1000, 3, 32, 32)
y_test: (1000,)

```

2 Copy necessary parts from A2.

Fill in the following functions by copying and pasting your answers from A2: `affine_forward` in `cse493g1/layers.py` `affine_backward` in `cse493g1/layers.py` `relu_forward` in `cse493g1/layers.py` `relu_backward` in `cse493g1/layers.py` `softmax_loss` in `cse493g1/layers.py` `sgd_momentum` in `cse493g1/optim.py` `rmsprop` in `cse493g1/optim.py` `adam` in `cse493g1/optim.py`

3 Train baseline model from A2

Copy and Paste your `FullyConnectedNet` model from `cse493g1/classifiers/fc_net.py` in Assignment 2 into `FullyConnectedNetBasic` in the file `cse493g1/classifiers/fc_net.py` in this assignment. Use the best hyperparams that you found from the previous assignment to train this model. Call this model `best_model_basic`

```

[7]: best_model_basic = None

#####
# TODO: Train the best FullyConnectedNetBasic that you can on CIFAR-10. Store
# your best model in #
# the best_model_basic variable.
#
#####
# *****START OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****

weight_scale = [1e-1]
learning_rate = [2e-4]
update_rule = ['rmsprop']
reg = [1e-3]

results = {}
best_val = 0
best_solver = None
solvers = {}

for i in weight_scale:
    for j in learning_rate:
        for rule in update_rule:
            model = FullyConnectedNetBasic(
                [100, 100, 100, 100, 100],
                weight_scale = i,
                dtype=np.float64,
                reg = reg[0],
            )
            solver = Solver(
                model,
                data,
                print_every=10,
                num_epochs=20,
                batch_size=25,
                update_rule= rule,
                optim_config={"learning_rate": j},
            )
            solver.train()

            solvers[(i,j,rule)] = solver

            if solver.best_val_acc > best_val:
                best_model = model
                best_solver = solver
                best_val = solver.best_val_acc

```

```

        results[(i,j,rule)] = (solver.train_acc_history[len(solver.
↪train_acc_history) - 1], solver.best_val_acc)

# Print out results.
for i, j, rule in sorted(results):
    train_accuracy, val_accuracy = results[(i,j,rule)]
    print('weight_scale %e lr %e rule %s train accuracy: %f val accuracy: %f'↪
↪% (
        i, j, rule, train_accuracy, val_accuracy))

print('best val accuracy achieved: %f' % best_val)

# Print out results.
for i, j, rule in sorted(results):
    train_accuracy, val_accuracy = results[(i,j,rule)]
    print('weight_scale %e lr %e rule %s train accuracy: %f val accuracy: %f'↪
↪% (
        i, j, rule, train_accuracy, val_accuracy))

print('best val accuracy achieved: %f' % best_val)

fig, axes = plt.subplots(3, 1, figsize=(15, 15))

axes[0].set_title('Training loss')
axes[0].set_xlabel('Iteration')
axes[1].set_title('Training accuracy')
axes[1].set_xlabel('Epoch')
axes[2].set_title('Validation accuracy')
axes[2].set_xlabel('Epoch')

for (i,j, update_rule), solver in solvers.items():
    axes[0].plot(solver.loss_history, label=f"{update_rule}, ws{i}, lr{j}")
    axes[1].plot(solver.train_acc_history, label=f"{update_rule}, ws{i}, lr{j}")
    axes[2].plot(solver.val_acc_history, label=f"{update_rule}, ws{i}, lr{j}")

for ax in axes:
    ax.legend(loc='best', ncol=4)
    ax.grid(linestyle='--', linewidth=0.5)

plt.show()

# *****END OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****
#####
#                               END OF YOUR CODE                               #
#####

```

(Iteration 1 / 39200) loss: 136.025363
(Epoch 0 / 20) train acc: 0.108000; val_acc: 0.107000
(Iteration 11 / 39200) loss: 34.201549
(Iteration 21 / 39200) loss: 19.649691
(Iteration 31 / 39200) loss: 17.571646
(Iteration 41 / 39200) loss: 13.036979
(Iteration 51 / 39200) loss: 15.432630
(Iteration 61 / 39200) loss: 14.035347
(Iteration 71 / 39200) loss: 15.253355
(Iteration 81 / 39200) loss: 9.394079
(Iteration 91 / 39200) loss: 8.834803
(Iteration 101 / 39200) loss: 11.137217
(Iteration 111 / 39200) loss: 8.550465
(Iteration 121 / 39200) loss: 6.022767
(Iteration 131 / 39200) loss: 9.207627
(Iteration 141 / 39200) loss: 9.160192
(Iteration 151 / 39200) loss: 9.442897
(Iteration 161 / 39200) loss: 9.698607
(Iteration 171 / 39200) loss: 7.549053
(Iteration 181 / 39200) loss: 9.846977
(Iteration 191 / 39200) loss: 7.382191
(Iteration 201 / 39200) loss: 9.222998
(Iteration 211 / 39200) loss: 7.341811
(Iteration 221 / 39200) loss: 7.060704
(Iteration 231 / 39200) loss: 7.809970
(Iteration 241 / 39200) loss: 6.146948
(Iteration 251 / 39200) loss: 6.734282
(Iteration 261 / 39200) loss: 6.937071
(Iteration 271 / 39200) loss: 6.593870
(Iteration 281 / 39200) loss: 6.862037
(Iteration 291 / 39200) loss: 6.961892
(Iteration 301 / 39200) loss: 5.918625
(Iteration 311 / 39200) loss: 6.959797
(Iteration 321 / 39200) loss: 5.487266
(Iteration 331 / 39200) loss: 5.363025
(Iteration 341 / 39200) loss: 4.001301
(Iteration 351 / 39200) loss: 6.393135
(Iteration 361 / 39200) loss: 6.217662
(Iteration 371 / 39200) loss: 5.995852
(Iteration 381 / 39200) loss: 5.390829
(Iteration 391 / 39200) loss: 5.059617
(Iteration 401 / 39200) loss: 5.306854
(Iteration 411 / 39200) loss: 4.872647
(Iteration 421 / 39200) loss: 6.013467
(Iteration 431 / 39200) loss: 6.511731
(Iteration 441 / 39200) loss: 5.779819
(Iteration 451 / 39200) loss: 4.723229
(Iteration 461 / 39200) loss: 5.678699

(Iteration 471 / 39200) loss: 6.098169
(Iteration 481 / 39200) loss: 5.854549
(Iteration 491 / 39200) loss: 5.122373
(Iteration 501 / 39200) loss: 5.781552
(Iteration 511 / 39200) loss: 5.309787
(Iteration 521 / 39200) loss: 4.442580
(Iteration 531 / 39200) loss: 5.018847
(Iteration 541 / 39200) loss: 4.643604
(Iteration 551 / 39200) loss: 4.197594
(Iteration 561 / 39200) loss: 6.161136
(Iteration 571 / 39200) loss: 4.198220
(Iteration 581 / 39200) loss: 4.226728
(Iteration 591 / 39200) loss: 4.632739
(Iteration 601 / 39200) loss: 4.544496
(Iteration 611 / 39200) loss: 4.419462
(Iteration 621 / 39200) loss: 4.275639
(Iteration 631 / 39200) loss: 4.339673
(Iteration 641 / 39200) loss: 5.452063
(Iteration 651 / 39200) loss: 4.646847
(Iteration 661 / 39200) loss: 3.748199
(Iteration 671 / 39200) loss: 5.173741
(Iteration 681 / 39200) loss: 4.276820
(Iteration 691 / 39200) loss: 4.160534
(Iteration 701 / 39200) loss: 4.795418
(Iteration 711 / 39200) loss: 3.865275
(Iteration 721 / 39200) loss: 4.705771
(Iteration 731 / 39200) loss: 3.723930
(Iteration 741 / 39200) loss: 3.808452
(Iteration 751 / 39200) loss: 4.348354
(Iteration 761 / 39200) loss: 4.131405
(Iteration 771 / 39200) loss: 3.790942
(Iteration 781 / 39200) loss: 4.428376
(Iteration 791 / 39200) loss: 3.672378
(Iteration 801 / 39200) loss: 4.236956
(Iteration 811 / 39200) loss: 4.615590
(Iteration 821 / 39200) loss: 4.031827
(Iteration 831 / 39200) loss: 4.277832
(Iteration 841 / 39200) loss: 3.924296
(Iteration 851 / 39200) loss: 3.939674
(Iteration 861 / 39200) loss: 3.661154
(Iteration 871 / 39200) loss: 4.074649
(Iteration 881 / 39200) loss: 4.382121
(Iteration 891 / 39200) loss: 3.925285
(Iteration 901 / 39200) loss: 4.396526
(Iteration 911 / 39200) loss: 4.104952
(Iteration 921 / 39200) loss: 4.459718
(Iteration 931 / 39200) loss: 3.593474
(Iteration 941 / 39200) loss: 3.447874

(Iteration 951 / 39200) loss: 4.297481
(Iteration 961 / 39200) loss: 4.417256
(Iteration 971 / 39200) loss: 4.279238
(Iteration 981 / 39200) loss: 3.709321
(Iteration 991 / 39200) loss: 4.295752
(Iteration 1001 / 39200) loss: 3.382206
(Iteration 1011 / 39200) loss: 3.917764
(Iteration 1021 / 39200) loss: 3.927647
(Iteration 1031 / 39200) loss: 3.623823
(Iteration 1041 / 39200) loss: 3.527910
(Iteration 1051 / 39200) loss: 3.809920
(Iteration 1061 / 39200) loss: 3.730634
(Iteration 1071 / 39200) loss: 3.888255
(Iteration 1081 / 39200) loss: 3.881709
(Iteration 1091 / 39200) loss: 3.545221
(Iteration 1101 / 39200) loss: 3.621532
(Iteration 1111 / 39200) loss: 3.777926
(Iteration 1121 / 39200) loss: 3.707530
(Iteration 1131 / 39200) loss: 4.188969
(Iteration 1141 / 39200) loss: 3.767342
(Iteration 1151 / 39200) loss: 3.621569
(Iteration 1161 / 39200) loss: 3.753102
(Iteration 1171 / 39200) loss: 3.643129
(Iteration 1181 / 39200) loss: 3.745049
(Iteration 1191 / 39200) loss: 3.609722
(Iteration 1201 / 39200) loss: 3.372212
(Iteration 1211 / 39200) loss: 3.762241
(Iteration 1221 / 39200) loss: 3.905555
(Iteration 1231 / 39200) loss: 3.707787
(Iteration 1241 / 39200) loss: 4.095039
(Iteration 1251 / 39200) loss: 3.637087
(Iteration 1261 / 39200) loss: 3.613788
(Iteration 1271 / 39200) loss: 4.072870
(Iteration 1281 / 39200) loss: 4.075749
(Iteration 1291 / 39200) loss: 3.306274
(Iteration 1301 / 39200) loss: 3.464838
(Iteration 1311 / 39200) loss: 3.730838
(Iteration 1321 / 39200) loss: 3.546851
(Iteration 1331 / 39200) loss: 3.986468
(Iteration 1341 / 39200) loss: 3.673615
(Iteration 1351 / 39200) loss: 3.683968
(Iteration 1361 / 39200) loss: 3.846573
(Iteration 1371 / 39200) loss: 3.653247
(Iteration 1381 / 39200) loss: 3.272872
(Iteration 1391 / 39200) loss: 3.324147
(Iteration 1401 / 39200) loss: 4.066754
(Iteration 1411 / 39200) loss: 3.530615
(Iteration 1421 / 39200) loss: 3.641669

(Iteration 1431 / 39200) loss: 4.114246
(Iteration 1441 / 39200) loss: 3.461043
(Iteration 1451 / 39200) loss: 4.080728
(Iteration 1461 / 39200) loss: 3.511174
(Iteration 1471 / 39200) loss: 4.177226
(Iteration 1481 / 39200) loss: 3.465334
(Iteration 1491 / 39200) loss: 3.405536
(Iteration 1501 / 39200) loss: 3.082319
(Iteration 1511 / 39200) loss: 3.711611
(Iteration 1521 / 39200) loss: 3.965991
(Iteration 1531 / 39200) loss: 3.524269
(Iteration 1541 / 39200) loss: 3.628432
(Iteration 1551 / 39200) loss: 3.517626
(Iteration 1561 / 39200) loss: 3.410591
(Iteration 1571 / 39200) loss: 3.887053
(Iteration 1581 / 39200) loss: 3.388327
(Iteration 1591 / 39200) loss: 3.646362
(Iteration 1601 / 39200) loss: 3.599800
(Iteration 1611 / 39200) loss: 3.942549
(Iteration 1621 / 39200) loss: 3.313638
(Iteration 1631 / 39200) loss: 3.531429
(Iteration 1641 / 39200) loss: 3.370290
(Iteration 1651 / 39200) loss: 3.607873
(Iteration 1661 / 39200) loss: 3.443613
(Iteration 1671 / 39200) loss: 3.554817
(Iteration 1681 / 39200) loss: 3.743007
(Iteration 1691 / 39200) loss: 3.482305
(Iteration 1701 / 39200) loss: 3.287734
(Iteration 1711 / 39200) loss: 3.649658
(Iteration 1721 / 39200) loss: 3.956443
(Iteration 1731 / 39200) loss: 3.545032
(Iteration 1741 / 39200) loss: 3.245558
(Iteration 1751 / 39200) loss: 2.880141
(Iteration 1761 / 39200) loss: 3.809497
(Iteration 1771 / 39200) loss: 3.779579
(Iteration 1781 / 39200) loss: 3.613121
(Iteration 1791 / 39200) loss: 3.715955
(Iteration 1801 / 39200) loss: 3.403771
(Iteration 1811 / 39200) loss: 3.446335
(Iteration 1821 / 39200) loss: 3.762267
(Iteration 1831 / 39200) loss: 3.470947
(Iteration 1841 / 39200) loss: 3.592198
(Iteration 1851 / 39200) loss: 3.336435
(Iteration 1861 / 39200) loss: 3.369352
(Iteration 1871 / 39200) loss: 3.660029
(Iteration 1881 / 39200) loss: 3.367476
(Iteration 1891 / 39200) loss: 3.700319
(Iteration 1901 / 39200) loss: 3.507377

(Iteration 1911 / 39200) loss: 3.828626
(Iteration 1921 / 39200) loss: 3.375310
(Iteration 1931 / 39200) loss: 3.282761
(Iteration 1941 / 39200) loss: 2.948577
(Iteration 1951 / 39200) loss: 3.724050
(Epoch 1 / 20) train acc: 0.334000; val_acc: 0.344000
(Iteration 1961 / 39200) loss: 3.505215
(Iteration 1971 / 39200) loss: 3.381288
(Iteration 1981 / 39200) loss: 3.885605
(Iteration 1991 / 39200) loss: 3.678607
(Iteration 2001 / 39200) loss: 3.237485
(Iteration 2011 / 39200) loss: 3.688078
(Iteration 2021 / 39200) loss: 3.551990
(Iteration 2031 / 39200) loss: 3.310651
(Iteration 2041 / 39200) loss: 3.731407
(Iteration 2051 / 39200) loss: 3.543005
(Iteration 2061 / 39200) loss: 3.614353
(Iteration 2071 / 39200) loss: 3.762950
(Iteration 2081 / 39200) loss: 3.436876
(Iteration 2091 / 39200) loss: 3.218670
(Iteration 2101 / 39200) loss: 3.384303
(Iteration 2111 / 39200) loss: 3.633491
(Iteration 2121 / 39200) loss: 3.159504
(Iteration 2131 / 39200) loss: 3.886621
(Iteration 2141 / 39200) loss: 3.051616
(Iteration 2151 / 39200) loss: 3.196541
(Iteration 2161 / 39200) loss: 3.188263
(Iteration 2171 / 39200) loss: 3.313371
(Iteration 2181 / 39200) loss: 3.610757
(Iteration 2191 / 39200) loss: 3.658412
(Iteration 2201 / 39200) loss: 3.203356
(Iteration 2211 / 39200) loss: 3.833743
(Iteration 2221 / 39200) loss: 3.440474
(Iteration 2231 / 39200) loss: 3.237467
(Iteration 2241 / 39200) loss: 2.918727
(Iteration 2251 / 39200) loss: 3.361240
(Iteration 2261 / 39200) loss: 2.934837
(Iteration 2271 / 39200) loss: 3.062722
(Iteration 2281 / 39200) loss: 3.245880
(Iteration 2291 / 39200) loss: 3.331604
(Iteration 2301 / 39200) loss: 3.224732
(Iteration 2311 / 39200) loss: 3.754081
(Iteration 2321 / 39200) loss: 3.367424
(Iteration 2331 / 39200) loss: 3.411556
(Iteration 2341 / 39200) loss: 3.348870
(Iteration 2351 / 39200) loss: 3.229725
(Iteration 2361 / 39200) loss: 2.921593
(Iteration 2371 / 39200) loss: 2.757574

(Iteration 2381 / 39200) loss: 3.082541
(Iteration 2391 / 39200) loss: 3.248590
(Iteration 2401 / 39200) loss: 3.185946
(Iteration 2411 / 39200) loss: 3.758231
(Iteration 2421 / 39200) loss: 3.368408
(Iteration 2431 / 39200) loss: 3.312982
(Iteration 2441 / 39200) loss: 3.321049
(Iteration 2451 / 39200) loss: 3.922624
(Iteration 2461 / 39200) loss: 3.124777
(Iteration 2471 / 39200) loss: 3.402528
(Iteration 2481 / 39200) loss: 3.515463
(Iteration 2491 / 39200) loss: 3.306169
(Iteration 2501 / 39200) loss: 3.375134
(Iteration 2511 / 39200) loss: 3.010265
(Iteration 2521 / 39200) loss: 3.353846
(Iteration 2531 / 39200) loss: 3.497432
(Iteration 2541 / 39200) loss: 3.262451
(Iteration 2551 / 39200) loss: 3.307880
(Iteration 2561 / 39200) loss: 3.545259
(Iteration 2571 / 39200) loss: 3.214456
(Iteration 2581 / 39200) loss: 3.116961
(Iteration 2591 / 39200) loss: 2.950983
(Iteration 2601 / 39200) loss: 3.426030
(Iteration 2611 / 39200) loss: 3.438726
(Iteration 2621 / 39200) loss: 3.243711
(Iteration 2631 / 39200) loss: 3.106790
(Iteration 2641 / 39200) loss: 3.481344
(Iteration 2651 / 39200) loss: 3.366021
(Iteration 2661 / 39200) loss: 3.385766
(Iteration 2671 / 39200) loss: 3.372037
(Iteration 2681 / 39200) loss: 2.886022
(Iteration 2691 / 39200) loss: 3.212081
(Iteration 2701 / 39200) loss: 3.202280
(Iteration 2711 / 39200) loss: 3.436256
(Iteration 2721 / 39200) loss: 3.209576
(Iteration 2731 / 39200) loss: 3.402579
(Iteration 2741 / 39200) loss: 3.190095
(Iteration 2751 / 39200) loss: 3.007539
(Iteration 2761 / 39200) loss: 3.592044
(Iteration 2771 / 39200) loss: 3.205891
(Iteration 2781 / 39200) loss: 2.936235
(Iteration 2791 / 39200) loss: 3.490517
(Iteration 2801 / 39200) loss: 3.286008
(Iteration 2811 / 39200) loss: 3.667073
(Iteration 2821 / 39200) loss: 3.258292
(Iteration 2831 / 39200) loss: 3.223933
(Iteration 2841 / 39200) loss: 3.407788
(Iteration 2851 / 39200) loss: 3.563093

(Iteration 2861 / 39200) loss: 3.620454
(Iteration 2871 / 39200) loss: 3.709186
(Iteration 2881 / 39200) loss: 3.424971
(Iteration 2891 / 39200) loss: 3.201710
(Iteration 2901 / 39200) loss: 3.406947
(Iteration 2911 / 39200) loss: 3.564361
(Iteration 2921 / 39200) loss: 3.081817
(Iteration 2931 / 39200) loss: 3.225124
(Iteration 2941 / 39200) loss: 3.142530
(Iteration 2951 / 39200) loss: 3.174958
(Iteration 2961 / 39200) loss: 3.119831
(Iteration 2971 / 39200) loss: 3.864131
(Iteration 2981 / 39200) loss: 3.729200
(Iteration 2991 / 39200) loss: 3.295252
(Iteration 3001 / 39200) loss: 3.384685
(Iteration 3011 / 39200) loss: 3.192480
(Iteration 3021 / 39200) loss: 3.541228
(Iteration 3031 / 39200) loss: 3.468862
(Iteration 3041 / 39200) loss: 3.276690
(Iteration 3051 / 39200) loss: 3.253134
(Iteration 3061 / 39200) loss: 3.325199
(Iteration 3071 / 39200) loss: 3.226794
(Iteration 3081 / 39200) loss: 2.958428
(Iteration 3091 / 39200) loss: 3.062960
(Iteration 3101 / 39200) loss: 3.030063
(Iteration 3111 / 39200) loss: 3.063672
(Iteration 3121 / 39200) loss: 3.223459
(Iteration 3131 / 39200) loss: 3.135352
(Iteration 3141 / 39200) loss: 2.900344
(Iteration 3151 / 39200) loss: 3.216591
(Iteration 3161 / 39200) loss: 2.862086
(Iteration 3171 / 39200) loss: 3.178461
(Iteration 3181 / 39200) loss: 3.366355
(Iteration 3191 / 39200) loss: 3.150247
(Iteration 3201 / 39200) loss: 3.419816
(Iteration 3211 / 39200) loss: 3.207987
(Iteration 3221 / 39200) loss: 3.116528
(Iteration 3231 / 39200) loss: 2.970011
(Iteration 3241 / 39200) loss: 2.833123
(Iteration 3251 / 39200) loss: 2.782196
(Iteration 3261 / 39200) loss: 3.002240
(Iteration 3271 / 39200) loss: 2.952597
(Iteration 3281 / 39200) loss: 3.263433
(Iteration 3291 / 39200) loss: 3.004744
(Iteration 3301 / 39200) loss: 3.345534
(Iteration 3311 / 39200) loss: 2.884347
(Iteration 3321 / 39200) loss: 3.494836
(Iteration 3331 / 39200) loss: 3.347905

(Iteration 3341 / 39200) loss: 3.178701
(Iteration 3351 / 39200) loss: 3.235277
(Iteration 3361 / 39200) loss: 3.223435
(Iteration 3371 / 39200) loss: 3.310693
(Iteration 3381 / 39200) loss: 3.030553
(Iteration 3391 / 39200) loss: 3.268487
(Iteration 3401 / 39200) loss: 3.448085
(Iteration 3411 / 39200) loss: 3.029840
(Iteration 3421 / 39200) loss: 3.127801
(Iteration 3431 / 39200) loss: 3.191963
(Iteration 3441 / 39200) loss: 2.862219
(Iteration 3451 / 39200) loss: 3.077274
(Iteration 3461 / 39200) loss: 3.163788
(Iteration 3471 / 39200) loss: 3.402293
(Iteration 3481 / 39200) loss: 3.191910
(Iteration 3491 / 39200) loss: 3.579309
(Iteration 3501 / 39200) loss: 3.381463
(Iteration 3511 / 39200) loss: 3.333033
(Iteration 3521 / 39200) loss: 2.845671
(Iteration 3531 / 39200) loss: 3.276480
(Iteration 3541 / 39200) loss: 3.317976
(Iteration 3551 / 39200) loss: 2.725640
(Iteration 3561 / 39200) loss: 3.024667
(Iteration 3571 / 39200) loss: 3.345777
(Iteration 3581 / 39200) loss: 2.875345
(Iteration 3591 / 39200) loss: 3.093898
(Iteration 3601 / 39200) loss: 3.732581
(Iteration 3611 / 39200) loss: 3.077805
(Iteration 3621 / 39200) loss: 2.767805
(Iteration 3631 / 39200) loss: 3.320561
(Iteration 3641 / 39200) loss: 2.816510
(Iteration 3651 / 39200) loss: 3.330732
(Iteration 3661 / 39200) loss: 3.575874
(Iteration 3671 / 39200) loss: 3.208624
(Iteration 3681 / 39200) loss: 3.285726
(Iteration 3691 / 39200) loss: 2.905352
(Iteration 3701 / 39200) loss: 3.169637
(Iteration 3711 / 39200) loss: 3.009014
(Iteration 3721 / 39200) loss: 3.343477
(Iteration 3731 / 39200) loss: 2.317568
(Iteration 3741 / 39200) loss: 2.954190
(Iteration 3751 / 39200) loss: 3.498424
(Iteration 3761 / 39200) loss: 3.476037
(Iteration 3771 / 39200) loss: 2.902821
(Iteration 3781 / 39200) loss: 2.875398
(Iteration 3791 / 39200) loss: 3.126293
(Iteration 3801 / 39200) loss: 3.204589
(Iteration 3811 / 39200) loss: 3.181433

(Iteration 3821 / 39200) loss: 2.985359
(Iteration 3831 / 39200) loss: 2.778364
(Iteration 3841 / 39200) loss: 3.074476
(Iteration 3851 / 39200) loss: 3.033009
(Iteration 3861 / 39200) loss: 3.332035
(Iteration 3871 / 39200) loss: 2.862709
(Iteration 3881 / 39200) loss: 3.204672
(Iteration 3891 / 39200) loss: 3.197316
(Iteration 3901 / 39200) loss: 3.146097
(Iteration 3911 / 39200) loss: 2.968321
(Epoch 2 / 20) train acc: 0.421000; val_acc: 0.406000
(Iteration 3921 / 39200) loss: 3.327144
(Iteration 3931 / 39200) loss: 2.823418
(Iteration 3941 / 39200) loss: 2.895685
(Iteration 3951 / 39200) loss: 3.281809
(Iteration 3961 / 39200) loss: 3.256995
(Iteration 3971 / 39200) loss: 3.020196
(Iteration 3981 / 39200) loss: 3.188438
(Iteration 3991 / 39200) loss: 3.276076
(Iteration 4001 / 39200) loss: 2.733888
(Iteration 4011 / 39200) loss: 3.182316
(Iteration 4021 / 39200) loss: 2.701340
(Iteration 4031 / 39200) loss: 3.107354
(Iteration 4041 / 39200) loss: 2.936245
(Iteration 4051 / 39200) loss: 2.739229
(Iteration 4061 / 39200) loss: 2.633908
(Iteration 4071 / 39200) loss: 3.230101
(Iteration 4081 / 39200) loss: 3.258620
(Iteration 4091 / 39200) loss: 3.303636
(Iteration 4101 / 39200) loss: 2.892619
(Iteration 4111 / 39200) loss: 2.919555
(Iteration 4121 / 39200) loss: 2.868130
(Iteration 4131 / 39200) loss: 3.301118
(Iteration 4141 / 39200) loss: 2.618923
(Iteration 4151 / 39200) loss: 3.237413
(Iteration 4161 / 39200) loss: 2.863193
(Iteration 4171 / 39200) loss: 3.023395
(Iteration 4181 / 39200) loss: 3.050468
(Iteration 4191 / 39200) loss: 3.094286
(Iteration 4201 / 39200) loss: 3.175768
(Iteration 4211 / 39200) loss: 2.947032
(Iteration 4221 / 39200) loss: 2.864340
(Iteration 4231 / 39200) loss: 3.105162
(Iteration 4241 / 39200) loss: 3.264139
(Iteration 4251 / 39200) loss: 3.036915
(Iteration 4261 / 39200) loss: 3.049275
(Iteration 4271 / 39200) loss: 3.110604
(Iteration 4281 / 39200) loss: 3.224573

(Iteration 4291 / 39200) loss: 3.158053
(Iteration 4301 / 39200) loss: 2.762339
(Iteration 4311 / 39200) loss: 2.803091
(Iteration 4321 / 39200) loss: 2.881840
(Iteration 4331 / 39200) loss: 3.179492
(Iteration 4341 / 39200) loss: 2.960142
(Iteration 4351 / 39200) loss: 3.242079
(Iteration 4361 / 39200) loss: 2.802050
(Iteration 4371 / 39200) loss: 2.954093
(Iteration 4381 / 39200) loss: 3.035370
(Iteration 4391 / 39200) loss: 2.756453
(Iteration 4401 / 39200) loss: 2.647021
(Iteration 4411 / 39200) loss: 3.049214
(Iteration 4421 / 39200) loss: 3.030488
(Iteration 4431 / 39200) loss: 2.877733
(Iteration 4441 / 39200) loss: 2.881748
(Iteration 4451 / 39200) loss: 3.189816
(Iteration 4461 / 39200) loss: 3.032055
(Iteration 4471 / 39200) loss: 3.128214
(Iteration 4481 / 39200) loss: 2.912879
(Iteration 4491 / 39200) loss: 2.972567
(Iteration 4501 / 39200) loss: 2.930211
(Iteration 4511 / 39200) loss: 3.059472
(Iteration 4521 / 39200) loss: 2.793582
(Iteration 4531 / 39200) loss: 2.878688
(Iteration 4541 / 39200) loss: 3.093980
(Iteration 4551 / 39200) loss: 2.762044
(Iteration 4561 / 39200) loss: 3.018310
(Iteration 4571 / 39200) loss: 3.011217
(Iteration 4581 / 39200) loss: 2.725128
(Iteration 4591 / 39200) loss: 3.210983
(Iteration 4601 / 39200) loss: 2.965601
(Iteration 4611 / 39200) loss: 2.997447
(Iteration 4621 / 39200) loss: 2.915439
(Iteration 4631 / 39200) loss: 3.053812
(Iteration 4641 / 39200) loss: 2.888320
(Iteration 4651 / 39200) loss: 3.417176
(Iteration 4661 / 39200) loss: 3.218676
(Iteration 4671 / 39200) loss: 2.854053
(Iteration 4681 / 39200) loss: 2.861007
(Iteration 4691 / 39200) loss: 3.007966
(Iteration 4701 / 39200) loss: 3.381682
(Iteration 4711 / 39200) loss: 2.971967
(Iteration 4721 / 39200) loss: 3.253596
(Iteration 4731 / 39200) loss: 2.852307
(Iteration 4741 / 39200) loss: 2.705278
(Iteration 4751 / 39200) loss: 2.763834
(Iteration 4761 / 39200) loss: 2.931174

(Iteration 4771 / 39200) loss: 3.391595
(Iteration 4781 / 39200) loss: 2.674834
(Iteration 4791 / 39200) loss: 2.865780
(Iteration 4801 / 39200) loss: 2.725842
(Iteration 4811 / 39200) loss: 2.676187
(Iteration 4821 / 39200) loss: 3.289843
(Iteration 4831 / 39200) loss: 2.693651
(Iteration 4841 / 39200) loss: 2.881165
(Iteration 4851 / 39200) loss: 2.554965
(Iteration 4861 / 39200) loss: 3.190941
(Iteration 4871 / 39200) loss: 2.853137
(Iteration 4881 / 39200) loss: 2.628571
(Iteration 4891 / 39200) loss: 2.497831
(Iteration 4901 / 39200) loss: 2.892716
(Iteration 4911 / 39200) loss: 3.137983
(Iteration 4921 / 39200) loss: 2.952922
(Iteration 4931 / 39200) loss: 2.994808
(Iteration 4941 / 39200) loss: 2.950455
(Iteration 4951 / 39200) loss: 2.793064
(Iteration 4961 / 39200) loss: 2.946297
(Iteration 4971 / 39200) loss: 2.634424
(Iteration 4981 / 39200) loss: 3.190968
(Iteration 4991 / 39200) loss: 3.019407
(Iteration 5001 / 39200) loss: 3.205560
(Iteration 5011 / 39200) loss: 3.212892
(Iteration 5021 / 39200) loss: 3.171984
(Iteration 5031 / 39200) loss: 2.652126
(Iteration 5041 / 39200) loss: 2.912676
(Iteration 5051 / 39200) loss: 2.890470
(Iteration 5061 / 39200) loss: 2.957491
(Iteration 5071 / 39200) loss: 2.733039
(Iteration 5081 / 39200) loss: 2.982285
(Iteration 5091 / 39200) loss: 3.047798
(Iteration 5101 / 39200) loss: 2.605297
(Iteration 5111 / 39200) loss: 2.741088
(Iteration 5121 / 39200) loss: 2.807489
(Iteration 5131 / 39200) loss: 2.792645
(Iteration 5141 / 39200) loss: 3.069767
(Iteration 5151 / 39200) loss: 2.421063
(Iteration 5161 / 39200) loss: 3.365627
(Iteration 5171 / 39200) loss: 2.931994
(Iteration 5181 / 39200) loss: 2.997925
(Iteration 5191 / 39200) loss: 2.714547
(Iteration 5201 / 39200) loss: 3.036561
(Iteration 5211 / 39200) loss: 2.802244
(Iteration 5221 / 39200) loss: 2.399456
(Iteration 5231 / 39200) loss: 2.906390
(Iteration 5241 / 39200) loss: 2.714423

(Iteration 5251 / 39200) loss: 3.679976
(Iteration 5261 / 39200) loss: 2.705924
(Iteration 5271 / 39200) loss: 2.867982
(Iteration 5281 / 39200) loss: 2.608660
(Iteration 5291 / 39200) loss: 2.845116
(Iteration 5301 / 39200) loss: 3.091342
(Iteration 5311 / 39200) loss: 2.850087
(Iteration 5321 / 39200) loss: 2.690519
(Iteration 5331 / 39200) loss: 3.145631
(Iteration 5341 / 39200) loss: 2.788824
(Iteration 5351 / 39200) loss: 3.116514
(Iteration 5361 / 39200) loss: 3.146677
(Iteration 5371 / 39200) loss: 2.803856
(Iteration 5381 / 39200) loss: 2.731618
(Iteration 5391 / 39200) loss: 2.904035
(Iteration 5401 / 39200) loss: 2.476748
(Iteration 5411 / 39200) loss: 2.808194
(Iteration 5421 / 39200) loss: 3.406035
(Iteration 5431 / 39200) loss: 2.802538
(Iteration 5441 / 39200) loss: 3.177813
(Iteration 5451 / 39200) loss: 2.606537
(Iteration 5461 / 39200) loss: 2.658827
(Iteration 5471 / 39200) loss: 2.976672
(Iteration 5481 / 39200) loss: 2.817103
(Iteration 5491 / 39200) loss: 3.033258
(Iteration 5501 / 39200) loss: 2.740711
(Iteration 5511 / 39200) loss: 3.009869
(Iteration 5521 / 39200) loss: 2.711364
(Iteration 5531 / 39200) loss: 2.784343
(Iteration 5541 / 39200) loss: 3.282230
(Iteration 5551 / 39200) loss: 2.809130
(Iteration 5561 / 39200) loss: 2.696763
(Iteration 5571 / 39200) loss: 3.027292
(Iteration 5581 / 39200) loss: 3.011193
(Iteration 5591 / 39200) loss: 2.931713
(Iteration 5601 / 39200) loss: 3.086819
(Iteration 5611 / 39200) loss: 2.942231
(Iteration 5621 / 39200) loss: 2.999569
(Iteration 5631 / 39200) loss: 2.867510
(Iteration 5641 / 39200) loss: 2.431661
(Iteration 5651 / 39200) loss: 3.011163
(Iteration 5661 / 39200) loss: 2.752628
(Iteration 5671 / 39200) loss: 2.774662
(Iteration 5681 / 39200) loss: 2.847132
(Iteration 5691 / 39200) loss: 2.481079
(Iteration 5701 / 39200) loss: 3.005304
(Iteration 5711 / 39200) loss: 2.670948
(Iteration 5721 / 39200) loss: 2.647782

(Iteration 5731 / 39200) loss: 2.913391
(Iteration 5741 / 39200) loss: 2.889358
(Iteration 5751 / 39200) loss: 2.876543
(Iteration 5761 / 39200) loss: 2.827345
(Iteration 5771 / 39200) loss: 2.615447
(Iteration 5781 / 39200) loss: 3.218281
(Iteration 5791 / 39200) loss: 2.827274
(Iteration 5801 / 39200) loss: 2.354421
(Iteration 5811 / 39200) loss: 2.800113
(Iteration 5821 / 39200) loss: 2.825613
(Iteration 5831 / 39200) loss: 2.660153
(Iteration 5841 / 39200) loss: 2.681163
(Iteration 5851 / 39200) loss: 2.710589
(Iteration 5861 / 39200) loss: 2.457250
(Iteration 5871 / 39200) loss: 2.502732
(Epoch 3 / 20) train acc: 0.450000; val_acc: 0.421000
(Iteration 5881 / 39200) loss: 2.668334
(Iteration 5891 / 39200) loss: 2.909033
(Iteration 5901 / 39200) loss: 2.849973
(Iteration 5911 / 39200) loss: 2.714370
(Iteration 5921 / 39200) loss: 2.759452
(Iteration 5931 / 39200) loss: 2.645315
(Iteration 5941 / 39200) loss: 3.033237
(Iteration 5951 / 39200) loss: 2.211707
(Iteration 5961 / 39200) loss: 3.123549
(Iteration 5971 / 39200) loss: 2.469376
(Iteration 5981 / 39200) loss: 2.488938
(Iteration 5991 / 39200) loss: 2.778204
(Iteration 6001 / 39200) loss: 2.668896
(Iteration 6011 / 39200) loss: 2.423245
(Iteration 6021 / 39200) loss: 3.121702
(Iteration 6031 / 39200) loss: 2.755394
(Iteration 6041 / 39200) loss: 2.749637
(Iteration 6051 / 39200) loss: 2.765754
(Iteration 6061 / 39200) loss: 2.799186
(Iteration 6071 / 39200) loss: 3.141227
(Iteration 6081 / 39200) loss: 2.685281
(Iteration 6091 / 39200) loss: 3.153937
(Iteration 6101 / 39200) loss: 2.661197
(Iteration 6111 / 39200) loss: 2.682207
(Iteration 6121 / 39200) loss: 2.821411
(Iteration 6131 / 39200) loss: 3.083307
(Iteration 6141 / 39200) loss: 2.423823
(Iteration 6151 / 39200) loss: 2.831933
(Iteration 6161 / 39200) loss: 2.642734
(Iteration 6171 / 39200) loss: 2.748274
(Iteration 6181 / 39200) loss: 2.774239
(Iteration 6191 / 39200) loss: 2.502963

(Iteration 6201 / 39200) loss: 2.657523
(Iteration 6211 / 39200) loss: 2.982870
(Iteration 6221 / 39200) loss: 2.424063
(Iteration 6231 / 39200) loss: 2.642800
(Iteration 6241 / 39200) loss: 2.846139
(Iteration 6251 / 39200) loss: 3.250602
(Iteration 6261 / 39200) loss: 2.610115
(Iteration 6271 / 39200) loss: 2.660797
(Iteration 6281 / 39200) loss: 2.525308
(Iteration 6291 / 39200) loss: 2.247276
(Iteration 6301 / 39200) loss: 2.654438
(Iteration 6311 / 39200) loss: 2.686503
(Iteration 6321 / 39200) loss: 2.983253
(Iteration 6331 / 39200) loss: 2.450293
(Iteration 6341 / 39200) loss: 2.570784
(Iteration 6351 / 39200) loss: 2.857131
(Iteration 6361 / 39200) loss: 2.654252
(Iteration 6371 / 39200) loss: 3.087453
(Iteration 6381 / 39200) loss: 2.504046
(Iteration 6391 / 39200) loss: 2.746351
(Iteration 6401 / 39200) loss: 2.626774
(Iteration 6411 / 39200) loss: 2.618683
(Iteration 6421 / 39200) loss: 3.091551
(Iteration 6431 / 39200) loss: 2.646409
(Iteration 6441 / 39200) loss: 3.016804
(Iteration 6451 / 39200) loss: 2.641526
(Iteration 6461 / 39200) loss: 2.623544
(Iteration 6471 / 39200) loss: 3.013099
(Iteration 6481 / 39200) loss: 2.696628
(Iteration 6491 / 39200) loss: 2.335746
(Iteration 6501 / 39200) loss: 2.958024
(Iteration 6511 / 39200) loss: 2.762600
(Iteration 6521 / 39200) loss: 2.233195
(Iteration 6531 / 39200) loss: 2.738965
(Iteration 6541 / 39200) loss: 2.522287
(Iteration 6551 / 39200) loss: 2.629154
(Iteration 6561 / 39200) loss: 2.365959
(Iteration 6571 / 39200) loss: 2.975256
(Iteration 6581 / 39200) loss: 2.519547
(Iteration 6591 / 39200) loss: 2.261763
(Iteration 6601 / 39200) loss: 2.777595
(Iteration 6611 / 39200) loss: 2.815921
(Iteration 6621 / 39200) loss: 2.553604
(Iteration 6631 / 39200) loss: 2.702639
(Iteration 6641 / 39200) loss: 2.861336
(Iteration 6651 / 39200) loss: 2.596599
(Iteration 6661 / 39200) loss: 2.514163
(Iteration 6671 / 39200) loss: 2.845718

(Iteration 6681 / 39200) loss: 2.646594
(Iteration 6691 / 39200) loss: 2.744455
(Iteration 6701 / 39200) loss: 2.866370
(Iteration 6711 / 39200) loss: 2.696940
(Iteration 6721 / 39200) loss: 2.533017
(Iteration 6731 / 39200) loss: 3.133445
(Iteration 6741 / 39200) loss: 2.584041
(Iteration 6751 / 39200) loss: 2.350286
(Iteration 6761 / 39200) loss: 3.480707
(Iteration 6771 / 39200) loss: 3.055332
(Iteration 6781 / 39200) loss: 2.476329
(Iteration 6791 / 39200) loss: 2.711625
(Iteration 6801 / 39200) loss: 2.752583
(Iteration 6811 / 39200) loss: 2.405592
(Iteration 6821 / 39200) loss: 2.554411
(Iteration 6831 / 39200) loss: 2.492598
(Iteration 6841 / 39200) loss: 3.095604
(Iteration 6851 / 39200) loss: 2.779596
(Iteration 6861 / 39200) loss: 2.625615
(Iteration 6871 / 39200) loss: 2.574955
(Iteration 6881 / 39200) loss: 2.708611
(Iteration 6891 / 39200) loss: 2.784097
(Iteration 6901 / 39200) loss: 2.665640
(Iteration 6911 / 39200) loss: 2.420119
(Iteration 6921 / 39200) loss: 2.837847
(Iteration 6931 / 39200) loss: 2.786603
(Iteration 6941 / 39200) loss: 2.744938
(Iteration 6951 / 39200) loss: 2.521980
(Iteration 6961 / 39200) loss: 2.231249
(Iteration 6971 / 39200) loss: 2.845528
(Iteration 6981 / 39200) loss: 2.392437
(Iteration 6991 / 39200) loss: 2.697481
(Iteration 7001 / 39200) loss: 2.421294
(Iteration 7011 / 39200) loss: 2.790968
(Iteration 7021 / 39200) loss: 2.389775
(Iteration 7031 / 39200) loss: 2.371238
(Iteration 7041 / 39200) loss: 2.802187
(Iteration 7051 / 39200) loss: 2.531364
(Iteration 7061 / 39200) loss: 2.866401
(Iteration 7071 / 39200) loss: 2.617858
(Iteration 7081 / 39200) loss: 2.957548
(Iteration 7091 / 39200) loss: 3.074583
(Iteration 7101 / 39200) loss: 2.446343
(Iteration 7111 / 39200) loss: 2.770505
(Iteration 7121 / 39200) loss: 2.466934
(Iteration 7131 / 39200) loss: 2.766138
(Iteration 7141 / 39200) loss: 2.603035
(Iteration 7151 / 39200) loss: 2.609371

(Iteration 7161 / 39200) loss: 2.371557
(Iteration 7171 / 39200) loss: 2.660049
(Iteration 7181 / 39200) loss: 2.572105
(Iteration 7191 / 39200) loss: 2.770726
(Iteration 7201 / 39200) loss: 2.760152
(Iteration 7211 / 39200) loss: 2.344936
(Iteration 7221 / 39200) loss: 2.436178
(Iteration 7231 / 39200) loss: 2.515421
(Iteration 7241 / 39200) loss: 2.680224
(Iteration 7251 / 39200) loss: 2.643974
(Iteration 7261 / 39200) loss: 2.685410
(Iteration 7271 / 39200) loss: 2.658667
(Iteration 7281 / 39200) loss: 2.714190
(Iteration 7291 / 39200) loss: 2.588402
(Iteration 7301 / 39200) loss: 2.868094
(Iteration 7311 / 39200) loss: 2.623471
(Iteration 7321 / 39200) loss: 2.671896
(Iteration 7331 / 39200) loss: 2.324403
(Iteration 7341 / 39200) loss: 2.496365
(Iteration 7351 / 39200) loss: 2.671946
(Iteration 7361 / 39200) loss: 2.707845
(Iteration 7371 / 39200) loss: 2.903040
(Iteration 7381 / 39200) loss: 2.707288
(Iteration 7391 / 39200) loss: 2.381962
(Iteration 7401 / 39200) loss: 2.592595
(Iteration 7411 / 39200) loss: 2.566368
(Iteration 7421 / 39200) loss: 2.847622
(Iteration 7431 / 39200) loss: 2.963820
(Iteration 7441 / 39200) loss: 2.565699
(Iteration 7451 / 39200) loss: 2.038674
(Iteration 7461 / 39200) loss: 2.421980
(Iteration 7471 / 39200) loss: 2.290548
(Iteration 7481 / 39200) loss: 2.498001
(Iteration 7491 / 39200) loss: 2.277483
(Iteration 7501 / 39200) loss: 2.469129
(Iteration 7511 / 39200) loss: 2.611782
(Iteration 7521 / 39200) loss: 2.311545
(Iteration 7531 / 39200) loss: 2.536740
(Iteration 7541 / 39200) loss: 2.477740
(Iteration 7551 / 39200) loss: 2.244616
(Iteration 7561 / 39200) loss: 2.651097
(Iteration 7571 / 39200) loss: 2.496937
(Iteration 7581 / 39200) loss: 2.096916
(Iteration 7591 / 39200) loss: 2.430009
(Iteration 7601 / 39200) loss: 2.480169
(Iteration 7611 / 39200) loss: 3.038095
(Iteration 7621 / 39200) loss: 2.485077
(Iteration 7631 / 39200) loss: 2.566776

(Iteration 7641 / 39200) loss: 2.358285
(Iteration 7651 / 39200) loss: 2.503650
(Iteration 7661 / 39200) loss: 2.543253
(Iteration 7671 / 39200) loss: 2.562905
(Iteration 7681 / 39200) loss: 2.660194
(Iteration 7691 / 39200) loss: 2.849635
(Iteration 7701 / 39200) loss: 2.680345
(Iteration 7711 / 39200) loss: 2.571681
(Iteration 7721 / 39200) loss: 2.824140
(Iteration 7731 / 39200) loss: 2.892156
(Iteration 7741 / 39200) loss: 2.274868
(Iteration 7751 / 39200) loss: 3.219013
(Iteration 7761 / 39200) loss: 2.803803
(Iteration 7771 / 39200) loss: 2.565754
(Iteration 7781 / 39200) loss: 2.677617
(Iteration 7791 / 39200) loss: 2.379078
(Iteration 7801 / 39200) loss: 2.378283
(Iteration 7811 / 39200) loss: 2.568295
(Iteration 7821 / 39200) loss: 2.579978
(Iteration 7831 / 39200) loss: 3.060615
(Epoch 4 / 20) train acc: 0.487000; val_acc: 0.480000
(Iteration 7841 / 39200) loss: 2.573188
(Iteration 7851 / 39200) loss: 2.598805
(Iteration 7861 / 39200) loss: 2.527967
(Iteration 7871 / 39200) loss: 2.366985
(Iteration 7881 / 39200) loss: 2.343847
(Iteration 7891 / 39200) loss: 2.626613
(Iteration 7901 / 39200) loss: 2.721923
(Iteration 7911 / 39200) loss: 2.380113
(Iteration 7921 / 39200) loss: 2.454006
(Iteration 7931 / 39200) loss: 2.409186
(Iteration 7941 / 39200) loss: 2.537513
(Iteration 7951 / 39200) loss: 2.848998
(Iteration 7961 / 39200) loss: 2.691312
(Iteration 7971 / 39200) loss: 2.514823
(Iteration 7981 / 39200) loss: 2.425919
(Iteration 7991 / 39200) loss: 2.332192
(Iteration 8001 / 39200) loss: 2.861710
(Iteration 8011 / 39200) loss: 2.626747
(Iteration 8021 / 39200) loss: 2.706984
(Iteration 8031 / 39200) loss: 2.561978
(Iteration 8041 / 39200) loss: 2.659201
(Iteration 8051 / 39200) loss: 2.415073
(Iteration 8061 / 39200) loss: 2.824569
(Iteration 8071 / 39200) loss: 2.377226
(Iteration 8081 / 39200) loss: 2.605318
(Iteration 8091 / 39200) loss: 2.524755
(Iteration 8101 / 39200) loss: 2.260833

(Iteration 8111 / 39200) loss: 2.189439
(Iteration 8121 / 39200) loss: 2.485910
(Iteration 8131 / 39200) loss: 2.354159
(Iteration 8141 / 39200) loss: 2.814896
(Iteration 8151 / 39200) loss: 2.375489
(Iteration 8161 / 39200) loss: 2.467582
(Iteration 8171 / 39200) loss: 2.467411
(Iteration 8181 / 39200) loss: 2.339504
(Iteration 8191 / 39200) loss: 2.403283
(Iteration 8201 / 39200) loss: 2.393150
(Iteration 8211 / 39200) loss: 2.307923
(Iteration 8221 / 39200) loss: 2.715563
(Iteration 8231 / 39200) loss: 2.466740
(Iteration 8241 / 39200) loss: 2.537277
(Iteration 8251 / 39200) loss: 2.542221
(Iteration 8261 / 39200) loss: 2.568313
(Iteration 8271 / 39200) loss: 2.233155
(Iteration 8281 / 39200) loss: 2.699570
(Iteration 8291 / 39200) loss: 2.728091
(Iteration 8301 / 39200) loss: 2.489922
(Iteration 8311 / 39200) loss: 2.447420
(Iteration 8321 / 39200) loss: 2.412616
(Iteration 8331 / 39200) loss: 1.902497
(Iteration 8341 / 39200) loss: 2.820611
(Iteration 8351 / 39200) loss: 2.814948
(Iteration 8361 / 39200) loss: 2.257629
(Iteration 8371 / 39200) loss: 3.010935
(Iteration 8381 / 39200) loss: 2.421739
(Iteration 8391 / 39200) loss: 2.194850
(Iteration 8401 / 39200) loss: 2.861876
(Iteration 8411 / 39200) loss: 2.648215
(Iteration 8421 / 39200) loss: 2.593763
(Iteration 8431 / 39200) loss: 2.160293
(Iteration 8441 / 39200) loss: 2.543811
(Iteration 8451 / 39200) loss: 2.424889
(Iteration 8461 / 39200) loss: 2.713090
(Iteration 8471 / 39200) loss: 2.469929
(Iteration 8481 / 39200) loss: 2.414311
(Iteration 8491 / 39200) loss: 2.347839
(Iteration 8501 / 39200) loss: 2.223453
(Iteration 8511 / 39200) loss: 2.149027
(Iteration 8521 / 39200) loss: 2.298448
(Iteration 8531 / 39200) loss: 2.256700
(Iteration 8541 / 39200) loss: 2.251163
(Iteration 8551 / 39200) loss: 2.149139
(Iteration 8561 / 39200) loss: 2.506242
(Iteration 8571 / 39200) loss: 2.700200
(Iteration 8581 / 39200) loss: 2.464331

(Iteration 8591 / 39200) loss: 2.558391
(Iteration 8601 / 39200) loss: 2.075998
(Iteration 8611 / 39200) loss: 2.454608
(Iteration 8621 / 39200) loss: 2.596545
(Iteration 8631 / 39200) loss: 2.250757
(Iteration 8641 / 39200) loss: 2.304627
(Iteration 8651 / 39200) loss: 2.462649
(Iteration 8661 / 39200) loss: 3.051459
(Iteration 8671 / 39200) loss: 2.205617
(Iteration 8681 / 39200) loss: 2.271658
(Iteration 8691 / 39200) loss: 2.471168
(Iteration 8701 / 39200) loss: 2.725760
(Iteration 8711 / 39200) loss: 2.745636
(Iteration 8721 / 39200) loss: 2.889607
(Iteration 8731 / 39200) loss: 2.918286
(Iteration 8741 / 39200) loss: 2.536131
(Iteration 8751 / 39200) loss: 2.215704
(Iteration 8761 / 39200) loss: 2.658144
(Iteration 8771 / 39200) loss: 2.415534
(Iteration 8781 / 39200) loss: 2.571412
(Iteration 8791 / 39200) loss: 2.419434
(Iteration 8801 / 39200) loss: 2.463124
(Iteration 8811 / 39200) loss: 2.105625
(Iteration 8821 / 39200) loss: 2.261575
(Iteration 8831 / 39200) loss: 2.354943
(Iteration 8841 / 39200) loss: 2.461807
(Iteration 8851 / 39200) loss: 2.553655
(Iteration 8861 / 39200) loss: 2.466487
(Iteration 8871 / 39200) loss: 2.008123
(Iteration 8881 / 39200) loss: 2.330517
(Iteration 8891 / 39200) loss: 2.371704
(Iteration 8901 / 39200) loss: 2.074365
(Iteration 8911 / 39200) loss: 2.068395
(Iteration 8921 / 39200) loss: 2.528511
(Iteration 8931 / 39200) loss: 2.808377
(Iteration 8941 / 39200) loss: 1.893650
(Iteration 8951 / 39200) loss: 2.419501
(Iteration 8961 / 39200) loss: 2.133909
(Iteration 8971 / 39200) loss: 1.916404
(Iteration 8981 / 39200) loss: 2.384985
(Iteration 8991 / 39200) loss: 2.417627
(Iteration 9001 / 39200) loss: 2.318866
(Iteration 9011 / 39200) loss: 2.164169
(Iteration 9021 / 39200) loss: 2.592901
(Iteration 9031 / 39200) loss: 2.486227
(Iteration 9041 / 39200) loss: 2.521779
(Iteration 9051 / 39200) loss: 2.221030
(Iteration 9061 / 39200) loss: 2.372434

(Iteration 9071 / 39200) loss: 1.961553
(Iteration 9081 / 39200) loss: 2.471691
(Iteration 9091 / 39200) loss: 3.174100
(Iteration 9101 / 39200) loss: 2.052427
(Iteration 9111 / 39200) loss: 1.922090
(Iteration 9121 / 39200) loss: 2.673620
(Iteration 9131 / 39200) loss: 2.961316
(Iteration 9141 / 39200) loss: 2.234471
(Iteration 9151 / 39200) loss: 2.582823
(Iteration 9161 / 39200) loss: 2.716674
(Iteration 9171 / 39200) loss: 2.213500
(Iteration 9181 / 39200) loss: 2.662025
(Iteration 9191 / 39200) loss: 2.488928
(Iteration 9201 / 39200) loss: 2.463184
(Iteration 9211 / 39200) loss: 2.165521
(Iteration 9221 / 39200) loss: 2.671915
(Iteration 9231 / 39200) loss: 2.068189
(Iteration 9241 / 39200) loss: 2.262567
(Iteration 9251 / 39200) loss: 2.276029
(Iteration 9261 / 39200) loss: 2.215782
(Iteration 9271 / 39200) loss: 1.917729
(Iteration 9281 / 39200) loss: 2.791116
(Iteration 9291 / 39200) loss: 2.342955
(Iteration 9301 / 39200) loss: 2.859350
(Iteration 9311 / 39200) loss: 2.097621
(Iteration 9321 / 39200) loss: 2.523186
(Iteration 9331 / 39200) loss: 2.551575
(Iteration 9341 / 39200) loss: 2.212461
(Iteration 9351 / 39200) loss: 2.575598
(Iteration 9361 / 39200) loss: 2.301718
(Iteration 9371 / 39200) loss: 2.213458
(Iteration 9381 / 39200) loss: 2.088056
(Iteration 9391 / 39200) loss: 2.541408
(Iteration 9401 / 39200) loss: 2.345035
(Iteration 9411 / 39200) loss: 1.977272
(Iteration 9421 / 39200) loss: 2.779647
(Iteration 9431 / 39200) loss: 2.482730
(Iteration 9441 / 39200) loss: 2.156272
(Iteration 9451 / 39200) loss: 2.222441
(Iteration 9461 / 39200) loss: 2.378169
(Iteration 9471 / 39200) loss: 2.628645
(Iteration 9481 / 39200) loss: 2.399140
(Iteration 9491 / 39200) loss: 2.337222
(Iteration 9501 / 39200) loss: 2.500235
(Iteration 9511 / 39200) loss: 2.388561
(Iteration 9521 / 39200) loss: 2.394518
(Iteration 9531 / 39200) loss: 2.365532
(Iteration 9541 / 39200) loss: 2.669108

(Iteration 9551 / 39200) loss: 2.291899
(Iteration 9561 / 39200) loss: 2.272724
(Iteration 9571 / 39200) loss: 2.619940
(Iteration 9581 / 39200) loss: 2.491473
(Iteration 9591 / 39200) loss: 2.471352
(Iteration 9601 / 39200) loss: 2.488189
(Iteration 9611 / 39200) loss: 2.148966
(Iteration 9621 / 39200) loss: 2.380240
(Iteration 9631 / 39200) loss: 2.359356
(Iteration 9641 / 39200) loss: 2.193709
(Iteration 9651 / 39200) loss: 2.191194
(Iteration 9661 / 39200) loss: 2.574924
(Iteration 9671 / 39200) loss: 2.619658
(Iteration 9681 / 39200) loss: 2.213366
(Iteration 9691 / 39200) loss: 2.721259
(Iteration 9701 / 39200) loss: 2.398499
(Iteration 9711 / 39200) loss: 2.551730
(Iteration 9721 / 39200) loss: 2.285205
(Iteration 9731 / 39200) loss: 2.255513
(Iteration 9741 / 39200) loss: 2.480795
(Iteration 9751 / 39200) loss: 2.441548
(Iteration 9761 / 39200) loss: 2.238620
(Iteration 9771 / 39200) loss: 2.795348
(Iteration 9781 / 39200) loss: 2.161564
(Iteration 9791 / 39200) loss: 2.120796
(Epoch 5 / 20) train acc: 0.522000; val_acc: 0.472000
(Iteration 9801 / 39200) loss: 2.819057
(Iteration 9811 / 39200) loss: 2.730790
(Iteration 9821 / 39200) loss: 1.956394
(Iteration 9831 / 39200) loss: 2.754986
(Iteration 9841 / 39200) loss: 2.379285
(Iteration 9851 / 39200) loss: 2.228479
(Iteration 9861 / 39200) loss: 2.312474
(Iteration 9871 / 39200) loss: 2.258245
(Iteration 9881 / 39200) loss: 2.197367
(Iteration 9891 / 39200) loss: 2.447372
(Iteration 9901 / 39200) loss: 2.442402
(Iteration 9911 / 39200) loss: 2.314949
(Iteration 9921 / 39200) loss: 2.293200
(Iteration 9931 / 39200) loss: 3.060301
(Iteration 9941 / 39200) loss: 2.582249
(Iteration 9951 / 39200) loss: 2.456840
(Iteration 9961 / 39200) loss: 2.260215
(Iteration 9971 / 39200) loss: 2.100689
(Iteration 9981 / 39200) loss: 2.822918
(Iteration 9991 / 39200) loss: 1.993851
(Iteration 10001 / 39200) loss: 2.031787
(Iteration 10011 / 39200) loss: 2.486383

(Iteration 10021 / 39200) loss: 2.469418
(Iteration 10031 / 39200) loss: 2.297340
(Iteration 10041 / 39200) loss: 2.699611
(Iteration 10051 / 39200) loss: 2.191359
(Iteration 10061 / 39200) loss: 2.124976
(Iteration 10071 / 39200) loss: 2.383719
(Iteration 10081 / 39200) loss: 2.305735
(Iteration 10091 / 39200) loss: 2.314705
(Iteration 10101 / 39200) loss: 2.124708
(Iteration 10111 / 39200) loss: 2.011005
(Iteration 10121 / 39200) loss: 2.403115
(Iteration 10131 / 39200) loss: 1.855214
(Iteration 10141 / 39200) loss: 2.323552
(Iteration 10151 / 39200) loss: 2.097708
(Iteration 10161 / 39200) loss: 2.280211
(Iteration 10171 / 39200) loss: 2.405678
(Iteration 10181 / 39200) loss: 2.392250
(Iteration 10191 / 39200) loss: 2.280255
(Iteration 10201 / 39200) loss: 2.440027
(Iteration 10211 / 39200) loss: 2.454421
(Iteration 10221 / 39200) loss: 2.208170
(Iteration 10231 / 39200) loss: 2.548241
(Iteration 10241 / 39200) loss: 2.163393
(Iteration 10251 / 39200) loss: 2.924408
(Iteration 10261 / 39200) loss: 2.235803
(Iteration 10271 / 39200) loss: 2.675099
(Iteration 10281 / 39200) loss: 2.266829
(Iteration 10291 / 39200) loss: 2.227034
(Iteration 10301 / 39200) loss: 1.739107
(Iteration 10311 / 39200) loss: 2.321009
(Iteration 10321 / 39200) loss: 2.389363
(Iteration 10331 / 39200) loss: 1.903742
(Iteration 10341 / 39200) loss: 2.608549
(Iteration 10351 / 39200) loss: 2.341280
(Iteration 10361 / 39200) loss: 2.412627
(Iteration 10371 / 39200) loss: 2.038562
(Iteration 10381 / 39200) loss: 2.120138
(Iteration 10391 / 39200) loss: 2.351252
(Iteration 10401 / 39200) loss: 2.060132
(Iteration 10411 / 39200) loss: 2.061017
(Iteration 10421 / 39200) loss: 2.211220
(Iteration 10431 / 39200) loss: 1.938460
(Iteration 10441 / 39200) loss: 2.547851
(Iteration 10451 / 39200) loss: 2.104190
(Iteration 10461 / 39200) loss: 1.977515
(Iteration 10471 / 39200) loss: 2.542133
(Iteration 10481 / 39200) loss: 2.026902
(Iteration 10491 / 39200) loss: 2.573861

(Iteration 10501 / 39200) loss: 2.246087
(Iteration 10511 / 39200) loss: 2.146558
(Iteration 10521 / 39200) loss: 2.250003
(Iteration 10531 / 39200) loss: 2.051625
(Iteration 10541 / 39200) loss: 2.039838
(Iteration 10551 / 39200) loss: 2.177132
(Iteration 10561 / 39200) loss: 2.121721
(Iteration 10571 / 39200) loss: 1.983034
(Iteration 10581 / 39200) loss: 2.025562
(Iteration 10591 / 39200) loss: 2.908348
(Iteration 10601 / 39200) loss: 2.122267
(Iteration 10611 / 39200) loss: 2.619896
(Iteration 10621 / 39200) loss: 2.554846
(Iteration 10631 / 39200) loss: 2.062008
(Iteration 10641 / 39200) loss: 2.475151
(Iteration 10651 / 39200) loss: 2.147280
(Iteration 10661 / 39200) loss: 2.116235
(Iteration 10671 / 39200) loss: 2.207772
(Iteration 10681 / 39200) loss: 2.101234
(Iteration 10691 / 39200) loss: 1.938854
(Iteration 10701 / 39200) loss: 2.652117
(Iteration 10711 / 39200) loss: 2.372080
(Iteration 10721 / 39200) loss: 2.582823
(Iteration 10731 / 39200) loss: 2.442888
(Iteration 10741 / 39200) loss: 2.312263
(Iteration 10751 / 39200) loss: 1.952181
(Iteration 10761 / 39200) loss: 2.138038
(Iteration 10771 / 39200) loss: 2.559653
(Iteration 10781 / 39200) loss: 2.390861
(Iteration 10791 / 39200) loss: 1.933547
(Iteration 10801 / 39200) loss: 2.162065
(Iteration 10811 / 39200) loss: 1.962591
(Iteration 10821 / 39200) loss: 2.206300
(Iteration 10831 / 39200) loss: 2.115310
(Iteration 10841 / 39200) loss: 2.273006
(Iteration 10851 / 39200) loss: 2.464562
(Iteration 10861 / 39200) loss: 2.651306
(Iteration 10871 / 39200) loss: 2.260605
(Iteration 10881 / 39200) loss: 2.168314
(Iteration 10891 / 39200) loss: 1.964662
(Iteration 10901 / 39200) loss: 2.336279
(Iteration 10911 / 39200) loss: 2.185087
(Iteration 10921 / 39200) loss: 2.401708
(Iteration 10931 / 39200) loss: 2.634674
(Iteration 10941 / 39200) loss: 2.183968
(Iteration 10951 / 39200) loss: 2.188523
(Iteration 10961 / 39200) loss: 2.142482
(Iteration 10971 / 39200) loss: 2.064777

(Iteration 10981 / 39200) loss: 2.416409
(Iteration 10991 / 39200) loss: 2.110485
(Iteration 11001 / 39200) loss: 2.380778
(Iteration 11011 / 39200) loss: 2.553457
(Iteration 11021 / 39200) loss: 2.400458
(Iteration 11031 / 39200) loss: 2.129037
(Iteration 11041 / 39200) loss: 2.195815
(Iteration 11051 / 39200) loss: 2.517239
(Iteration 11061 / 39200) loss: 2.607149
(Iteration 11071 / 39200) loss: 2.058160
(Iteration 11081 / 39200) loss: 2.532297
(Iteration 11091 / 39200) loss: 2.308832
(Iteration 11101 / 39200) loss: 2.321974
(Iteration 11111 / 39200) loss: 2.054633
(Iteration 11121 / 39200) loss: 2.418233
(Iteration 11131 / 39200) loss: 2.229016
(Iteration 11141 / 39200) loss: 2.606355
(Iteration 11151 / 39200) loss: 2.073194
(Iteration 11161 / 39200) loss: 2.123998
(Iteration 11171 / 39200) loss: 2.198915
(Iteration 11181 / 39200) loss: 2.211760
(Iteration 11191 / 39200) loss: 2.809042
(Iteration 11201 / 39200) loss: 1.974030
(Iteration 11211 / 39200) loss: 1.996612
(Iteration 11221 / 39200) loss: 2.180436
(Iteration 11231 / 39200) loss: 2.297965
(Iteration 11241 / 39200) loss: 2.399915
(Iteration 11251 / 39200) loss: 2.243537
(Iteration 11261 / 39200) loss: 2.178591
(Iteration 11271 / 39200) loss: 2.383759
(Iteration 11281 / 39200) loss: 2.461915
(Iteration 11291 / 39200) loss: 1.998913
(Iteration 11301 / 39200) loss: 2.855732
(Iteration 11311 / 39200) loss: 2.327346
(Iteration 11321 / 39200) loss: 2.557566
(Iteration 11331 / 39200) loss: 2.996914
(Iteration 11341 / 39200) loss: 2.163900
(Iteration 11351 / 39200) loss: 2.049819
(Iteration 11361 / 39200) loss: 1.933901
(Iteration 11371 / 39200) loss: 2.368062
(Iteration 11381 / 39200) loss: 2.378727
(Iteration 11391 / 39200) loss: 2.300680
(Iteration 11401 / 39200) loss: 2.078596
(Iteration 11411 / 39200) loss: 2.923256
(Iteration 11421 / 39200) loss: 2.162627
(Iteration 11431 / 39200) loss: 2.018192
(Iteration 11441 / 39200) loss: 1.944244
(Iteration 11451 / 39200) loss: 2.101246

(Iteration 11461 / 39200) loss: 1.920115
(Iteration 11471 / 39200) loss: 2.109096
(Iteration 11481 / 39200) loss: 2.391211
(Iteration 11491 / 39200) loss: 2.242583
(Iteration 11501 / 39200) loss: 2.282538
(Iteration 11511 / 39200) loss: 2.575959
(Iteration 11521 / 39200) loss: 2.280819
(Iteration 11531 / 39200) loss: 2.250221
(Iteration 11541 / 39200) loss: 2.110986
(Iteration 11551 / 39200) loss: 2.071874
(Iteration 11561 / 39200) loss: 2.520599
(Iteration 11571 / 39200) loss: 2.031180
(Iteration 11581 / 39200) loss: 2.296421
(Iteration 11591 / 39200) loss: 2.442326
(Iteration 11601 / 39200) loss: 2.469439
(Iteration 11611 / 39200) loss: 2.259272
(Iteration 11621 / 39200) loss: 2.535157
(Iteration 11631 / 39200) loss: 2.334855
(Iteration 11641 / 39200) loss: 1.973957
(Iteration 11651 / 39200) loss: 2.413676
(Iteration 11661 / 39200) loss: 2.269406
(Iteration 11671 / 39200) loss: 2.415601
(Iteration 11681 / 39200) loss: 2.206889
(Iteration 11691 / 39200) loss: 2.376594
(Iteration 11701 / 39200) loss: 2.191371
(Iteration 11711 / 39200) loss: 2.339769
(Iteration 11721 / 39200) loss: 2.166804
(Iteration 11731 / 39200) loss: 2.114427
(Iteration 11741 / 39200) loss: 2.056124
(Iteration 11751 / 39200) loss: 1.853031
(Epoch 6 / 20) train acc: 0.521000; val_acc: 0.475000
(Iteration 11761 / 39200) loss: 2.183754
(Iteration 11771 / 39200) loss: 2.064949
(Iteration 11781 / 39200) loss: 1.989651
(Iteration 11791 / 39200) loss: 2.183784
(Iteration 11801 / 39200) loss: 2.262922
(Iteration 11811 / 39200) loss: 2.371148
(Iteration 11821 / 39200) loss: 2.206720
(Iteration 11831 / 39200) loss: 2.476363
(Iteration 11841 / 39200) loss: 2.100311
(Iteration 11851 / 39200) loss: 2.590260
(Iteration 11861 / 39200) loss: 1.789166
(Iteration 11871 / 39200) loss: 2.351535
(Iteration 11881 / 39200) loss: 2.117275
(Iteration 11891 / 39200) loss: 2.438939
(Iteration 11901 / 39200) loss: 2.344012
(Iteration 11911 / 39200) loss: 2.935177
(Iteration 11921 / 39200) loss: 2.407169

(Iteration 11931 / 39200) loss: 2.110079
(Iteration 11941 / 39200) loss: 2.098883
(Iteration 11951 / 39200) loss: 1.996804
(Iteration 11961 / 39200) loss: 1.968058
(Iteration 11971 / 39200) loss: 2.426577
(Iteration 11981 / 39200) loss: 2.304426
(Iteration 11991 / 39200) loss: 2.350690
(Iteration 12001 / 39200) loss: 2.235541
(Iteration 12011 / 39200) loss: 1.986338
(Iteration 12021 / 39200) loss: 1.933956
(Iteration 12031 / 39200) loss: 2.239341
(Iteration 12041 / 39200) loss: 2.200832
(Iteration 12051 / 39200) loss: 2.472926
(Iteration 12061 / 39200) loss: 2.111117
(Iteration 12071 / 39200) loss: 2.017549
(Iteration 12081 / 39200) loss: 1.826108
(Iteration 12091 / 39200) loss: 2.375447
(Iteration 12101 / 39200) loss: 1.809831
(Iteration 12111 / 39200) loss: 2.146012
(Iteration 12121 / 39200) loss: 1.843145
(Iteration 12131 / 39200) loss: 2.011979
(Iteration 12141 / 39200) loss: 2.011912
(Iteration 12151 / 39200) loss: 2.138161
(Iteration 12161 / 39200) loss: 2.191328
(Iteration 12171 / 39200) loss: 2.619411
(Iteration 12181 / 39200) loss: 2.449876
(Iteration 12191 / 39200) loss: 2.346865
(Iteration 12201 / 39200) loss: 1.894530
(Iteration 12211 / 39200) loss: 1.647886
(Iteration 12221 / 39200) loss: 1.836192
(Iteration 12231 / 39200) loss: 2.116728
(Iteration 12241 / 39200) loss: 1.981321
(Iteration 12251 / 39200) loss: 2.346453
(Iteration 12261 / 39200) loss: 2.254821
(Iteration 12271 / 39200) loss: 2.503716
(Iteration 12281 / 39200) loss: 2.537448
(Iteration 12291 / 39200) loss: 1.747947
(Iteration 12301 / 39200) loss: 2.102249
(Iteration 12311 / 39200) loss: 2.189152
(Iteration 12321 / 39200) loss: 2.276080
(Iteration 12331 / 39200) loss: 2.223010
(Iteration 12341 / 39200) loss: 2.219722
(Iteration 12351 / 39200) loss: 1.699555
(Iteration 12361 / 39200) loss: 1.901078
(Iteration 12371 / 39200) loss: 2.145378
(Iteration 12381 / 39200) loss: 2.273085
(Iteration 12391 / 39200) loss: 2.092568
(Iteration 12401 / 39200) loss: 2.067318

(Iteration 12411 / 39200) loss: 2.155029
(Iteration 12421 / 39200) loss: 2.250706
(Iteration 12431 / 39200) loss: 2.065350
(Iteration 12441 / 39200) loss: 2.132094
(Iteration 12451 / 39200) loss: 2.191811
(Iteration 12461 / 39200) loss: 2.274337
(Iteration 12471 / 39200) loss: 2.027348
(Iteration 12481 / 39200) loss: 1.825791
(Iteration 12491 / 39200) loss: 1.770523
(Iteration 12501 / 39200) loss: 2.183671
(Iteration 12511 / 39200) loss: 1.983136
(Iteration 12521 / 39200) loss: 2.091772
(Iteration 12531 / 39200) loss: 2.371155
(Iteration 12541 / 39200) loss: 2.078261
(Iteration 12551 / 39200) loss: 2.138595
(Iteration 12561 / 39200) loss: 2.055967
(Iteration 12571 / 39200) loss: 2.383604
(Iteration 12581 / 39200) loss: 2.168647
(Iteration 12591 / 39200) loss: 1.848765
(Iteration 12601 / 39200) loss: 2.178242
(Iteration 12611 / 39200) loss: 2.169175
(Iteration 12621 / 39200) loss: 2.090065
(Iteration 12631 / 39200) loss: 2.020986
(Iteration 12641 / 39200) loss: 2.229780
(Iteration 12651 / 39200) loss: 2.133612
(Iteration 12661 / 39200) loss: 1.859364
(Iteration 12671 / 39200) loss: 1.872412
(Iteration 12681 / 39200) loss: 1.999912
(Iteration 12691 / 39200) loss: 2.185408
(Iteration 12701 / 39200) loss: 2.574626
(Iteration 12711 / 39200) loss: 2.356514
(Iteration 12721 / 39200) loss: 2.931773
(Iteration 12731 / 39200) loss: 2.242151
(Iteration 12741 / 39200) loss: 2.621191
(Iteration 12751 / 39200) loss: 2.100122
(Iteration 12761 / 39200) loss: 2.112291
(Iteration 12771 / 39200) loss: 2.129900
(Iteration 12781 / 39200) loss: 2.250028
(Iteration 12791 / 39200) loss: 2.061080
(Iteration 12801 / 39200) loss: 2.374449
(Iteration 12811 / 39200) loss: 2.004857
(Iteration 12821 / 39200) loss: 1.962927
(Iteration 12831 / 39200) loss: 2.250184
(Iteration 12841 / 39200) loss: 1.821917
(Iteration 12851 / 39200) loss: 2.518806
(Iteration 12861 / 39200) loss: 2.460087
(Iteration 12871 / 39200) loss: 1.876338
(Iteration 12881 / 39200) loss: 2.183147

(Iteration 12891 / 39200) loss: 2.283456
(Iteration 12901 / 39200) loss: 2.136390
(Iteration 12911 / 39200) loss: 1.912424
(Iteration 12921 / 39200) loss: 2.378505
(Iteration 12931 / 39200) loss: 2.672515
(Iteration 12941 / 39200) loss: 2.103649
(Iteration 12951 / 39200) loss: 1.885236
(Iteration 12961 / 39200) loss: 2.064822
(Iteration 12971 / 39200) loss: 1.662964
(Iteration 12981 / 39200) loss: 2.021259
(Iteration 12991 / 39200) loss: 2.577039
(Iteration 13001 / 39200) loss: 1.753629
(Iteration 13011 / 39200) loss: 1.970683
(Iteration 13021 / 39200) loss: 1.832723
(Iteration 13031 / 39200) loss: 2.331968
(Iteration 13041 / 39200) loss: 2.244694
(Iteration 13051 / 39200) loss: 2.002822
(Iteration 13061 / 39200) loss: 2.014039
(Iteration 13071 / 39200) loss: 2.825802
(Iteration 13081 / 39200) loss: 2.272199
(Iteration 13091 / 39200) loss: 1.742167
(Iteration 13101 / 39200) loss: 2.175814
(Iteration 13111 / 39200) loss: 2.302833
(Iteration 13121 / 39200) loss: 2.065362
(Iteration 13131 / 39200) loss: 2.140789
(Iteration 13141 / 39200) loss: 2.083388
(Iteration 13151 / 39200) loss: 2.369719
(Iteration 13161 / 39200) loss: 1.988840
(Iteration 13171 / 39200) loss: 2.095541
(Iteration 13181 / 39200) loss: 2.067730
(Iteration 13191 / 39200) loss: 2.232650
(Iteration 13201 / 39200) loss: 3.015685
(Iteration 13211 / 39200) loss: 2.299518
(Iteration 13221 / 39200) loss: 2.083868
(Iteration 13231 / 39200) loss: 1.991728
(Iteration 13241 / 39200) loss: 2.053432
(Iteration 13251 / 39200) loss: 2.429199
(Iteration 13261 / 39200) loss: 2.178654
(Iteration 13271 / 39200) loss: 1.965169
(Iteration 13281 / 39200) loss: 2.249431
(Iteration 13291 / 39200) loss: 1.616305
(Iteration 13301 / 39200) loss: 2.209365
(Iteration 13311 / 39200) loss: 1.929445
(Iteration 13321 / 39200) loss: 2.416815
(Iteration 13331 / 39200) loss: 1.796179
(Iteration 13341 / 39200) loss: 2.510386
(Iteration 13351 / 39200) loss: 1.959490
(Iteration 13361 / 39200) loss: 2.031776

(Iteration 13371 / 39200) loss: 2.221103
(Iteration 13381 / 39200) loss: 2.019954
(Iteration 13391 / 39200) loss: 2.035281
(Iteration 13401 / 39200) loss: 1.980452
(Iteration 13411 / 39200) loss: 1.813026
(Iteration 13421 / 39200) loss: 2.258929
(Iteration 13431 / 39200) loss: 1.972858
(Iteration 13441 / 39200) loss: 1.832189
(Iteration 13451 / 39200) loss: 2.778013
(Iteration 13461 / 39200) loss: 1.979475
(Iteration 13471 / 39200) loss: 2.018657
(Iteration 13481 / 39200) loss: 2.398355
(Iteration 13491 / 39200) loss: 2.067139
(Iteration 13501 / 39200) loss: 1.971369
(Iteration 13511 / 39200) loss: 2.159128
(Iteration 13521 / 39200) loss: 1.578726
(Iteration 13531 / 39200) loss: 2.131416
(Iteration 13541 / 39200) loss: 2.129190
(Iteration 13551 / 39200) loss: 2.096905
(Iteration 13561 / 39200) loss: 2.049799
(Iteration 13571 / 39200) loss: 1.945283
(Iteration 13581 / 39200) loss: 1.943347
(Iteration 13591 / 39200) loss: 2.161664
(Iteration 13601 / 39200) loss: 2.490574
(Iteration 13611 / 39200) loss: 1.813437
(Iteration 13621 / 39200) loss: 1.706981
(Iteration 13631 / 39200) loss: 2.074765
(Iteration 13641 / 39200) loss: 1.982715
(Iteration 13651 / 39200) loss: 2.236186
(Iteration 13661 / 39200) loss: 2.091583
(Iteration 13671 / 39200) loss: 1.798710
(Iteration 13681 / 39200) loss: 2.088790
(Iteration 13691 / 39200) loss: 2.042577
(Iteration 13701 / 39200) loss: 1.789449
(Iteration 13711 / 39200) loss: 2.399644
(Epoch 7 / 20) train acc: 0.505000; val_acc: 0.505000
(Iteration 13721 / 39200) loss: 1.758089
(Iteration 13731 / 39200) loss: 1.868556
(Iteration 13741 / 39200) loss: 2.077669
(Iteration 13751 / 39200) loss: 1.949753
(Iteration 13761 / 39200) loss: 1.982642
(Iteration 13771 / 39200) loss: 1.984655
(Iteration 13781 / 39200) loss: 1.730682
(Iteration 13791 / 39200) loss: 1.871884
(Iteration 13801 / 39200) loss: 2.207342
(Iteration 13811 / 39200) loss: 2.246092
(Iteration 13821 / 39200) loss: 2.100259
(Iteration 13831 / 39200) loss: 2.001115

(Iteration 13841 / 39200) loss: 1.888329
(Iteration 13851 / 39200) loss: 1.814264
(Iteration 13861 / 39200) loss: 1.903598
(Iteration 13871 / 39200) loss: 1.947700
(Iteration 13881 / 39200) loss: 2.025878
(Iteration 13891 / 39200) loss: 1.922518
(Iteration 13901 / 39200) loss: 2.138189
(Iteration 13911 / 39200) loss: 1.761928
(Iteration 13921 / 39200) loss: 1.976592
(Iteration 13931 / 39200) loss: 2.151915
(Iteration 13941 / 39200) loss: 2.069920
(Iteration 13951 / 39200) loss: 2.301074
(Iteration 13961 / 39200) loss: 2.361071
(Iteration 13971 / 39200) loss: 2.133827
(Iteration 13981 / 39200) loss: 2.264046
(Iteration 13991 / 39200) loss: 2.102147
(Iteration 14001 / 39200) loss: 1.761046
(Iteration 14011 / 39200) loss: 2.128468
(Iteration 14021 / 39200) loss: 1.918153
(Iteration 14031 / 39200) loss: 2.249012
(Iteration 14041 / 39200) loss: 2.074116
(Iteration 14051 / 39200) loss: 2.021382
(Iteration 14061 / 39200) loss: 2.352940
(Iteration 14071 / 39200) loss: 1.998361
(Iteration 14081 / 39200) loss: 1.803170
(Iteration 14091 / 39200) loss: 1.920125
(Iteration 14101 / 39200) loss: 1.648256
(Iteration 14111 / 39200) loss: 1.641316
(Iteration 14121 / 39200) loss: 1.834128
(Iteration 14131 / 39200) loss: 2.570197
(Iteration 14141 / 39200) loss: 2.128363
(Iteration 14151 / 39200) loss: 1.738554
(Iteration 14161 / 39200) loss: 2.147724
(Iteration 14171 / 39200) loss: 2.228551
(Iteration 14181 / 39200) loss: 2.063014
(Iteration 14191 / 39200) loss: 1.954724
(Iteration 14201 / 39200) loss: 2.242721
(Iteration 14211 / 39200) loss: 2.118454
(Iteration 14221 / 39200) loss: 1.972155
(Iteration 14231 / 39200) loss: 2.149137
(Iteration 14241 / 39200) loss: 1.812358
(Iteration 14251 / 39200) loss: 2.093795
(Iteration 14261 / 39200) loss: 2.248510
(Iteration 14271 / 39200) loss: 2.045167
(Iteration 14281 / 39200) loss: 1.993761
(Iteration 14291 / 39200) loss: 2.141648
(Iteration 14301 / 39200) loss: 2.596380
(Iteration 14311 / 39200) loss: 2.325915

(Iteration 14321 / 39200) loss: 2.019538
(Iteration 14331 / 39200) loss: 1.613202
(Iteration 14341 / 39200) loss: 1.745516
(Iteration 14351 / 39200) loss: 1.902500
(Iteration 14361 / 39200) loss: 1.811986
(Iteration 14371 / 39200) loss: 1.981543
(Iteration 14381 / 39200) loss: 1.737026
(Iteration 14391 / 39200) loss: 2.458708
(Iteration 14401 / 39200) loss: 2.008342
(Iteration 14411 / 39200) loss: 2.341583
(Iteration 14421 / 39200) loss: 1.829179
(Iteration 14431 / 39200) loss: 2.333175
(Iteration 14441 / 39200) loss: 1.757585
(Iteration 14451 / 39200) loss: 2.115194
(Iteration 14461 / 39200) loss: 2.175323
(Iteration 14471 / 39200) loss: 2.035252
(Iteration 14481 / 39200) loss: 1.931879
(Iteration 14491 / 39200) loss: 2.484457
(Iteration 14501 / 39200) loss: 2.302595
(Iteration 14511 / 39200) loss: 2.559579
(Iteration 14521 / 39200) loss: 2.298297
(Iteration 14531 / 39200) loss: 1.732433
(Iteration 14541 / 39200) loss: 2.023327
(Iteration 14551 / 39200) loss: 2.516704
(Iteration 14561 / 39200) loss: 1.974922
(Iteration 14571 / 39200) loss: 2.366679
(Iteration 14581 / 39200) loss: 2.054359
(Iteration 14591 / 39200) loss: 1.738056
(Iteration 14601 / 39200) loss: 1.983100
(Iteration 14611 / 39200) loss: 2.316970
(Iteration 14621 / 39200) loss: 1.788113
(Iteration 14631 / 39200) loss: 1.704349
(Iteration 14641 / 39200) loss: 1.680830
(Iteration 14651 / 39200) loss: 2.254556
(Iteration 14661 / 39200) loss: 1.923409
(Iteration 14671 / 39200) loss: 2.421421
(Iteration 14681 / 39200) loss: 2.056365
(Iteration 14691 / 39200) loss: 1.935778
(Iteration 14701 / 39200) loss: 2.044854
(Iteration 14711 / 39200) loss: 2.115342
(Iteration 14721 / 39200) loss: 1.976036
(Iteration 14731 / 39200) loss: 1.971156
(Iteration 14741 / 39200) loss: 1.785962
(Iteration 14751 / 39200) loss: 2.995541
(Iteration 14761 / 39200) loss: 1.919564
(Iteration 14771 / 39200) loss: 1.796197
(Iteration 14781 / 39200) loss: 2.211154
(Iteration 14791 / 39200) loss: 2.012502

(Iteration 14801 / 39200) loss: 1.601968
(Iteration 14811 / 39200) loss: 2.270615
(Iteration 14821 / 39200) loss: 1.840121
(Iteration 14831 / 39200) loss: 2.062918
(Iteration 14841 / 39200) loss: 1.681429
(Iteration 14851 / 39200) loss: 2.105611
(Iteration 14861 / 39200) loss: 1.774160
(Iteration 14871 / 39200) loss: 1.879702
(Iteration 14881 / 39200) loss: 2.004356
(Iteration 14891 / 39200) loss: 1.850287
(Iteration 14901 / 39200) loss: 1.996645
(Iteration 14911 / 39200) loss: 1.995249
(Iteration 14921 / 39200) loss: 2.020445
(Iteration 14931 / 39200) loss: 2.202872
(Iteration 14941 / 39200) loss: 2.059755
(Iteration 14951 / 39200) loss: 2.339906
(Iteration 14961 / 39200) loss: 2.222233
(Iteration 14971 / 39200) loss: 2.189052
(Iteration 14981 / 39200) loss: 2.101242
(Iteration 14991 / 39200) loss: 1.762569
(Iteration 15001 / 39200) loss: 1.899461
(Iteration 15011 / 39200) loss: 2.285626
(Iteration 15021 / 39200) loss: 2.007317
(Iteration 15031 / 39200) loss: 2.100845
(Iteration 15041 / 39200) loss: 2.031018
(Iteration 15051 / 39200) loss: 1.989989
(Iteration 15061 / 39200) loss: 1.729693
(Iteration 15071 / 39200) loss: 2.042860
(Iteration 15081 / 39200) loss: 1.671689
(Iteration 15091 / 39200) loss: 2.025983
(Iteration 15101 / 39200) loss: 2.046975
(Iteration 15111 / 39200) loss: 1.975893
(Iteration 15121 / 39200) loss: 2.197533
(Iteration 15131 / 39200) loss: 1.472539
(Iteration 15141 / 39200) loss: 2.075587
(Iteration 15151 / 39200) loss: 2.164247
(Iteration 15161 / 39200) loss: 2.341729
(Iteration 15171 / 39200) loss: 1.786902
(Iteration 15181 / 39200) loss: 2.256444
(Iteration 15191 / 39200) loss: 1.893831
(Iteration 15201 / 39200) loss: 1.621501
(Iteration 15211 / 39200) loss: 2.052240
(Iteration 15221 / 39200) loss: 1.830942
(Iteration 15231 / 39200) loss: 2.822931
(Iteration 15241 / 39200) loss: 2.055256
(Iteration 15251 / 39200) loss: 1.948337
(Iteration 15261 / 39200) loss: 1.932063
(Iteration 15271 / 39200) loss: 1.927095

(Iteration 15281 / 39200) loss: 1.589273
(Iteration 15291 / 39200) loss: 2.021475
(Iteration 15301 / 39200) loss: 1.791996
(Iteration 15311 / 39200) loss: 1.988196
(Iteration 15321 / 39200) loss: 1.965225
(Iteration 15331 / 39200) loss: 1.780707
(Iteration 15341 / 39200) loss: 1.959866
(Iteration 15351 / 39200) loss: 2.278212
(Iteration 15361 / 39200) loss: 2.017349
(Iteration 15371 / 39200) loss: 1.657416
(Iteration 15381 / 39200) loss: 1.592996
(Iteration 15391 / 39200) loss: 2.792407
(Iteration 15401 / 39200) loss: 2.129446
(Iteration 15411 / 39200) loss: 1.897811
(Iteration 15421 / 39200) loss: 2.515520
(Iteration 15431 / 39200) loss: 1.720452
(Iteration 15441 / 39200) loss: 1.813051
(Iteration 15451 / 39200) loss: 1.913193
(Iteration 15461 / 39200) loss: 2.030240
(Iteration 15471 / 39200) loss: 2.182297
(Iteration 15481 / 39200) loss: 2.147777
(Iteration 15491 / 39200) loss: 1.823104
(Iteration 15501 / 39200) loss: 1.994858
(Iteration 15511 / 39200) loss: 1.553133
(Iteration 15521 / 39200) loss: 2.284959
(Iteration 15531 / 39200) loss: 1.679034
(Iteration 15541 / 39200) loss: 2.441835
(Iteration 15551 / 39200) loss: 2.100288
(Iteration 15561 / 39200) loss: 2.456166
(Iteration 15571 / 39200) loss: 2.495620
(Iteration 15581 / 39200) loss: 1.686178
(Iteration 15591 / 39200) loss: 2.107136
(Iteration 15601 / 39200) loss: 2.147532
(Iteration 15611 / 39200) loss: 1.987072
(Iteration 15621 / 39200) loss: 2.022341
(Iteration 15631 / 39200) loss: 2.117045
(Iteration 15641 / 39200) loss: 1.798693
(Iteration 15651 / 39200) loss: 1.741242
(Iteration 15661 / 39200) loss: 2.161896
(Iteration 15671 / 39200) loss: 2.370432
(Epoch 8 / 20) train acc: 0.545000; val_acc: 0.501000
(Iteration 15681 / 39200) loss: 2.018251
(Iteration 15691 / 39200) loss: 2.104233
(Iteration 15701 / 39200) loss: 1.997826
(Iteration 15711 / 39200) loss: 1.986868
(Iteration 15721 / 39200) loss: 2.069130
(Iteration 15731 / 39200) loss: 1.987406
(Iteration 15741 / 39200) loss: 1.962072

(Iteration 15751 / 39200) loss: 1.874096
(Iteration 15761 / 39200) loss: 1.488146
(Iteration 15771 / 39200) loss: 1.547350
(Iteration 15781 / 39200) loss: 2.079222
(Iteration 15791 / 39200) loss: 2.391991
(Iteration 15801 / 39200) loss: 1.792794
(Iteration 15811 / 39200) loss: 2.016421
(Iteration 15821 / 39200) loss: 1.762115
(Iteration 15831 / 39200) loss: 2.117732
(Iteration 15841 / 39200) loss: 1.563599
(Iteration 15851 / 39200) loss: 1.778636
(Iteration 15861 / 39200) loss: 1.917868
(Iteration 15871 / 39200) loss: 1.988754
(Iteration 15881 / 39200) loss: 2.166223
(Iteration 15891 / 39200) loss: 1.793950
(Iteration 15901 / 39200) loss: 1.738225
(Iteration 15911 / 39200) loss: 2.202928
(Iteration 15921 / 39200) loss: 2.065464
(Iteration 15931 / 39200) loss: 2.254217
(Iteration 15941 / 39200) loss: 1.878958
(Iteration 15951 / 39200) loss: 1.955980
(Iteration 15961 / 39200) loss: 1.940847
(Iteration 15971 / 39200) loss: 1.645834
(Iteration 15981 / 39200) loss: 1.707878
(Iteration 15991 / 39200) loss: 2.132786
(Iteration 16001 / 39200) loss: 2.156939
(Iteration 16011 / 39200) loss: 2.337906
(Iteration 16021 / 39200) loss: 2.100709
(Iteration 16031 / 39200) loss: 2.096398
(Iteration 16041 / 39200) loss: 2.152412
(Iteration 16051 / 39200) loss: 1.803597
(Iteration 16061 / 39200) loss: 1.615644
(Iteration 16071 / 39200) loss: 2.500628
(Iteration 16081 / 39200) loss: 1.756353
(Iteration 16091 / 39200) loss: 1.788568
(Iteration 16101 / 39200) loss: 1.925113
(Iteration 16111 / 39200) loss: 2.372655
(Iteration 16121 / 39200) loss: 1.842463
(Iteration 16131 / 39200) loss: 1.705539
(Iteration 16141 / 39200) loss: 2.064623
(Iteration 16151 / 39200) loss: 1.755117
(Iteration 16161 / 39200) loss: 1.781009
(Iteration 16171 / 39200) loss: 1.845799
(Iteration 16181 / 39200) loss: 1.651847
(Iteration 16191 / 39200) loss: 1.916008
(Iteration 16201 / 39200) loss: 2.135783
(Iteration 16211 / 39200) loss: 1.983441
(Iteration 16221 / 39200) loss: 1.790271

(Iteration 16231 / 39200) loss: 1.750191
(Iteration 16241 / 39200) loss: 1.712369
(Iteration 16251 / 39200) loss: 2.134586
(Iteration 16261 / 39200) loss: 1.973306
(Iteration 16271 / 39200) loss: 1.732532
(Iteration 16281 / 39200) loss: 1.795778
(Iteration 16291 / 39200) loss: 1.424670
(Iteration 16301 / 39200) loss: 1.744286
(Iteration 16311 / 39200) loss: 2.468885
(Iteration 16321 / 39200) loss: 1.699234
(Iteration 16331 / 39200) loss: 2.231631
(Iteration 16341 / 39200) loss: 2.168035
(Iteration 16351 / 39200) loss: 1.938252
(Iteration 16361 / 39200) loss: 1.927362
(Iteration 16371 / 39200) loss: 1.917512
(Iteration 16381 / 39200) loss: 1.933467
(Iteration 16391 / 39200) loss: 1.701355
(Iteration 16401 / 39200) loss: 1.923383
(Iteration 16411 / 39200) loss: 1.982105
(Iteration 16421 / 39200) loss: 1.781669
(Iteration 16431 / 39200) loss: 2.050417
(Iteration 16441 / 39200) loss: 1.801472
(Iteration 16451 / 39200) loss: 1.612761
(Iteration 16461 / 39200) loss: 2.133101
(Iteration 16471 / 39200) loss: 2.001092
(Iteration 16481 / 39200) loss: 1.709718
(Iteration 16491 / 39200) loss: 1.778292
(Iteration 16501 / 39200) loss: 1.629145
(Iteration 16511 / 39200) loss: 1.551121
(Iteration 16521 / 39200) loss: 2.364336
(Iteration 16531 / 39200) loss: 1.902479
(Iteration 16541 / 39200) loss: 1.863819
(Iteration 16551 / 39200) loss: 2.008806
(Iteration 16561 / 39200) loss: 2.092545
(Iteration 16571 / 39200) loss: 2.256214
(Iteration 16581 / 39200) loss: 2.073152
(Iteration 16591 / 39200) loss: 1.870517
(Iteration 16601 / 39200) loss: 1.781287
(Iteration 16611 / 39200) loss: 2.103434
(Iteration 16621 / 39200) loss: 1.747221
(Iteration 16631 / 39200) loss: 2.075548
(Iteration 16641 / 39200) loss: 1.992105
(Iteration 16651 / 39200) loss: 1.910136
(Iteration 16661 / 39200) loss: 1.889336
(Iteration 16671 / 39200) loss: 1.685075
(Iteration 16681 / 39200) loss: 1.824860
(Iteration 16691 / 39200) loss: 2.399854
(Iteration 16701 / 39200) loss: 1.952765

(Iteration 16711 / 39200) loss: 1.513078
(Iteration 16721 / 39200) loss: 1.789471
(Iteration 16731 / 39200) loss: 1.713544
(Iteration 16741 / 39200) loss: 1.789523
(Iteration 16751 / 39200) loss: 1.872298
(Iteration 16761 / 39200) loss: 1.726187
(Iteration 16771 / 39200) loss: 1.609662
(Iteration 16781 / 39200) loss: 1.814078
(Iteration 16791 / 39200) loss: 2.360475
(Iteration 16801 / 39200) loss: 1.791085
(Iteration 16811 / 39200) loss: 1.711237
(Iteration 16821 / 39200) loss: 1.629294
(Iteration 16831 / 39200) loss: 2.136111
(Iteration 16841 / 39200) loss: 1.779334
(Iteration 16851 / 39200) loss: 1.577205
(Iteration 16861 / 39200) loss: 1.698591
(Iteration 16871 / 39200) loss: 1.686354
(Iteration 16881 / 39200) loss: 1.873862
(Iteration 16891 / 39200) loss: 1.553615
(Iteration 16901 / 39200) loss: 1.748629
(Iteration 16911 / 39200) loss: 2.444097
(Iteration 16921 / 39200) loss: 1.975127
(Iteration 16931 / 39200) loss: 1.823304
(Iteration 16941 / 39200) loss: 1.504916
(Iteration 16951 / 39200) loss: 1.785661
(Iteration 16961 / 39200) loss: 1.787095
(Iteration 16971 / 39200) loss: 1.727419
(Iteration 16981 / 39200) loss: 1.820014
(Iteration 16991 / 39200) loss: 1.466799
(Iteration 17001 / 39200) loss: 1.981637
(Iteration 17011 / 39200) loss: 1.875871
(Iteration 17021 / 39200) loss: 1.895755
(Iteration 17031 / 39200) loss: 1.946305
(Iteration 17041 / 39200) loss: 1.935048
(Iteration 17051 / 39200) loss: 2.117099
(Iteration 17061 / 39200) loss: 1.695537
(Iteration 17071 / 39200) loss: 1.929590
(Iteration 17081 / 39200) loss: 1.933879
(Iteration 17091 / 39200) loss: 1.746260
(Iteration 17101 / 39200) loss: 2.013606
(Iteration 17111 / 39200) loss: 1.691646
(Iteration 17121 / 39200) loss: 1.533997
(Iteration 17131 / 39200) loss: 2.311143
(Iteration 17141 / 39200) loss: 2.046146
(Iteration 17151 / 39200) loss: 1.742773
(Iteration 17161 / 39200) loss: 1.960528
(Iteration 17171 / 39200) loss: 1.472322
(Iteration 17181 / 39200) loss: 2.503453

(Iteration 17191 / 39200) loss: 1.895979
(Iteration 17201 / 39200) loss: 2.299140
(Iteration 17211 / 39200) loss: 1.876228
(Iteration 17221 / 39200) loss: 1.678461
(Iteration 17231 / 39200) loss: 2.182384
(Iteration 17241 / 39200) loss: 2.039427
(Iteration 17251 / 39200) loss: 1.751453
(Iteration 17261 / 39200) loss: 1.788706
(Iteration 17271 / 39200) loss: 1.908763
(Iteration 17281 / 39200) loss: 1.870498
(Iteration 17291 / 39200) loss: 2.146074
(Iteration 17301 / 39200) loss: 1.743554
(Iteration 17311 / 39200) loss: 1.870925
(Iteration 17321 / 39200) loss: 2.103611
(Iteration 17331 / 39200) loss: 1.437297
(Iteration 17341 / 39200) loss: 2.162893
(Iteration 17351 / 39200) loss: 1.851424
(Iteration 17361 / 39200) loss: 1.929648
(Iteration 17371 / 39200) loss: 1.898310
(Iteration 17381 / 39200) loss: 1.806864
(Iteration 17391 / 39200) loss: 2.063734
(Iteration 17401 / 39200) loss: 1.819443
(Iteration 17411 / 39200) loss: 2.148193
(Iteration 17421 / 39200) loss: 2.372744
(Iteration 17431 / 39200) loss: 2.015618
(Iteration 17441 / 39200) loss: 2.074708
(Iteration 17451 / 39200) loss: 1.721017
(Iteration 17461 / 39200) loss: 2.272540
(Iteration 17471 / 39200) loss: 2.147386
(Iteration 17481 / 39200) loss: 1.510401
(Iteration 17491 / 39200) loss: 1.837816
(Iteration 17501 / 39200) loss: 1.855790
(Iteration 17511 / 39200) loss: 1.724509
(Iteration 17521 / 39200) loss: 1.508971
(Iteration 17531 / 39200) loss: 1.961306
(Iteration 17541 / 39200) loss: 1.725906
(Iteration 17551 / 39200) loss: 1.933108
(Iteration 17561 / 39200) loss: 1.801790
(Iteration 17571 / 39200) loss: 1.614213
(Iteration 17581 / 39200) loss: 2.036306
(Iteration 17591 / 39200) loss: 2.277118
(Iteration 17601 / 39200) loss: 2.260447
(Iteration 17611 / 39200) loss: 1.674135
(Iteration 17621 / 39200) loss: 1.868118
(Iteration 17631 / 39200) loss: 1.587677
(Epoch 9 / 20) train acc: 0.583000; val_acc: 0.487000
(Iteration 17641 / 39200) loss: 1.988216
(Iteration 17651 / 39200) loss: 2.018262

(Iteration 17661 / 39200) loss: 1.846128
(Iteration 17671 / 39200) loss: 1.932951
(Iteration 17681 / 39200) loss: 2.147219
(Iteration 17691 / 39200) loss: 1.833493
(Iteration 17701 / 39200) loss: 1.985503
(Iteration 17711 / 39200) loss: 1.588069
(Iteration 17721 / 39200) loss: 1.942248
(Iteration 17731 / 39200) loss: 2.033701
(Iteration 17741 / 39200) loss: 2.047245
(Iteration 17751 / 39200) loss: 2.029273
(Iteration 17761 / 39200) loss: 1.717182
(Iteration 17771 / 39200) loss: 1.790239
(Iteration 17781 / 39200) loss: 1.575377
(Iteration 17791 / 39200) loss: 1.502649
(Iteration 17801 / 39200) loss: 1.648535
(Iteration 17811 / 39200) loss: 1.572695
(Iteration 17821 / 39200) loss: 1.813489
(Iteration 17831 / 39200) loss: 1.807466
(Iteration 17841 / 39200) loss: 2.130739
(Iteration 17851 / 39200) loss: 1.868112
(Iteration 17861 / 39200) loss: 1.953676
(Iteration 17871 / 39200) loss: 2.014475
(Iteration 17881 / 39200) loss: 1.854970
(Iteration 17891 / 39200) loss: 1.536192
(Iteration 17901 / 39200) loss: 2.003205
(Iteration 17911 / 39200) loss: 1.964431
(Iteration 17921 / 39200) loss: 1.765823
(Iteration 17931 / 39200) loss: 1.958636
(Iteration 17941 / 39200) loss: 1.502123
(Iteration 17951 / 39200) loss: 1.825028
(Iteration 17961 / 39200) loss: 1.804669
(Iteration 17971 / 39200) loss: 2.109089
(Iteration 17981 / 39200) loss: 1.843587
(Iteration 17991 / 39200) loss: 2.004932
(Iteration 18001 / 39200) loss: 2.228597
(Iteration 18011 / 39200) loss: 1.851549
(Iteration 18021 / 39200) loss: 1.677778
(Iteration 18031 / 39200) loss: 1.473989
(Iteration 18041 / 39200) loss: 1.538704
(Iteration 18051 / 39200) loss: 1.519408
(Iteration 18061 / 39200) loss: 1.558163
(Iteration 18071 / 39200) loss: 1.957450
(Iteration 18081 / 39200) loss: 1.505396
(Iteration 18091 / 39200) loss: 2.002846
(Iteration 18101 / 39200) loss: 2.109977
(Iteration 18111 / 39200) loss: 1.922619
(Iteration 18121 / 39200) loss: 1.924135
(Iteration 18131 / 39200) loss: 2.123617

(Iteration 18141 / 39200) loss: 1.847742
(Iteration 18151 / 39200) loss: 1.549963
(Iteration 18161 / 39200) loss: 1.454279
(Iteration 18171 / 39200) loss: 2.087860
(Iteration 18181 / 39200) loss: 1.987220
(Iteration 18191 / 39200) loss: 1.880212
(Iteration 18201 / 39200) loss: 1.655365
(Iteration 18211 / 39200) loss: 1.935675
(Iteration 18221 / 39200) loss: 1.729297
(Iteration 18231 / 39200) loss: 1.933296
(Iteration 18241 / 39200) loss: 2.282882
(Iteration 18251 / 39200) loss: 2.120148
(Iteration 18261 / 39200) loss: 1.394332
(Iteration 18271 / 39200) loss: 1.568528
(Iteration 18281 / 39200) loss: 1.555199
(Iteration 18291 / 39200) loss: 1.687402
(Iteration 18301 / 39200) loss: 1.374917
(Iteration 18311 / 39200) loss: 1.911820
(Iteration 18321 / 39200) loss: 2.117476
(Iteration 18331 / 39200) loss: 1.957664
(Iteration 18341 / 39200) loss: 2.116484
(Iteration 18351 / 39200) loss: 2.476303
(Iteration 18361 / 39200) loss: 1.930249
(Iteration 18371 / 39200) loss: 2.018566
(Iteration 18381 / 39200) loss: 1.768654
(Iteration 18391 / 39200) loss: 1.958234
(Iteration 18401 / 39200) loss: 1.597984
(Iteration 18411 / 39200) loss: 1.838134
(Iteration 18421 / 39200) loss: 1.965698
(Iteration 18431 / 39200) loss: 1.722091
(Iteration 18441 / 39200) loss: 2.013643
(Iteration 18451 / 39200) loss: 1.617112
(Iteration 18461 / 39200) loss: 1.622098
(Iteration 18471 / 39200) loss: 1.897932
(Iteration 18481 / 39200) loss: 1.766015
(Iteration 18491 / 39200) loss: 2.240239
(Iteration 18501 / 39200) loss: 1.394780
(Iteration 18511 / 39200) loss: 1.643591
(Iteration 18521 / 39200) loss: 2.057287
(Iteration 18531 / 39200) loss: 2.384908
(Iteration 18541 / 39200) loss: 1.803787
(Iteration 18551 / 39200) loss: 2.072918
(Iteration 18561 / 39200) loss: 2.214769
(Iteration 18571 / 39200) loss: 1.940768
(Iteration 18581 / 39200) loss: 1.877849
(Iteration 18591 / 39200) loss: 1.921393
(Iteration 18601 / 39200) loss: 1.573907
(Iteration 18611 / 39200) loss: 1.810972

(Iteration 18621 / 39200) loss: 1.623579
(Iteration 18631 / 39200) loss: 1.613292
(Iteration 18641 / 39200) loss: 2.395141
(Iteration 18651 / 39200) loss: 1.777845
(Iteration 18661 / 39200) loss: 1.410286
(Iteration 18671 / 39200) loss: 1.930328
(Iteration 18681 / 39200) loss: 2.187563
(Iteration 18691 / 39200) loss: 1.989951
(Iteration 18701 / 39200) loss: 1.649820
(Iteration 18711 / 39200) loss: 1.928032
(Iteration 18721 / 39200) loss: 1.778631
(Iteration 18731 / 39200) loss: 1.951749
(Iteration 18741 / 39200) loss: 1.692437
(Iteration 18751 / 39200) loss: 1.959616
(Iteration 18761 / 39200) loss: 1.909709
(Iteration 18771 / 39200) loss: 1.720373
(Iteration 18781 / 39200) loss: 1.540269
(Iteration 18791 / 39200) loss: 1.646910
(Iteration 18801 / 39200) loss: 1.873207
(Iteration 18811 / 39200) loss: 1.869766
(Iteration 18821 / 39200) loss: 1.712747
(Iteration 18831 / 39200) loss: 1.996866
(Iteration 18841 / 39200) loss: 1.839249
(Iteration 18851 / 39200) loss: 1.660737
(Iteration 18861 / 39200) loss: 1.597351
(Iteration 18871 / 39200) loss: 1.740924
(Iteration 18881 / 39200) loss: 2.259933
(Iteration 18891 / 39200) loss: 1.788987
(Iteration 18901 / 39200) loss: 1.724704
(Iteration 18911 / 39200) loss: 2.021309
(Iteration 18921 / 39200) loss: 1.694943
(Iteration 18931 / 39200) loss: 1.808679
(Iteration 18941 / 39200) loss: 1.597503
(Iteration 18951 / 39200) loss: 1.398322
(Iteration 18961 / 39200) loss: 1.698928
(Iteration 18971 / 39200) loss: 1.687133
(Iteration 18981 / 39200) loss: 1.574081
(Iteration 18991 / 39200) loss: 1.697451
(Iteration 19001 / 39200) loss: 1.576180
(Iteration 19011 / 39200) loss: 2.068744
(Iteration 19021 / 39200) loss: 1.967938
(Iteration 19031 / 39200) loss: 1.547732
(Iteration 19041 / 39200) loss: 2.132482
(Iteration 19051 / 39200) loss: 1.611703
(Iteration 19061 / 39200) loss: 1.893657
(Iteration 19071 / 39200) loss: 1.794214
(Iteration 19081 / 39200) loss: 1.662598
(Iteration 19091 / 39200) loss: 1.860542

(Iteration 19101 / 39200) loss: 2.171725
(Iteration 19111 / 39200) loss: 1.578653
(Iteration 19121 / 39200) loss: 1.774057
(Iteration 19131 / 39200) loss: 1.727458
(Iteration 19141 / 39200) loss: 1.794977
(Iteration 19151 / 39200) loss: 2.029285
(Iteration 19161 / 39200) loss: 1.648155
(Iteration 19171 / 39200) loss: 1.919915
(Iteration 19181 / 39200) loss: 1.816008
(Iteration 19191 / 39200) loss: 1.874209
(Iteration 19201 / 39200) loss: 1.548867
(Iteration 19211 / 39200) loss: 2.038919
(Iteration 19221 / 39200) loss: 1.819312
(Iteration 19231 / 39200) loss: 1.732258
(Iteration 19241 / 39200) loss: 2.079171
(Iteration 19251 / 39200) loss: 1.411979
(Iteration 19261 / 39200) loss: 1.720987
(Iteration 19271 / 39200) loss: 1.964513
(Iteration 19281 / 39200) loss: 1.634702
(Iteration 19291 / 39200) loss: 1.856698
(Iteration 19301 / 39200) loss: 1.941283
(Iteration 19311 / 39200) loss: 1.970888
(Iteration 19321 / 39200) loss: 1.518673
(Iteration 19331 / 39200) loss: 2.164154
(Iteration 19341 / 39200) loss: 2.092589
(Iteration 19351 / 39200) loss: 1.753199
(Iteration 19361 / 39200) loss: 2.189758
(Iteration 19371 / 39200) loss: 1.859674
(Iteration 19381 / 39200) loss: 1.766823
(Iteration 19391 / 39200) loss: 1.700488
(Iteration 19401 / 39200) loss: 1.989813
(Iteration 19411 / 39200) loss: 2.458260
(Iteration 19421 / 39200) loss: 2.164938
(Iteration 19431 / 39200) loss: 2.143472
(Iteration 19441 / 39200) loss: 1.998943
(Iteration 19451 / 39200) loss: 2.017894
(Iteration 19461 / 39200) loss: 2.098863
(Iteration 19471 / 39200) loss: 1.932921
(Iteration 19481 / 39200) loss: 1.972053
(Iteration 19491 / 39200) loss: 2.051006
(Iteration 19501 / 39200) loss: 1.561958
(Iteration 19511 / 39200) loss: 1.610500
(Iteration 19521 / 39200) loss: 1.734235
(Iteration 19531 / 39200) loss: 1.968927
(Iteration 19541 / 39200) loss: 1.736121
(Iteration 19551 / 39200) loss: 1.631620
(Iteration 19561 / 39200) loss: 1.622782
(Iteration 19571 / 39200) loss: 1.474081

(Iteration 19581 / 39200) loss: 1.892317
(Iteration 19591 / 39200) loss: 1.630752
(Epoch 10 / 20) train acc: 0.570000; val_acc: 0.499000
(Iteration 19601 / 39200) loss: 1.772093
(Iteration 19611 / 39200) loss: 1.755809
(Iteration 19621 / 39200) loss: 1.833493
(Iteration 19631 / 39200) loss: 1.846376
(Iteration 19641 / 39200) loss: 1.853668
(Iteration 19651 / 39200) loss: 1.801726
(Iteration 19661 / 39200) loss: 1.818576
(Iteration 19671 / 39200) loss: 1.452913
(Iteration 19681 / 39200) loss: 1.917894
(Iteration 19691 / 39200) loss: 2.074176
(Iteration 19701 / 39200) loss: 1.850954
(Iteration 19711 / 39200) loss: 1.749176
(Iteration 19721 / 39200) loss: 1.679793
(Iteration 19731 / 39200) loss: 1.434394
(Iteration 19741 / 39200) loss: 2.059122
(Iteration 19751 / 39200) loss: 1.752968
(Iteration 19761 / 39200) loss: 1.855343
(Iteration 19771 / 39200) loss: 1.914541
(Iteration 19781 / 39200) loss: 1.597528
(Iteration 19791 / 39200) loss: 1.398458
(Iteration 19801 / 39200) loss: 1.696229
(Iteration 19811 / 39200) loss: 1.427186
(Iteration 19821 / 39200) loss: 1.824654
(Iteration 19831 / 39200) loss: 1.821287
(Iteration 19841 / 39200) loss: 1.864938
(Iteration 19851 / 39200) loss: 1.378552
(Iteration 19861 / 39200) loss: 1.983176
(Iteration 19871 / 39200) loss: 1.784848
(Iteration 19881 / 39200) loss: 1.755798
(Iteration 19891 / 39200) loss: 1.670441
(Iteration 19901 / 39200) loss: 1.599515
(Iteration 19911 / 39200) loss: 1.776453
(Iteration 19921 / 39200) loss: 1.534258
(Iteration 19931 / 39200) loss: 1.739016
(Iteration 19941 / 39200) loss: 2.089638
(Iteration 19951 / 39200) loss: 1.994294
(Iteration 19961 / 39200) loss: 1.716567
(Iteration 19971 / 39200) loss: 1.734334
(Iteration 19981 / 39200) loss: 1.469456
(Iteration 19991 / 39200) loss: 1.769190
(Iteration 20001 / 39200) loss: 1.550833
(Iteration 20011 / 39200) loss: 1.876449
(Iteration 20021 / 39200) loss: 1.909287
(Iteration 20031 / 39200) loss: 2.054813
(Iteration 20041 / 39200) loss: 1.843055

(Iteration 20051 / 39200) loss: 2.035249
(Iteration 20061 / 39200) loss: 1.650638
(Iteration 20071 / 39200) loss: 1.784588
(Iteration 20081 / 39200) loss: 1.797785
(Iteration 20091 / 39200) loss: 1.944891
(Iteration 20101 / 39200) loss: 2.003396
(Iteration 20111 / 39200) loss: 1.710008
(Iteration 20121 / 39200) loss: 2.146366
(Iteration 20131 / 39200) loss: 1.558179
(Iteration 20141 / 39200) loss: 1.577103
(Iteration 20151 / 39200) loss: 1.527587
(Iteration 20161 / 39200) loss: 2.098311
(Iteration 20171 / 39200) loss: 1.864619
(Iteration 20181 / 39200) loss: 1.786941
(Iteration 20191 / 39200) loss: 1.582952
(Iteration 20201 / 39200) loss: 1.739427
(Iteration 20211 / 39200) loss: 1.677310
(Iteration 20221 / 39200) loss: 1.542674
(Iteration 20231 / 39200) loss: 1.606212
(Iteration 20241 / 39200) loss: 1.632185
(Iteration 20251 / 39200) loss: 1.635721
(Iteration 20261 / 39200) loss: 2.084014
(Iteration 20271 / 39200) loss: 1.491940
(Iteration 20281 / 39200) loss: 1.775337
(Iteration 20291 / 39200) loss: 2.151405
(Iteration 20301 / 39200) loss: 1.870720
(Iteration 20311 / 39200) loss: 1.607156
(Iteration 20321 / 39200) loss: 1.783651
(Iteration 20331 / 39200) loss: 1.708158
(Iteration 20341 / 39200) loss: 1.942736
(Iteration 20351 / 39200) loss: 1.837019
(Iteration 20361 / 39200) loss: 1.803860
(Iteration 20371 / 39200) loss: 1.693816
(Iteration 20381 / 39200) loss: 1.841634
(Iteration 20391 / 39200) loss: 1.899969
(Iteration 20401 / 39200) loss: 2.129540
(Iteration 20411 / 39200) loss: 1.637412
(Iteration 20421 / 39200) loss: 1.603928
(Iteration 20431 / 39200) loss: 2.593991
(Iteration 20441 / 39200) loss: 1.597515
(Iteration 20451 / 39200) loss: 1.667304
(Iteration 20461 / 39200) loss: 1.656451
(Iteration 20471 / 39200) loss: 2.111286
(Iteration 20481 / 39200) loss: 1.771131
(Iteration 20491 / 39200) loss: 2.145843
(Iteration 20501 / 39200) loss: 1.743679
(Iteration 20511 / 39200) loss: 1.721413
(Iteration 20521 / 39200) loss: 1.618024

(Iteration 20531 / 39200) loss: 1.548119
(Iteration 20541 / 39200) loss: 1.699722
(Iteration 20551 / 39200) loss: 1.463160
(Iteration 20561 / 39200) loss: 1.626098
(Iteration 20571 / 39200) loss: 1.854385
(Iteration 20581 / 39200) loss: 1.609033
(Iteration 20591 / 39200) loss: 1.993617
(Iteration 20601 / 39200) loss: 1.668808
(Iteration 20611 / 39200) loss: 1.472528
(Iteration 20621 / 39200) loss: 1.540382
(Iteration 20631 / 39200) loss: 1.603810
(Iteration 20641 / 39200) loss: 2.430361
(Iteration 20651 / 39200) loss: 1.927544
(Iteration 20661 / 39200) loss: 1.780295
(Iteration 20671 / 39200) loss: 1.685963
(Iteration 20681 / 39200) loss: 1.696771
(Iteration 20691 / 39200) loss: 1.366692
(Iteration 20701 / 39200) loss: 1.642128
(Iteration 20711 / 39200) loss: 1.902704
(Iteration 20721 / 39200) loss: 1.275483
(Iteration 20731 / 39200) loss: 2.006853
(Iteration 20741 / 39200) loss: 1.891186
(Iteration 20751 / 39200) loss: 1.781437
(Iteration 20761 / 39200) loss: 1.801112
(Iteration 20771 / 39200) loss: 1.449421
(Iteration 20781 / 39200) loss: 2.198560
(Iteration 20791 / 39200) loss: 1.917102
(Iteration 20801 / 39200) loss: 1.615648
(Iteration 20811 / 39200) loss: 1.625856
(Iteration 20821 / 39200) loss: 2.102360
(Iteration 20831 / 39200) loss: 1.449078
(Iteration 20841 / 39200) loss: 1.625195
(Iteration 20851 / 39200) loss: 1.520393
(Iteration 20861 / 39200) loss: 1.784372
(Iteration 20871 / 39200) loss: 1.706294
(Iteration 20881 / 39200) loss: 2.121961
(Iteration 20891 / 39200) loss: 1.595674
(Iteration 20901 / 39200) loss: 1.916504
(Iteration 20911 / 39200) loss: 1.564565
(Iteration 20921 / 39200) loss: 1.605778
(Iteration 20931 / 39200) loss: 1.372904
(Iteration 20941 / 39200) loss: 1.777840
(Iteration 20951 / 39200) loss: 2.376588
(Iteration 20961 / 39200) loss: 1.979529
(Iteration 20971 / 39200) loss: 1.542511
(Iteration 20981 / 39200) loss: 1.534096
(Iteration 20991 / 39200) loss: 1.790853
(Iteration 21001 / 39200) loss: 1.594229

(Iteration 21011 / 39200) loss: 1.617202
(Iteration 21021 / 39200) loss: 1.641650
(Iteration 21031 / 39200) loss: 1.492084
(Iteration 21041 / 39200) loss: 1.988184
(Iteration 21051 / 39200) loss: 1.805547
(Iteration 21061 / 39200) loss: 1.588481
(Iteration 21071 / 39200) loss: 1.745576
(Iteration 21081 / 39200) loss: 1.550421
(Iteration 21091 / 39200) loss: 1.824338
(Iteration 21101 / 39200) loss: 1.756681
(Iteration 21111 / 39200) loss: 1.657595
(Iteration 21121 / 39200) loss: 1.443278
(Iteration 21131 / 39200) loss: 1.700041
(Iteration 21141 / 39200) loss: 1.321617
(Iteration 21151 / 39200) loss: 1.686473
(Iteration 21161 / 39200) loss: 1.778712
(Iteration 21171 / 39200) loss: 1.206546
(Iteration 21181 / 39200) loss: 1.643311
(Iteration 21191 / 39200) loss: 1.781900
(Iteration 21201 / 39200) loss: 1.625293
(Iteration 21211 / 39200) loss: 1.816996
(Iteration 21221 / 39200) loss: 1.890412
(Iteration 21231 / 39200) loss: 1.878808
(Iteration 21241 / 39200) loss: 1.545486
(Iteration 21251 / 39200) loss: 1.706175
(Iteration 21261 / 39200) loss: 1.887157
(Iteration 21271 / 39200) loss: 1.827134
(Iteration 21281 / 39200) loss: 1.419268
(Iteration 21291 / 39200) loss: 1.684833
(Iteration 21301 / 39200) loss: 1.683955
(Iteration 21311 / 39200) loss: 1.530848
(Iteration 21321 / 39200) loss: 2.138763
(Iteration 21331 / 39200) loss: 1.891517
(Iteration 21341 / 39200) loss: 1.965168
(Iteration 21351 / 39200) loss: 1.990950
(Iteration 21361 / 39200) loss: 1.717229
(Iteration 21371 / 39200) loss: 1.872865
(Iteration 21381 / 39200) loss: 1.832028
(Iteration 21391 / 39200) loss: 1.798658
(Iteration 21401 / 39200) loss: 1.345561
(Iteration 21411 / 39200) loss: 1.583282
(Iteration 21421 / 39200) loss: 1.568949
(Iteration 21431 / 39200) loss: 1.855974
(Iteration 21441 / 39200) loss: 1.981312
(Iteration 21451 / 39200) loss: 2.125422
(Iteration 21461 / 39200) loss: 1.809481
(Iteration 21471 / 39200) loss: 2.113585
(Iteration 21481 / 39200) loss: 1.674317

(Iteration 21491 / 39200) loss: 2.142188
(Iteration 21501 / 39200) loss: 1.655175
(Iteration 21511 / 39200) loss: 1.422006
(Iteration 21521 / 39200) loss: 1.324978
(Iteration 21531 / 39200) loss: 1.184198
(Iteration 21541 / 39200) loss: 1.603788
(Iteration 21551 / 39200) loss: 1.665669
(Epoch 11 / 20) train acc: 0.581000; val_acc: 0.500000
(Iteration 21561 / 39200) loss: 1.704496
(Iteration 21571 / 39200) loss: 1.548548
(Iteration 21581 / 39200) loss: 1.807229
(Iteration 21591 / 39200) loss: 1.704083
(Iteration 21601 / 39200) loss: 1.319758
(Iteration 21611 / 39200) loss: 1.567333
(Iteration 21621 / 39200) loss: 1.532660
(Iteration 21631 / 39200) loss: 2.000640
(Iteration 21641 / 39200) loss: 1.765195
(Iteration 21651 / 39200) loss: 1.834778
(Iteration 21661 / 39200) loss: 2.180682
(Iteration 21671 / 39200) loss: 1.309047
(Iteration 21681 / 39200) loss: 1.700867
(Iteration 21691 / 39200) loss: 1.894863
(Iteration 21701 / 39200) loss: 1.845600
(Iteration 21711 / 39200) loss: 1.533541
(Iteration 21721 / 39200) loss: 1.575450
(Iteration 21731 / 39200) loss: 1.521528
(Iteration 21741 / 39200) loss: 1.806239
(Iteration 21751 / 39200) loss: 1.716019
(Iteration 21761 / 39200) loss: 1.892207
(Iteration 21771 / 39200) loss: 1.905512
(Iteration 21781 / 39200) loss: 1.754073
(Iteration 21791 / 39200) loss: 1.915163
(Iteration 21801 / 39200) loss: 1.595699
(Iteration 21811 / 39200) loss: 1.505875
(Iteration 21821 / 39200) loss: 1.802908
(Iteration 21831 / 39200) loss: 1.638697
(Iteration 21841 / 39200) loss: 1.906600
(Iteration 21851 / 39200) loss: 1.495837
(Iteration 21861 / 39200) loss: 2.116435
(Iteration 21871 / 39200) loss: 1.859477
(Iteration 21881 / 39200) loss: 1.581002
(Iteration 21891 / 39200) loss: 1.743778
(Iteration 21901 / 39200) loss: 1.645482
(Iteration 21911 / 39200) loss: 1.322797
(Iteration 21921 / 39200) loss: 1.924603
(Iteration 21931 / 39200) loss: 1.625267
(Iteration 21941 / 39200) loss: 1.620608
(Iteration 21951 / 39200) loss: 1.415033

(Iteration 21961 / 39200) loss: 1.786574
(Iteration 21971 / 39200) loss: 1.525013
(Iteration 21981 / 39200) loss: 1.466718
(Iteration 21991 / 39200) loss: 1.434882
(Iteration 22001 / 39200) loss: 1.245078
(Iteration 22011 / 39200) loss: 1.737601
(Iteration 22021 / 39200) loss: 1.857128
(Iteration 22031 / 39200) loss: 1.575787
(Iteration 22041 / 39200) loss: 1.408585
(Iteration 22051 / 39200) loss: 1.430556
(Iteration 22061 / 39200) loss: 1.602208
(Iteration 22071 / 39200) loss: 1.619194
(Iteration 22081 / 39200) loss: 1.413095
(Iteration 22091 / 39200) loss: 1.467329
(Iteration 22101 / 39200) loss: 1.690196
(Iteration 22111 / 39200) loss: 1.662971
(Iteration 22121 / 39200) loss: 1.680536
(Iteration 22131 / 39200) loss: 2.034781
(Iteration 22141 / 39200) loss: 1.709836
(Iteration 22151 / 39200) loss: 1.844236
(Iteration 22161 / 39200) loss: 1.614295
(Iteration 22171 / 39200) loss: 1.901927
(Iteration 22181 / 39200) loss: 1.651291
(Iteration 22191 / 39200) loss: 1.889157
(Iteration 22201 / 39200) loss: 1.710850
(Iteration 22211 / 39200) loss: 2.028853
(Iteration 22221 / 39200) loss: 1.594862
(Iteration 22231 / 39200) loss: 1.569010
(Iteration 22241 / 39200) loss: 1.756559
(Iteration 22251 / 39200) loss: 1.583924
(Iteration 22261 / 39200) loss: 1.915795
(Iteration 22271 / 39200) loss: 1.903930
(Iteration 22281 / 39200) loss: 1.302974
(Iteration 22291 / 39200) loss: 1.722298
(Iteration 22301 / 39200) loss: 1.971045
(Iteration 22311 / 39200) loss: 1.670462
(Iteration 22321 / 39200) loss: 1.599883
(Iteration 22331 / 39200) loss: 1.884383
(Iteration 22341 / 39200) loss: 1.552349
(Iteration 22351 / 39200) loss: 1.737723
(Iteration 22361 / 39200) loss: 1.492650
(Iteration 22371 / 39200) loss: 1.223155
(Iteration 22381 / 39200) loss: 1.617576
(Iteration 22391 / 39200) loss: 1.782394
(Iteration 22401 / 39200) loss: 2.003350
(Iteration 22411 / 39200) loss: 1.864653
(Iteration 22421 / 39200) loss: 1.702488
(Iteration 22431 / 39200) loss: 1.795046

(Iteration 22441 / 39200) loss: 1.492288
(Iteration 22451 / 39200) loss: 1.582389
(Iteration 22461 / 39200) loss: 2.128232
(Iteration 22471 / 39200) loss: 1.905108
(Iteration 22481 / 39200) loss: 2.144401
(Iteration 22491 / 39200) loss: 1.635549
(Iteration 22501 / 39200) loss: 1.358451
(Iteration 22511 / 39200) loss: 2.005438
(Iteration 22521 / 39200) loss: 1.719642
(Iteration 22531 / 39200) loss: 1.765757
(Iteration 22541 / 39200) loss: 1.808947
(Iteration 22551 / 39200) loss: 1.883763
(Iteration 22561 / 39200) loss: 1.542284
(Iteration 22571 / 39200) loss: 2.150712
(Iteration 22581 / 39200) loss: 1.958532
(Iteration 22591 / 39200) loss: 1.862139
(Iteration 22601 / 39200) loss: 1.901222
(Iteration 22611 / 39200) loss: 1.789566
(Iteration 22621 / 39200) loss: 1.627041
(Iteration 22631 / 39200) loss: 1.444502
(Iteration 22641 / 39200) loss: 1.741608
(Iteration 22651 / 39200) loss: 1.507269
(Iteration 22661 / 39200) loss: 1.677110
(Iteration 22671 / 39200) loss: 1.640363
(Iteration 22681 / 39200) loss: 1.607783
(Iteration 22691 / 39200) loss: 1.611833
(Iteration 22701 / 39200) loss: 1.632374
(Iteration 22711 / 39200) loss: 1.633120
(Iteration 22721 / 39200) loss: 1.670976
(Iteration 22731 / 39200) loss: 1.408138
(Iteration 22741 / 39200) loss: 1.481357
(Iteration 22751 / 39200) loss: 1.356709
(Iteration 22761 / 39200) loss: 1.506995
(Iteration 22771 / 39200) loss: 1.607363
(Iteration 22781 / 39200) loss: 1.545566
(Iteration 22791 / 39200) loss: 1.500850
(Iteration 22801 / 39200) loss: 1.699715
(Iteration 22811 / 39200) loss: 1.743075
(Iteration 22821 / 39200) loss: 1.754080
(Iteration 22831 / 39200) loss: 1.540023
(Iteration 22841 / 39200) loss: 1.904432
(Iteration 22851 / 39200) loss: 1.411736
(Iteration 22861 / 39200) loss: 1.730303
(Iteration 22871 / 39200) loss: 1.485664
(Iteration 22881 / 39200) loss: 1.733130
(Iteration 22891 / 39200) loss: 1.589451
(Iteration 22901 / 39200) loss: 1.722381
(Iteration 22911 / 39200) loss: 1.777383

(Iteration 22921 / 39200) loss: 1.585240
(Iteration 22931 / 39200) loss: 1.983835
(Iteration 22941 / 39200) loss: 1.698257
(Iteration 22951 / 39200) loss: 1.618233
(Iteration 22961 / 39200) loss: 1.689794
(Iteration 22971 / 39200) loss: 1.706799
(Iteration 22981 / 39200) loss: 1.558142
(Iteration 22991 / 39200) loss: 1.726124
(Iteration 23001 / 39200) loss: 1.667325
(Iteration 23011 / 39200) loss: 1.511054
(Iteration 23021 / 39200) loss: 1.811412
(Iteration 23031 / 39200) loss: 1.802021
(Iteration 23041 / 39200) loss: 1.607070
(Iteration 23051 / 39200) loss: 1.740931
(Iteration 23061 / 39200) loss: 1.704778
(Iteration 23071 / 39200) loss: 1.487841
(Iteration 23081 / 39200) loss: 1.748995
(Iteration 23091 / 39200) loss: 1.746682
(Iteration 23101 / 39200) loss: 1.387244
(Iteration 23111 / 39200) loss: 1.540049
(Iteration 23121 / 39200) loss: 1.624157
(Iteration 23131 / 39200) loss: 1.784881
(Iteration 23141 / 39200) loss: 1.496069
(Iteration 23151 / 39200) loss: 1.972685
(Iteration 23161 / 39200) loss: 1.831299
(Iteration 23171 / 39200) loss: 1.936622
(Iteration 23181 / 39200) loss: 1.707480
(Iteration 23191 / 39200) loss: 1.458147
(Iteration 23201 / 39200) loss: 1.693038
(Iteration 23211 / 39200) loss: 1.802214
(Iteration 23221 / 39200) loss: 1.446509
(Iteration 23231 / 39200) loss: 1.349657
(Iteration 23241 / 39200) loss: 1.405203
(Iteration 23251 / 39200) loss: 1.932865
(Iteration 23261 / 39200) loss: 1.645038
(Iteration 23271 / 39200) loss: 1.367489
(Iteration 23281 / 39200) loss: 1.984980
(Iteration 23291 / 39200) loss: 1.542161
(Iteration 23301 / 39200) loss: 1.694061
(Iteration 23311 / 39200) loss: 1.382517
(Iteration 23321 / 39200) loss: 1.606827
(Iteration 23331 / 39200) loss: 1.726026
(Iteration 23341 / 39200) loss: 1.956829
(Iteration 23351 / 39200) loss: 1.690203
(Iteration 23361 / 39200) loss: 1.656442
(Iteration 23371 / 39200) loss: 1.800927
(Iteration 23381 / 39200) loss: 1.740412
(Iteration 23391 / 39200) loss: 1.660815

(Iteration 23401 / 39200) loss: 1.954044
(Iteration 23411 / 39200) loss: 1.559194
(Iteration 23421 / 39200) loss: 1.462047
(Iteration 23431 / 39200) loss: 1.459036
(Iteration 23441 / 39200) loss: 1.286641
(Iteration 23451 / 39200) loss: 1.998749
(Iteration 23461 / 39200) loss: 1.302967
(Iteration 23471 / 39200) loss: 1.753678
(Iteration 23481 / 39200) loss: 1.966904
(Iteration 23491 / 39200) loss: 1.799095
(Iteration 23501 / 39200) loss: 1.634530
(Iteration 23511 / 39200) loss: 1.887462
(Epoch 12 / 20) train acc: 0.596000; val_acc: 0.485000
(Iteration 23521 / 39200) loss: 1.386217
(Iteration 23531 / 39200) loss: 2.081737
(Iteration 23541 / 39200) loss: 1.606406
(Iteration 23551 / 39200) loss: 1.795008
(Iteration 23561 / 39200) loss: 1.728137
(Iteration 23571 / 39200) loss: 1.933565
(Iteration 23581 / 39200) loss: 1.990970
(Iteration 23591 / 39200) loss: 1.611501
(Iteration 23601 / 39200) loss: 1.787729
(Iteration 23611 / 39200) loss: 1.890312
(Iteration 23621 / 39200) loss: 1.666281
(Iteration 23631 / 39200) loss: 2.022096
(Iteration 23641 / 39200) loss: 1.724627
(Iteration 23651 / 39200) loss: 1.540475
(Iteration 23661 / 39200) loss: 1.337190
(Iteration 23671 / 39200) loss: 1.686598
(Iteration 23681 / 39200) loss: 1.891310
(Iteration 23691 / 39200) loss: 1.466744
(Iteration 23701 / 39200) loss: 1.370635
(Iteration 23711 / 39200) loss: 2.040035
(Iteration 23721 / 39200) loss: 1.698067
(Iteration 23731 / 39200) loss: 1.493104
(Iteration 23741 / 39200) loss: 1.954357
(Iteration 23751 / 39200) loss: 1.736055
(Iteration 23761 / 39200) loss: 1.791579
(Iteration 23771 / 39200) loss: 1.368758
(Iteration 23781 / 39200) loss: 1.220158
(Iteration 23791 / 39200) loss: 1.228663
(Iteration 23801 / 39200) loss: 1.816931
(Iteration 23811 / 39200) loss: 2.125940
(Iteration 23821 / 39200) loss: 1.599185
(Iteration 23831 / 39200) loss: 1.450012
(Iteration 23841 / 39200) loss: 1.754472
(Iteration 23851 / 39200) loss: 1.676367
(Iteration 23861 / 39200) loss: 1.775815

(Iteration 23871 / 39200) loss: 1.833660
(Iteration 23881 / 39200) loss: 1.639822
(Iteration 23891 / 39200) loss: 1.954839
(Iteration 23901 / 39200) loss: 2.141302
(Iteration 23911 / 39200) loss: 1.678157
(Iteration 23921 / 39200) loss: 1.678490
(Iteration 23931 / 39200) loss: 1.606504
(Iteration 23941 / 39200) loss: 1.556053
(Iteration 23951 / 39200) loss: 1.699741
(Iteration 23961 / 39200) loss: 1.826645
(Iteration 23971 / 39200) loss: 2.058454
(Iteration 23981 / 39200) loss: 1.706387
(Iteration 23991 / 39200) loss: 1.777549
(Iteration 24001 / 39200) loss: 1.636235
(Iteration 24011 / 39200) loss: 1.529424
(Iteration 24021 / 39200) loss: 1.521252
(Iteration 24031 / 39200) loss: 1.547609
(Iteration 24041 / 39200) loss: 1.917803
(Iteration 24051 / 39200) loss: 1.579730
(Iteration 24061 / 39200) loss: 1.655082
(Iteration 24071 / 39200) loss: 1.531806
(Iteration 24081 / 39200) loss: 1.485378
(Iteration 24091 / 39200) loss: 1.532566
(Iteration 24101 / 39200) loss: 1.717245
(Iteration 24111 / 39200) loss: 1.727428
(Iteration 24121 / 39200) loss: 1.445499
(Iteration 24131 / 39200) loss: 2.141117
(Iteration 24141 / 39200) loss: 1.958338
(Iteration 24151 / 39200) loss: 1.220595
(Iteration 24161 / 39200) loss: 1.412444
(Iteration 24171 / 39200) loss: 1.177326
(Iteration 24181 / 39200) loss: 1.537896
(Iteration 24191 / 39200) loss: 1.490301
(Iteration 24201 / 39200) loss: 1.728576
(Iteration 24211 / 39200) loss: 1.818267
(Iteration 24221 / 39200) loss: 1.876934
(Iteration 24231 / 39200) loss: 1.607665
(Iteration 24241 / 39200) loss: 1.178037
(Iteration 24251 / 39200) loss: 1.587280
(Iteration 24261 / 39200) loss: 1.966615
(Iteration 24271 / 39200) loss: 1.791805
(Iteration 24281 / 39200) loss: 2.091608
(Iteration 24291 / 39200) loss: 1.610295
(Iteration 24301 / 39200) loss: 1.658744
(Iteration 24311 / 39200) loss: 1.469806
(Iteration 24321 / 39200) loss: 1.593679
(Iteration 24331 / 39200) loss: 1.821811
(Iteration 24341 / 39200) loss: 1.416448

(Iteration 24351 / 39200) loss: 1.474740
(Iteration 24361 / 39200) loss: 1.582518
(Iteration 24371 / 39200) loss: 2.160403
(Iteration 24381 / 39200) loss: 1.722629
(Iteration 24391 / 39200) loss: 1.428145
(Iteration 24401 / 39200) loss: 1.492564
(Iteration 24411 / 39200) loss: 1.645862
(Iteration 24421 / 39200) loss: 2.038715
(Iteration 24431 / 39200) loss: 1.797470
(Iteration 24441 / 39200) loss: 1.727043
(Iteration 24451 / 39200) loss: 1.376503
(Iteration 24461 / 39200) loss: 1.701712
(Iteration 24471 / 39200) loss: 1.898337
(Iteration 24481 / 39200) loss: 1.633894
(Iteration 24491 / 39200) loss: 1.388979
(Iteration 24501 / 39200) loss: 1.890708
(Iteration 24511 / 39200) loss: 1.865603
(Iteration 24521 / 39200) loss: 1.962235
(Iteration 24531 / 39200) loss: 1.785236
(Iteration 24541 / 39200) loss: 1.551313
(Iteration 24551 / 39200) loss: 1.639391
(Iteration 24561 / 39200) loss: 1.498348
(Iteration 24571 / 39200) loss: 1.578488
(Iteration 24581 / 39200) loss: 1.898219
(Iteration 24591 / 39200) loss: 1.888742
(Iteration 24601 / 39200) loss: 1.712239
(Iteration 24611 / 39200) loss: 1.565744
(Iteration 24621 / 39200) loss: 1.893523
(Iteration 24631 / 39200) loss: 1.523637
(Iteration 24641 / 39200) loss: 1.310841
(Iteration 24651 / 39200) loss: 1.402060
(Iteration 24661 / 39200) loss: 1.675103
(Iteration 24671 / 39200) loss: 2.004326
(Iteration 24681 / 39200) loss: 1.704541
(Iteration 24691 / 39200) loss: 1.510290
(Iteration 24701 / 39200) loss: 1.987777
(Iteration 24711 / 39200) loss: 1.626119
(Iteration 24721 / 39200) loss: 1.415655
(Iteration 24731 / 39200) loss: 1.577496
(Iteration 24741 / 39200) loss: 1.677852
(Iteration 24751 / 39200) loss: 1.544093
(Iteration 24761 / 39200) loss: 1.288727
(Iteration 24771 / 39200) loss: 1.549901
(Iteration 24781 / 39200) loss: 1.646243
(Iteration 24791 / 39200) loss: 1.182196
(Iteration 24801 / 39200) loss: 1.601164
(Iteration 24811 / 39200) loss: 1.320442
(Iteration 24821 / 39200) loss: 1.702225

(Iteration 24831 / 39200) loss: 1.680031
(Iteration 24841 / 39200) loss: 1.528407
(Iteration 24851 / 39200) loss: 1.753324
(Iteration 24861 / 39200) loss: 1.511415
(Iteration 24871 / 39200) loss: 1.944028
(Iteration 24881 / 39200) loss: 1.315221
(Iteration 24891 / 39200) loss: 1.646775
(Iteration 24901 / 39200) loss: 1.882652
(Iteration 24911 / 39200) loss: 1.686987
(Iteration 24921 / 39200) loss: 1.532484
(Iteration 24931 / 39200) loss: 1.348395
(Iteration 24941 / 39200) loss: 1.371765
(Iteration 24951 / 39200) loss: 1.384162
(Iteration 24961 / 39200) loss: 1.640079
(Iteration 24971 / 39200) loss: 1.835121
(Iteration 24981 / 39200) loss: 1.778282
(Iteration 24991 / 39200) loss: 1.528304
(Iteration 25001 / 39200) loss: 1.416416
(Iteration 25011 / 39200) loss: 1.225399
(Iteration 25021 / 39200) loss: 1.609262
(Iteration 25031 / 39200) loss: 1.800121
(Iteration 25041 / 39200) loss: 2.436261
(Iteration 25051 / 39200) loss: 1.568266
(Iteration 25061 / 39200) loss: 2.011130
(Iteration 25071 / 39200) loss: 1.664837
(Iteration 25081 / 39200) loss: 1.611713
(Iteration 25091 / 39200) loss: 1.689318
(Iteration 25101 / 39200) loss: 1.702627
(Iteration 25111 / 39200) loss: 1.503806
(Iteration 25121 / 39200) loss: 1.693300
(Iteration 25131 / 39200) loss: 2.131845
(Iteration 25141 / 39200) loss: 1.774954
(Iteration 25151 / 39200) loss: 1.497003
(Iteration 25161 / 39200) loss: 1.769383
(Iteration 25171 / 39200) loss: 1.359020
(Iteration 25181 / 39200) loss: 1.368079
(Iteration 25191 / 39200) loss: 1.559885
(Iteration 25201 / 39200) loss: 1.341085
(Iteration 25211 / 39200) loss: 1.535133
(Iteration 25221 / 39200) loss: 1.269017
(Iteration 25231 / 39200) loss: 1.502678
(Iteration 25241 / 39200) loss: 1.643993
(Iteration 25251 / 39200) loss: 1.609948
(Iteration 25261 / 39200) loss: 1.461807
(Iteration 25271 / 39200) loss: 1.650318
(Iteration 25281 / 39200) loss: 1.354303
(Iteration 25291 / 39200) loss: 1.330942
(Iteration 25301 / 39200) loss: 1.615984

(Iteration 25311 / 39200) loss: 1.916489
(Iteration 25321 / 39200) loss: 1.672623
(Iteration 25331 / 39200) loss: 1.939564
(Iteration 25341 / 39200) loss: 1.396416
(Iteration 25351 / 39200) loss: 1.283916
(Iteration 25361 / 39200) loss: 1.098731
(Iteration 25371 / 39200) loss: 1.279090
(Iteration 25381 / 39200) loss: 1.721317
(Iteration 25391 / 39200) loss: 1.821818
(Iteration 25401 / 39200) loss: 1.378822
(Iteration 25411 / 39200) loss: 1.566414
(Iteration 25421 / 39200) loss: 1.356455
(Iteration 25431 / 39200) loss: 1.290115
(Iteration 25441 / 39200) loss: 1.665674
(Iteration 25451 / 39200) loss: 1.253765
(Iteration 25461 / 39200) loss: 1.958234
(Iteration 25471 / 39200) loss: 1.533790
(Epoch 13 / 20) train acc: 0.618000; val_acc: 0.503000
(Iteration 25481 / 39200) loss: 1.617229
(Iteration 25491 / 39200) loss: 1.689321
(Iteration 25501 / 39200) loss: 1.693495
(Iteration 25511 / 39200) loss: 1.607582
(Iteration 25521 / 39200) loss: 1.451255
(Iteration 25531 / 39200) loss: 1.478660
(Iteration 25541 / 39200) loss: 1.895377
(Iteration 25551 / 39200) loss: 1.252414
(Iteration 25561 / 39200) loss: 1.681360
(Iteration 25571 / 39200) loss: 1.562311
(Iteration 25581 / 39200) loss: 1.443632
(Iteration 25591 / 39200) loss: 1.795057
(Iteration 25601 / 39200) loss: 1.569693
(Iteration 25611 / 39200) loss: 1.688265
(Iteration 25621 / 39200) loss: 1.775011
(Iteration 25631 / 39200) loss: 1.361490
(Iteration 25641 / 39200) loss: 1.789875
(Iteration 25651 / 39200) loss: 1.629734
(Iteration 25661 / 39200) loss: 1.472972
(Iteration 25671 / 39200) loss: 1.611722
(Iteration 25681 / 39200) loss: 1.517957
(Iteration 25691 / 39200) loss: 1.597537
(Iteration 25701 / 39200) loss: 1.430821
(Iteration 25711 / 39200) loss: 1.390103
(Iteration 25721 / 39200) loss: 1.443326
(Iteration 25731 / 39200) loss: 1.654027
(Iteration 25741 / 39200) loss: 1.935343
(Iteration 25751 / 39200) loss: 1.759313
(Iteration 25761 / 39200) loss: 1.178622
(Iteration 25771 / 39200) loss: 1.200309

(Iteration 25781 / 39200) loss: 1.621879
(Iteration 25791 / 39200) loss: 1.614884
(Iteration 25801 / 39200) loss: 2.004288
(Iteration 25811 / 39200) loss: 1.845245
(Iteration 25821 / 39200) loss: 1.848637
(Iteration 25831 / 39200) loss: 1.794940
(Iteration 25841 / 39200) loss: 1.559324
(Iteration 25851 / 39200) loss: 1.609625
(Iteration 25861 / 39200) loss: 1.693608
(Iteration 25871 / 39200) loss: 1.254065
(Iteration 25881 / 39200) loss: 1.337920
(Iteration 25891 / 39200) loss: 1.449730
(Iteration 25901 / 39200) loss: 1.501473
(Iteration 25911 / 39200) loss: 1.511397
(Iteration 25921 / 39200) loss: 1.747128
(Iteration 25931 / 39200) loss: 1.555195
(Iteration 25941 / 39200) loss: 1.326855
(Iteration 25951 / 39200) loss: 1.519189
(Iteration 25961 / 39200) loss: 2.080318
(Iteration 25971 / 39200) loss: 1.903527
(Iteration 25981 / 39200) loss: 1.865161
(Iteration 25991 / 39200) loss: 1.699755
(Iteration 26001 / 39200) loss: 1.693164
(Iteration 26011 / 39200) loss: 1.220680
(Iteration 26021 / 39200) loss: 1.585265
(Iteration 26031 / 39200) loss: 1.689414
(Iteration 26041 / 39200) loss: 1.572865
(Iteration 26051 / 39200) loss: 1.723171
(Iteration 26061 / 39200) loss: 1.389798
(Iteration 26071 / 39200) loss: 1.347831
(Iteration 26081 / 39200) loss: 1.426788
(Iteration 26091 / 39200) loss: 1.470024
(Iteration 26101 / 39200) loss: 1.689539
(Iteration 26111 / 39200) loss: 1.365493
(Iteration 26121 / 39200) loss: 1.668074
(Iteration 26131 / 39200) loss: 1.681666
(Iteration 26141 / 39200) loss: 1.271834
(Iteration 26151 / 39200) loss: 1.445041
(Iteration 26161 / 39200) loss: 1.497057
(Iteration 26171 / 39200) loss: 1.334054
(Iteration 26181 / 39200) loss: 1.541176
(Iteration 26191 / 39200) loss: 1.769402
(Iteration 26201 / 39200) loss: 1.545157
(Iteration 26211 / 39200) loss: 1.771172
(Iteration 26221 / 39200) loss: 1.688772
(Iteration 26231 / 39200) loss: 1.641694
(Iteration 26241 / 39200) loss: 1.547772
(Iteration 26251 / 39200) loss: 1.525706

(Iteration 26261 / 39200) loss: 1.615579
(Iteration 26271 / 39200) loss: 1.878488
(Iteration 26281 / 39200) loss: 1.441178
(Iteration 26291 / 39200) loss: 1.272004
(Iteration 26301 / 39200) loss: 1.418263
(Iteration 26311 / 39200) loss: 1.374597
(Iteration 26321 / 39200) loss: 1.714285
(Iteration 26331 / 39200) loss: 1.287360
(Iteration 26341 / 39200) loss: 1.517942
(Iteration 26351 / 39200) loss: 1.337752
(Iteration 26361 / 39200) loss: 1.450469
(Iteration 26371 / 39200) loss: 1.485072
(Iteration 26381 / 39200) loss: 1.529708
(Iteration 26391 / 39200) loss: 1.295991
(Iteration 26401 / 39200) loss: 1.469948
(Iteration 26411 / 39200) loss: 1.580888
(Iteration 26421 / 39200) loss: 1.587092
(Iteration 26431 / 39200) loss: 1.321973
(Iteration 26441 / 39200) loss: 1.556900
(Iteration 26451 / 39200) loss: 1.132722
(Iteration 26461 / 39200) loss: 1.286890
(Iteration 26471 / 39200) loss: 1.796562
(Iteration 26481 / 39200) loss: 1.670749
(Iteration 26491 / 39200) loss: 1.371702
(Iteration 26501 / 39200) loss: 1.667199
(Iteration 26511 / 39200) loss: 1.320563
(Iteration 26521 / 39200) loss: 1.856477
(Iteration 26531 / 39200) loss: 1.335659
(Iteration 26541 / 39200) loss: 1.589397
(Iteration 26551 / 39200) loss: 1.640999
(Iteration 26561 / 39200) loss: 1.544431
(Iteration 26571 / 39200) loss: 1.558197
(Iteration 26581 / 39200) loss: 1.624183
(Iteration 26591 / 39200) loss: 1.485038
(Iteration 26601 / 39200) loss: 1.570817
(Iteration 26611 / 39200) loss: 1.481337
(Iteration 26621 / 39200) loss: 1.597560
(Iteration 26631 / 39200) loss: 1.706691
(Iteration 26641 / 39200) loss: 1.909757
(Iteration 26651 / 39200) loss: 1.509338
(Iteration 26661 / 39200) loss: 1.757200
(Iteration 26671 / 39200) loss: 1.918040
(Iteration 26681 / 39200) loss: 1.567775
(Iteration 26691 / 39200) loss: 1.730584
(Iteration 26701 / 39200) loss: 1.257005
(Iteration 26711 / 39200) loss: 1.816509
(Iteration 26721 / 39200) loss: 1.627881
(Iteration 26731 / 39200) loss: 1.483983

(Iteration 26741 / 39200) loss: 1.364947
(Iteration 26751 / 39200) loss: 1.824443
(Iteration 26761 / 39200) loss: 1.726353
(Iteration 26771 / 39200) loss: 1.380914
(Iteration 26781 / 39200) loss: 1.688926
(Iteration 26791 / 39200) loss: 1.329304
(Iteration 26801 / 39200) loss: 1.958054
(Iteration 26811 / 39200) loss: 1.559529
(Iteration 26821 / 39200) loss: 1.671368
(Iteration 26831 / 39200) loss: 1.904595
(Iteration 26841 / 39200) loss: 1.377000
(Iteration 26851 / 39200) loss: 1.926069
(Iteration 26861 / 39200) loss: 1.527719
(Iteration 26871 / 39200) loss: 1.449047
(Iteration 26881 / 39200) loss: 1.325723
(Iteration 26891 / 39200) loss: 1.595850
(Iteration 26901 / 39200) loss: 1.489020
(Iteration 26911 / 39200) loss: 1.317946
(Iteration 26921 / 39200) loss: 1.629229
(Iteration 26931 / 39200) loss: 1.391184
(Iteration 26941 / 39200) loss: 1.325366
(Iteration 26951 / 39200) loss: 1.321951
(Iteration 26961 / 39200) loss: 1.343483
(Iteration 26971 / 39200) loss: 1.385779
(Iteration 26981 / 39200) loss: 1.537760
(Iteration 26991 / 39200) loss: 1.328035
(Iteration 27001 / 39200) loss: 1.210223
(Iteration 27011 / 39200) loss: 1.464319
(Iteration 27021 / 39200) loss: 1.429564
(Iteration 27031 / 39200) loss: 1.592951
(Iteration 27041 / 39200) loss: 1.767932
(Iteration 27051 / 39200) loss: 1.365795
(Iteration 27061 / 39200) loss: 1.516363
(Iteration 27071 / 39200) loss: 1.433994
(Iteration 27081 / 39200) loss: 1.972870
(Iteration 27091 / 39200) loss: 1.194123
(Iteration 27101 / 39200) loss: 1.731063
(Iteration 27111 / 39200) loss: 1.239881
(Iteration 27121 / 39200) loss: 1.642212
(Iteration 27131 / 39200) loss: 1.817908
(Iteration 27141 / 39200) loss: 1.650924
(Iteration 27151 / 39200) loss: 1.613539
(Iteration 27161 / 39200) loss: 1.403045
(Iteration 27171 / 39200) loss: 1.330021
(Iteration 27181 / 39200) loss: 1.524640
(Iteration 27191 / 39200) loss: 1.298532
(Iteration 27201 / 39200) loss: 1.523602
(Iteration 27211 / 39200) loss: 1.323519

(Iteration 27221 / 39200) loss: 1.245613
(Iteration 27231 / 39200) loss: 1.740952
(Iteration 27241 / 39200) loss: 1.748867
(Iteration 27251 / 39200) loss: 1.305077
(Iteration 27261 / 39200) loss: 1.242293
(Iteration 27271 / 39200) loss: 1.154367
(Iteration 27281 / 39200) loss: 1.524394
(Iteration 27291 / 39200) loss: 1.771352
(Iteration 27301 / 39200) loss: 1.396707
(Iteration 27311 / 39200) loss: 1.128678
(Iteration 27321 / 39200) loss: 1.811522
(Iteration 27331 / 39200) loss: 1.510037
(Iteration 27341 / 39200) loss: 1.755977
(Iteration 27351 / 39200) loss: 1.437215
(Iteration 27361 / 39200) loss: 1.437366
(Iteration 27371 / 39200) loss: 1.444573
(Iteration 27381 / 39200) loss: 2.024454
(Iteration 27391 / 39200) loss: 1.630209
(Iteration 27401 / 39200) loss: 1.628925
(Iteration 27411 / 39200) loss: 1.488304
(Iteration 27421 / 39200) loss: 1.502032
(Iteration 27431 / 39200) loss: 1.438416
(Epoch 14 / 20) train acc: 0.637000; val_acc: 0.500000
(Iteration 27441 / 39200) loss: 1.853270
(Iteration 27451 / 39200) loss: 1.720079
(Iteration 27461 / 39200) loss: 1.330193
(Iteration 27471 / 39200) loss: 1.782758
(Iteration 27481 / 39200) loss: 1.561226
(Iteration 27491 / 39200) loss: 1.555429
(Iteration 27501 / 39200) loss: 1.665393
(Iteration 27511 / 39200) loss: 1.809324
(Iteration 27521 / 39200) loss: 1.426785
(Iteration 27531 / 39200) loss: 1.563758
(Iteration 27541 / 39200) loss: 1.222948
(Iteration 27551 / 39200) loss: 1.292603
(Iteration 27561 / 39200) loss: 1.490502
(Iteration 27571 / 39200) loss: 1.312599
(Iteration 27581 / 39200) loss: 1.961902
(Iteration 27591 / 39200) loss: 1.521840
(Iteration 27601 / 39200) loss: 1.311844
(Iteration 27611 / 39200) loss: 1.387030
(Iteration 27621 / 39200) loss: 2.208014
(Iteration 27631 / 39200) loss: 1.619002
(Iteration 27641 / 39200) loss: 1.423229
(Iteration 27651 / 39200) loss: 1.710602
(Iteration 27661 / 39200) loss: 2.023113
(Iteration 27671 / 39200) loss: 1.615230
(Iteration 27681 / 39200) loss: 1.427592

(Iteration 27691 / 39200) loss: 1.322005
(Iteration 27701 / 39200) loss: 1.636951
(Iteration 27711 / 39200) loss: 1.755225
(Iteration 27721 / 39200) loss: 1.356894
(Iteration 27731 / 39200) loss: 1.514002
(Iteration 27741 / 39200) loss: 1.883036
(Iteration 27751 / 39200) loss: 1.512125
(Iteration 27761 / 39200) loss: 1.669420
(Iteration 27771 / 39200) loss: 1.285728
(Iteration 27781 / 39200) loss: 1.225397
(Iteration 27791 / 39200) loss: 1.336377
(Iteration 27801 / 39200) loss: 1.711634
(Iteration 27811 / 39200) loss: 1.970167
(Iteration 27821 / 39200) loss: 1.921881
(Iteration 27831 / 39200) loss: 1.491441
(Iteration 27841 / 39200) loss: 2.022060
(Iteration 27851 / 39200) loss: 1.239825
(Iteration 27861 / 39200) loss: 1.637670
(Iteration 27871 / 39200) loss: 1.771869
(Iteration 27881 / 39200) loss: 1.355428
(Iteration 27891 / 39200) loss: 1.614182
(Iteration 27901 / 39200) loss: 1.792454
(Iteration 27911 / 39200) loss: 1.153391
(Iteration 27921 / 39200) loss: 1.704615
(Iteration 27931 / 39200) loss: 1.468285
(Iteration 27941 / 39200) loss: 1.695802
(Iteration 27951 / 39200) loss: 1.320400
(Iteration 27961 / 39200) loss: 1.293099
(Iteration 27971 / 39200) loss: 1.569576
(Iteration 27981 / 39200) loss: 1.374327
(Iteration 27991 / 39200) loss: 1.294149
(Iteration 28001 / 39200) loss: 1.475458
(Iteration 28011 / 39200) loss: 1.344829
(Iteration 28021 / 39200) loss: 1.478989
(Iteration 28031 / 39200) loss: 1.707580
(Iteration 28041 / 39200) loss: 1.521416
(Iteration 28051 / 39200) loss: 1.480761
(Iteration 28061 / 39200) loss: 1.593611
(Iteration 28071 / 39200) loss: 1.607954
(Iteration 28081 / 39200) loss: 1.284787
(Iteration 28091 / 39200) loss: 1.550954
(Iteration 28101 / 39200) loss: 1.357938
(Iteration 28111 / 39200) loss: 1.164558
(Iteration 28121 / 39200) loss: 1.213054
(Iteration 28131 / 39200) loss: 1.266216
(Iteration 28141 / 39200) loss: 1.641485
(Iteration 28151 / 39200) loss: 1.253977
(Iteration 28161 / 39200) loss: 1.312095

(Iteration 28171 / 39200) loss: 1.509220
(Iteration 28181 / 39200) loss: 1.602574
(Iteration 28191 / 39200) loss: 1.410805
(Iteration 28201 / 39200) loss: 1.743191
(Iteration 28211 / 39200) loss: 1.466198
(Iteration 28221 / 39200) loss: 1.466975
(Iteration 28231 / 39200) loss: 1.557676
(Iteration 28241 / 39200) loss: 1.747014
(Iteration 28251 / 39200) loss: 1.646086
(Iteration 28261 / 39200) loss: 1.516295
(Iteration 28271 / 39200) loss: 1.314162
(Iteration 28281 / 39200) loss: 1.615408
(Iteration 28291 / 39200) loss: 1.447567
(Iteration 28301 / 39200) loss: 1.715391
(Iteration 28311 / 39200) loss: 1.586697
(Iteration 28321 / 39200) loss: 1.336292
(Iteration 28331 / 39200) loss: 1.412413
(Iteration 28341 / 39200) loss: 1.666108
(Iteration 28351 / 39200) loss: 1.399861
(Iteration 28361 / 39200) loss: 1.558807
(Iteration 28371 / 39200) loss: 1.347079
(Iteration 28381 / 39200) loss: 1.637563
(Iteration 28391 / 39200) loss: 1.800828
(Iteration 28401 / 39200) loss: 1.688510
(Iteration 28411 / 39200) loss: 1.694156
(Iteration 28421 / 39200) loss: 1.663691
(Iteration 28431 / 39200) loss: 1.332501
(Iteration 28441 / 39200) loss: 1.346029
(Iteration 28451 / 39200) loss: 1.772447
(Iteration 28461 / 39200) loss: 1.588270
(Iteration 28471 / 39200) loss: 1.576031
(Iteration 28481 / 39200) loss: 1.399988
(Iteration 28491 / 39200) loss: 1.523196
(Iteration 28501 / 39200) loss: 1.278741
(Iteration 28511 / 39200) loss: 1.672401
(Iteration 28521 / 39200) loss: 1.679768
(Iteration 28531 / 39200) loss: 1.218475
(Iteration 28541 / 39200) loss: 1.567941
(Iteration 28551 / 39200) loss: 1.458367
(Iteration 28561 / 39200) loss: 1.579135
(Iteration 28571 / 39200) loss: 1.670403
(Iteration 28581 / 39200) loss: 1.896493
(Iteration 28591 / 39200) loss: 1.622165
(Iteration 28601 / 39200) loss: 1.128674
(Iteration 28611 / 39200) loss: 1.917778
(Iteration 28621 / 39200) loss: 2.209722
(Iteration 28631 / 39200) loss: 1.510491
(Iteration 28641 / 39200) loss: 1.152790

(Iteration 28651 / 39200) loss: 1.600557
(Iteration 28661 / 39200) loss: 1.464030
(Iteration 28671 / 39200) loss: 1.297917
(Iteration 28681 / 39200) loss: 1.340079
(Iteration 28691 / 39200) loss: 1.390925
(Iteration 28701 / 39200) loss: 1.262669
(Iteration 28711 / 39200) loss: 1.896364
(Iteration 28721 / 39200) loss: 1.531876
(Iteration 28731 / 39200) loss: 1.582548
(Iteration 28741 / 39200) loss: 1.340442
(Iteration 28751 / 39200) loss: 1.615769
(Iteration 28761 / 39200) loss: 1.452939
(Iteration 28771 / 39200) loss: 1.695145
(Iteration 28781 / 39200) loss: 1.648900
(Iteration 28791 / 39200) loss: 1.377536
(Iteration 28801 / 39200) loss: 1.900890
(Iteration 28811 / 39200) loss: 1.771754
(Iteration 28821 / 39200) loss: 1.244125
(Iteration 28831 / 39200) loss: 1.928767
(Iteration 28841 / 39200) loss: 1.574869
(Iteration 28851 / 39200) loss: 1.539541
(Iteration 28861 / 39200) loss: 1.548346
(Iteration 28871 / 39200) loss: 1.310627
(Iteration 28881 / 39200) loss: 1.280542
(Iteration 28891 / 39200) loss: 1.177062
(Iteration 28901 / 39200) loss: 1.938486
(Iteration 28911 / 39200) loss: 1.664966
(Iteration 28921 / 39200) loss: 1.753049
(Iteration 28931 / 39200) loss: 1.676032
(Iteration 28941 / 39200) loss: 1.925523
(Iteration 28951 / 39200) loss: 1.491212
(Iteration 28961 / 39200) loss: 1.716805
(Iteration 28971 / 39200) loss: 1.540070
(Iteration 28981 / 39200) loss: 1.456620
(Iteration 28991 / 39200) loss: 1.698033
(Iteration 29001 / 39200) loss: 1.484786
(Iteration 29011 / 39200) loss: 1.611119
(Iteration 29021 / 39200) loss: 1.442055
(Iteration 29031 / 39200) loss: 1.608708
(Iteration 29041 / 39200) loss: 1.187198
(Iteration 29051 / 39200) loss: 1.726373
(Iteration 29061 / 39200) loss: 1.402996
(Iteration 29071 / 39200) loss: 0.938455
(Iteration 29081 / 39200) loss: 1.308393
(Iteration 29091 / 39200) loss: 1.493411
(Iteration 29101 / 39200) loss: 1.524680
(Iteration 29111 / 39200) loss: 1.679373
(Iteration 29121 / 39200) loss: 1.289279

(Iteration 29131 / 39200) loss: 1.423650
(Iteration 29141 / 39200) loss: 1.603864
(Iteration 29151 / 39200) loss: 1.209287
(Iteration 29161 / 39200) loss: 1.978352
(Iteration 29171 / 39200) loss: 1.350682
(Iteration 29181 / 39200) loss: 0.998240
(Iteration 29191 / 39200) loss: 1.440919
(Iteration 29201 / 39200) loss: 1.248973
(Iteration 29211 / 39200) loss: 1.772755
(Iteration 29221 / 39200) loss: 1.493460
(Iteration 29231 / 39200) loss: 1.034587
(Iteration 29241 / 39200) loss: 1.713278
(Iteration 29251 / 39200) loss: 1.478854
(Iteration 29261 / 39200) loss: 1.540938
(Iteration 29271 / 39200) loss: 1.357915
(Iteration 29281 / 39200) loss: 1.646329
(Iteration 29291 / 39200) loss: 1.298081
(Iteration 29301 / 39200) loss: 1.424712
(Iteration 29311 / 39200) loss: 1.350505
(Iteration 29321 / 39200) loss: 1.714905
(Iteration 29331 / 39200) loss: 1.211820
(Iteration 29341 / 39200) loss: 1.741350
(Iteration 29351 / 39200) loss: 1.267617
(Iteration 29361 / 39200) loss: 1.310008
(Iteration 29371 / 39200) loss: 1.236945
(Iteration 29381 / 39200) loss: 1.461984
(Iteration 29391 / 39200) loss: 1.599367
(Epoch 15 / 20) train acc: 0.685000; val_acc: 0.495000
(Iteration 29401 / 39200) loss: 1.614947
(Iteration 29411 / 39200) loss: 1.425960
(Iteration 29421 / 39200) loss: 1.924524
(Iteration 29431 / 39200) loss: 1.488825
(Iteration 29441 / 39200) loss: 1.394133
(Iteration 29451 / 39200) loss: 1.221870
(Iteration 29461 / 39200) loss: 1.305990
(Iteration 29471 / 39200) loss: 1.594995
(Iteration 29481 / 39200) loss: 1.366512
(Iteration 29491 / 39200) loss: 1.687672
(Iteration 29501 / 39200) loss: 1.350442
(Iteration 29511 / 39200) loss: 1.691047
(Iteration 29521 / 39200) loss: 1.482588
(Iteration 29531 / 39200) loss: 1.317570
(Iteration 29541 / 39200) loss: 1.356026
(Iteration 29551 / 39200) loss: 1.244571
(Iteration 29561 / 39200) loss: 1.574579
(Iteration 29571 / 39200) loss: 1.768608
(Iteration 29581 / 39200) loss: 1.649254
(Iteration 29591 / 39200) loss: 1.352661

(Iteration 29601 / 39200) loss: 1.355735
(Iteration 29611 / 39200) loss: 1.568698
(Iteration 29621 / 39200) loss: 1.511919
(Iteration 29631 / 39200) loss: 1.237765
(Iteration 29641 / 39200) loss: 1.281400
(Iteration 29651 / 39200) loss: 1.454878
(Iteration 29661 / 39200) loss: 1.925934
(Iteration 29671 / 39200) loss: 1.363812
(Iteration 29681 / 39200) loss: 1.393448
(Iteration 29691 / 39200) loss: 1.222122
(Iteration 29701 / 39200) loss: 1.525699
(Iteration 29711 / 39200) loss: 1.208881
(Iteration 29721 / 39200) loss: 1.061118
(Iteration 29731 / 39200) loss: 1.614487
(Iteration 29741 / 39200) loss: 1.676614
(Iteration 29751 / 39200) loss: 1.394823
(Iteration 29761 / 39200) loss: 1.318892
(Iteration 29771 / 39200) loss: 1.229526
(Iteration 29781 / 39200) loss: 1.908738
(Iteration 29791 / 39200) loss: 1.478872
(Iteration 29801 / 39200) loss: 1.618609
(Iteration 29811 / 39200) loss: 1.620425
(Iteration 29821 / 39200) loss: 1.343962
(Iteration 29831 / 39200) loss: 1.546627
(Iteration 29841 / 39200) loss: 1.534831
(Iteration 29851 / 39200) loss: 1.315263
(Iteration 29861 / 39200) loss: 1.268338
(Iteration 29871 / 39200) loss: 1.538925
(Iteration 29881 / 39200) loss: 1.052271
(Iteration 29891 / 39200) loss: 1.774381
(Iteration 29901 / 39200) loss: 1.200285
(Iteration 29911 / 39200) loss: 1.697821
(Iteration 29921 / 39200) loss: 1.335741
(Iteration 29931 / 39200) loss: 1.372488
(Iteration 29941 / 39200) loss: 1.173890
(Iteration 29951 / 39200) loss: 1.458278
(Iteration 29961 / 39200) loss: 1.531710
(Iteration 29971 / 39200) loss: 1.409205
(Iteration 29981 / 39200) loss: 1.403805
(Iteration 29991 / 39200) loss: 1.467498
(Iteration 30001 / 39200) loss: 1.602368
(Iteration 30011 / 39200) loss: 1.586143
(Iteration 30021 / 39200) loss: 1.486955
(Iteration 30031 / 39200) loss: 2.088180
(Iteration 30041 / 39200) loss: 1.543565
(Iteration 30051 / 39200) loss: 1.486971
(Iteration 30061 / 39200) loss: 1.348862
(Iteration 30071 / 39200) loss: 1.558829

(Iteration 30081 / 39200) loss: 1.475320
(Iteration 30091 / 39200) loss: 1.387968
(Iteration 30101 / 39200) loss: 1.935137
(Iteration 30111 / 39200) loss: 1.580311
(Iteration 30121 / 39200) loss: 1.373519
(Iteration 30131 / 39200) loss: 1.020069
(Iteration 30141 / 39200) loss: 1.454323
(Iteration 30151 / 39200) loss: 1.711740
(Iteration 30161 / 39200) loss: 1.335074
(Iteration 30171 / 39200) loss: 1.388441
(Iteration 30181 / 39200) loss: 1.271922
(Iteration 30191 / 39200) loss: 1.441100
(Iteration 30201 / 39200) loss: 1.430471
(Iteration 30211 / 39200) loss: 1.644315
(Iteration 30221 / 39200) loss: 1.211717
(Iteration 30231 / 39200) loss: 1.302076
(Iteration 30241 / 39200) loss: 1.649815
(Iteration 30251 / 39200) loss: 1.234469
(Iteration 30261 / 39200) loss: 1.449962
(Iteration 30271 / 39200) loss: 1.745081
(Iteration 30281 / 39200) loss: 1.269315
(Iteration 30291 / 39200) loss: 1.195045
(Iteration 30301 / 39200) loss: 1.354969
(Iteration 30311 / 39200) loss: 1.424437
(Iteration 30321 / 39200) loss: 0.971493
(Iteration 30331 / 39200) loss: 1.320550
(Iteration 30341 / 39200) loss: 1.310150
(Iteration 30351 / 39200) loss: 1.494634
(Iteration 30361 / 39200) loss: 1.692060
(Iteration 30371 / 39200) loss: 1.436929
(Iteration 30381 / 39200) loss: 1.837575
(Iteration 30391 / 39200) loss: 1.375749
(Iteration 30401 / 39200) loss: 1.233003
(Iteration 30411 / 39200) loss: 1.493169
(Iteration 30421 / 39200) loss: 1.586413
(Iteration 30431 / 39200) loss: 1.330252
(Iteration 30441 / 39200) loss: 1.407322
(Iteration 30451 / 39200) loss: 1.350690
(Iteration 30461 / 39200) loss: 1.358749
(Iteration 30471 / 39200) loss: 1.504500
(Iteration 30481 / 39200) loss: 1.192195
(Iteration 30491 / 39200) loss: 1.282882
(Iteration 30501 / 39200) loss: 1.260218
(Iteration 30511 / 39200) loss: 1.247688
(Iteration 30521 / 39200) loss: 1.510567
(Iteration 30531 / 39200) loss: 1.371261
(Iteration 30541 / 39200) loss: 1.607813
(Iteration 30551 / 39200) loss: 1.284793

(Iteration 30561 / 39200) loss: 1.054479
(Iteration 30571 / 39200) loss: 1.552154
(Iteration 30581 / 39200) loss: 1.611586
(Iteration 30591 / 39200) loss: 1.340926
(Iteration 30601 / 39200) loss: 1.141695
(Iteration 30611 / 39200) loss: 1.565374
(Iteration 30621 / 39200) loss: 1.404460
(Iteration 30631 / 39200) loss: 1.907378
(Iteration 30641 / 39200) loss: 1.444781
(Iteration 30651 / 39200) loss: 1.394523
(Iteration 30661 / 39200) loss: 1.561705
(Iteration 30671 / 39200) loss: 1.596739
(Iteration 30681 / 39200) loss: 1.264147
(Iteration 30691 / 39200) loss: 1.583993
(Iteration 30701 / 39200) loss: 1.307590
(Iteration 30711 / 39200) loss: 1.321051
(Iteration 30721 / 39200) loss: 1.542648
(Iteration 30731 / 39200) loss: 1.315352
(Iteration 30741 / 39200) loss: 1.353306
(Iteration 30751 / 39200) loss: 1.890855
(Iteration 30761 / 39200) loss: 1.413703
(Iteration 30771 / 39200) loss: 1.336297
(Iteration 30781 / 39200) loss: 1.067379
(Iteration 30791 / 39200) loss: 1.239062
(Iteration 30801 / 39200) loss: 1.313610
(Iteration 30811 / 39200) loss: 1.450671
(Iteration 30821 / 39200) loss: 1.407027
(Iteration 30831 / 39200) loss: 1.096029
(Iteration 30841 / 39200) loss: 1.477260
(Iteration 30851 / 39200) loss: 1.474945
(Iteration 30861 / 39200) loss: 1.473333
(Iteration 30871 / 39200) loss: 1.097228
(Iteration 30881 / 39200) loss: 1.322598
(Iteration 30891 / 39200) loss: 1.443938
(Iteration 30901 / 39200) loss: 1.369112
(Iteration 30911 / 39200) loss: 1.632633
(Iteration 30921 / 39200) loss: 1.091281
(Iteration 30931 / 39200) loss: 1.291196
(Iteration 30941 / 39200) loss: 1.220953
(Iteration 30951 / 39200) loss: 1.717434
(Iteration 30961 / 39200) loss: 1.546110
(Iteration 30971 / 39200) loss: 1.377476
(Iteration 30981 / 39200) loss: 1.182853
(Iteration 30991 / 39200) loss: 1.416005
(Iteration 31001 / 39200) loss: 1.512371
(Iteration 31011 / 39200) loss: 1.618451
(Iteration 31021 / 39200) loss: 1.488180
(Iteration 31031 / 39200) loss: 1.766254

(Iteration 31041 / 39200) loss: 1.832507
(Iteration 31051 / 39200) loss: 1.167719
(Iteration 31061 / 39200) loss: 1.715728
(Iteration 31071 / 39200) loss: 1.123775
(Iteration 31081 / 39200) loss: 1.440792
(Iteration 31091 / 39200) loss: 1.349807
(Iteration 31101 / 39200) loss: 1.261993
(Iteration 31111 / 39200) loss: 1.654012
(Iteration 31121 / 39200) loss: 1.470840
(Iteration 31131 / 39200) loss: 1.534832
(Iteration 31141 / 39200) loss: 1.522642
(Iteration 31151 / 39200) loss: 1.332320
(Iteration 31161 / 39200) loss: 1.454109
(Iteration 31171 / 39200) loss: 1.758160
(Iteration 31181 / 39200) loss: 1.415757
(Iteration 31191 / 39200) loss: 1.041498
(Iteration 31201 / 39200) loss: 1.890543
(Iteration 31211 / 39200) loss: 1.756239
(Iteration 31221 / 39200) loss: 1.458169
(Iteration 31231 / 39200) loss: 1.436254
(Iteration 31241 / 39200) loss: 1.582093
(Iteration 31251 / 39200) loss: 1.381980
(Iteration 31261 / 39200) loss: 1.585546
(Iteration 31271 / 39200) loss: 1.443692
(Iteration 31281 / 39200) loss: 1.611968
(Iteration 31291 / 39200) loss: 1.493351
(Iteration 31301 / 39200) loss: 1.520438
(Iteration 31311 / 39200) loss: 1.539544
(Iteration 31321 / 39200) loss: 1.145532
(Iteration 31331 / 39200) loss: 1.434815
(Iteration 31341 / 39200) loss: 1.536983
(Iteration 31351 / 39200) loss: 1.336645
(Epoch 16 / 20) train acc: 0.666000; val_acc: 0.517000
(Iteration 31361 / 39200) loss: 1.312022
(Iteration 31371 / 39200) loss: 1.230898
(Iteration 31381 / 39200) loss: 1.152879
(Iteration 31391 / 39200) loss: 1.792925
(Iteration 31401 / 39200) loss: 1.382436
(Iteration 31411 / 39200) loss: 1.633949
(Iteration 31421 / 39200) loss: 1.590293
(Iteration 31431 / 39200) loss: 1.269909
(Iteration 31441 / 39200) loss: 1.053777
(Iteration 31451 / 39200) loss: 1.314779
(Iteration 31461 / 39200) loss: 1.326468
(Iteration 31471 / 39200) loss: 1.222807
(Iteration 31481 / 39200) loss: 1.587737
(Iteration 31491 / 39200) loss: 1.288899
(Iteration 31501 / 39200) loss: 1.865382

(Iteration 31511 / 39200) loss: 1.553938
(Iteration 31521 / 39200) loss: 1.401856
(Iteration 31531 / 39200) loss: 1.623170
(Iteration 31541 / 39200) loss: 1.339128
(Iteration 31551 / 39200) loss: 1.442350
(Iteration 31561 / 39200) loss: 1.249067
(Iteration 31571 / 39200) loss: 1.223487
(Iteration 31581 / 39200) loss: 1.534718
(Iteration 31591 / 39200) loss: 1.824114
(Iteration 31601 / 39200) loss: 1.268758
(Iteration 31611 / 39200) loss: 1.213554
(Iteration 31621 / 39200) loss: 1.594096
(Iteration 31631 / 39200) loss: 1.273188
(Iteration 31641 / 39200) loss: 1.331522
(Iteration 31651 / 39200) loss: 1.226507
(Iteration 31661 / 39200) loss: 1.650784
(Iteration 31671 / 39200) loss: 1.508567
(Iteration 31681 / 39200) loss: 1.332017
(Iteration 31691 / 39200) loss: 1.426507
(Iteration 31701 / 39200) loss: 1.330871
(Iteration 31711 / 39200) loss: 1.491855
(Iteration 31721 / 39200) loss: 1.178217
(Iteration 31731 / 39200) loss: 1.759075
(Iteration 31741 / 39200) loss: 1.890397
(Iteration 31751 / 39200) loss: 1.241777
(Iteration 31761 / 39200) loss: 1.406799
(Iteration 31771 / 39200) loss: 1.504068
(Iteration 31781 / 39200) loss: 1.801699
(Iteration 31791 / 39200) loss: 1.199010
(Iteration 31801 / 39200) loss: 1.868657
(Iteration 31811 / 39200) loss: 1.138688
(Iteration 31821 / 39200) loss: 1.224530
(Iteration 31831 / 39200) loss: 1.359847
(Iteration 31841 / 39200) loss: 1.356970
(Iteration 31851 / 39200) loss: 1.584533
(Iteration 31861 / 39200) loss: 1.423553
(Iteration 31871 / 39200) loss: 1.346703
(Iteration 31881 / 39200) loss: 1.421704
(Iteration 31891 / 39200) loss: 1.283493
(Iteration 31901 / 39200) loss: 1.506531
(Iteration 31911 / 39200) loss: 1.227907
(Iteration 31921 / 39200) loss: 1.552545
(Iteration 31931 / 39200) loss: 1.385308
(Iteration 31941 / 39200) loss: 1.432626
(Iteration 31951 / 39200) loss: 1.514912
(Iteration 31961 / 39200) loss: 1.618362
(Iteration 31971 / 39200) loss: 1.692933
(Iteration 31981 / 39200) loss: 1.163448

(Iteration 31991 / 39200) loss: 1.400618
(Iteration 32001 / 39200) loss: 1.489363
(Iteration 32011 / 39200) loss: 1.310855
(Iteration 32021 / 39200) loss: 1.368732
(Iteration 32031 / 39200) loss: 1.137531
(Iteration 32041 / 39200) loss: 1.463684
(Iteration 32051 / 39200) loss: 1.337220
(Iteration 32061 / 39200) loss: 1.420199
(Iteration 32071 / 39200) loss: 1.639456
(Iteration 32081 / 39200) loss: 1.307210
(Iteration 32091 / 39200) loss: 1.319545
(Iteration 32101 / 39200) loss: 1.308226
(Iteration 32111 / 39200) loss: 1.113488
(Iteration 32121 / 39200) loss: 1.435474
(Iteration 32131 / 39200) loss: 1.393922
(Iteration 32141 / 39200) loss: 1.028123
(Iteration 32151 / 39200) loss: 1.114335
(Iteration 32161 / 39200) loss: 1.344616
(Iteration 32171 / 39200) loss: 1.379375
(Iteration 32181 / 39200) loss: 1.264832
(Iteration 32191 / 39200) loss: 1.302995
(Iteration 32201 / 39200) loss: 1.506599
(Iteration 32211 / 39200) loss: 1.654257
(Iteration 32221 / 39200) loss: 1.572151
(Iteration 32231 / 39200) loss: 1.595477
(Iteration 32241 / 39200) loss: 1.264791
(Iteration 32251 / 39200) loss: 1.325013
(Iteration 32261 / 39200) loss: 1.162496
(Iteration 32271 / 39200) loss: 1.130741
(Iteration 32281 / 39200) loss: 1.775217
(Iteration 32291 / 39200) loss: 1.614220
(Iteration 32301 / 39200) loss: 1.473941
(Iteration 32311 / 39200) loss: 1.486945
(Iteration 32321 / 39200) loss: 1.686469
(Iteration 32331 / 39200) loss: 1.556377
(Iteration 32341 / 39200) loss: 1.578217
(Iteration 32351 / 39200) loss: 1.413540
(Iteration 32361 / 39200) loss: 1.238721
(Iteration 32371 / 39200) loss: 1.910906
(Iteration 32381 / 39200) loss: 1.295503
(Iteration 32391 / 39200) loss: 1.237585
(Iteration 32401 / 39200) loss: 1.429189
(Iteration 32411 / 39200) loss: 1.394822
(Iteration 32421 / 39200) loss: 1.535091
(Iteration 32431 / 39200) loss: 1.506449
(Iteration 32441 / 39200) loss: 1.175142
(Iteration 32451 / 39200) loss: 1.805770
(Iteration 32461 / 39200) loss: 1.247432

(Iteration 32471 / 39200) loss: 1.383334
(Iteration 32481 / 39200) loss: 1.331780
(Iteration 32491 / 39200) loss: 1.749870
(Iteration 32501 / 39200) loss: 1.303582
(Iteration 32511 / 39200) loss: 1.125260
(Iteration 32521 / 39200) loss: 1.372679
(Iteration 32531 / 39200) loss: 1.318030
(Iteration 32541 / 39200) loss: 1.300694
(Iteration 32551 / 39200) loss: 1.486825
(Iteration 32561 / 39200) loss: 1.452463
(Iteration 32571 / 39200) loss: 1.497946
(Iteration 32581 / 39200) loss: 1.348428
(Iteration 32591 / 39200) loss: 1.767031
(Iteration 32601 / 39200) loss: 1.177889
(Iteration 32611 / 39200) loss: 1.212011
(Iteration 32621 / 39200) loss: 1.289716
(Iteration 32631 / 39200) loss: 1.046421
(Iteration 32641 / 39200) loss: 1.591536
(Iteration 32651 / 39200) loss: 1.330748
(Iteration 32661 / 39200) loss: 1.593445
(Iteration 32671 / 39200) loss: 1.463768
(Iteration 32681 / 39200) loss: 1.678002
(Iteration 32691 / 39200) loss: 1.542894
(Iteration 32701 / 39200) loss: 1.386516
(Iteration 32711 / 39200) loss: 1.231294
(Iteration 32721 / 39200) loss: 1.461783
(Iteration 32731 / 39200) loss: 1.361139
(Iteration 32741 / 39200) loss: 1.480128
(Iteration 32751 / 39200) loss: 1.428683
(Iteration 32761 / 39200) loss: 1.623177
(Iteration 32771 / 39200) loss: 1.295702
(Iteration 32781 / 39200) loss: 1.559424
(Iteration 32791 / 39200) loss: 1.155238
(Iteration 32801 / 39200) loss: 1.055125
(Iteration 32811 / 39200) loss: 1.338742
(Iteration 32821 / 39200) loss: 1.573184
(Iteration 32831 / 39200) loss: 1.317330
(Iteration 32841 / 39200) loss: 1.539024
(Iteration 32851 / 39200) loss: 1.446291
(Iteration 32861 / 39200) loss: 1.238049
(Iteration 32871 / 39200) loss: 1.460809
(Iteration 32881 / 39200) loss: 1.442718
(Iteration 32891 / 39200) loss: 1.223102
(Iteration 32901 / 39200) loss: 1.223941
(Iteration 32911 / 39200) loss: 1.442504
(Iteration 32921 / 39200) loss: 1.268606
(Iteration 32931 / 39200) loss: 1.596112
(Iteration 32941 / 39200) loss: 1.304834

(Iteration 32951 / 39200) loss: 1.988037
(Iteration 32961 / 39200) loss: 1.745269
(Iteration 32971 / 39200) loss: 1.111125
(Iteration 32981 / 39200) loss: 1.202829
(Iteration 32991 / 39200) loss: 1.381092
(Iteration 33001 / 39200) loss: 1.269217
(Iteration 33011 / 39200) loss: 1.078574
(Iteration 33021 / 39200) loss: 1.599736
(Iteration 33031 / 39200) loss: 1.410318
(Iteration 33041 / 39200) loss: 1.259782
(Iteration 33051 / 39200) loss: 0.972109
(Iteration 33061 / 39200) loss: 1.994164
(Iteration 33071 / 39200) loss: 1.409127
(Iteration 33081 / 39200) loss: 1.095316
(Iteration 33091 / 39200) loss: 1.478104
(Iteration 33101 / 39200) loss: 1.192825
(Iteration 33111 / 39200) loss: 1.687580
(Iteration 33121 / 39200) loss: 1.344067
(Iteration 33131 / 39200) loss: 1.281950
(Iteration 33141 / 39200) loss: 1.030818
(Iteration 33151 / 39200) loss: 1.101920
(Iteration 33161 / 39200) loss: 1.403153
(Iteration 33171 / 39200) loss: 1.061351
(Iteration 33181 / 39200) loss: 1.548536
(Iteration 33191 / 39200) loss: 1.342693
(Iteration 33201 / 39200) loss: 1.124549
(Iteration 33211 / 39200) loss: 1.445368
(Iteration 33221 / 39200) loss: 1.580685
(Iteration 33231 / 39200) loss: 1.567491
(Iteration 33241 / 39200) loss: 1.812958
(Iteration 33251 / 39200) loss: 1.451667
(Iteration 33261 / 39200) loss: 1.508356
(Iteration 33271 / 39200) loss: 1.498358
(Iteration 33281 / 39200) loss: 1.060296
(Iteration 33291 / 39200) loss: 1.354715
(Iteration 33301 / 39200) loss: 1.612892
(Iteration 33311 / 39200) loss: 1.487156
(Epoch 17 / 20) train acc: 0.643000; val_acc: 0.486000
(Iteration 33321 / 39200) loss: 1.633382
(Iteration 33331 / 39200) loss: 1.173190
(Iteration 33341 / 39200) loss: 1.074807
(Iteration 33351 / 39200) loss: 1.028817
(Iteration 33361 / 39200) loss: 1.199215
(Iteration 33371 / 39200) loss: 1.250195
(Iteration 33381 / 39200) loss: 1.524785
(Iteration 33391 / 39200) loss: 1.261935
(Iteration 33401 / 39200) loss: 1.526119
(Iteration 33411 / 39200) loss: 1.470986

(Iteration 33421 / 39200) loss: 1.412209
(Iteration 33431 / 39200) loss: 0.969118
(Iteration 33441 / 39200) loss: 1.084158
(Iteration 33451 / 39200) loss: 1.285966
(Iteration 33461 / 39200) loss: 1.667339
(Iteration 33471 / 39200) loss: 1.459001
(Iteration 33481 / 39200) loss: 1.007132
(Iteration 33491 / 39200) loss: 1.051562
(Iteration 33501 / 39200) loss: 1.430154
(Iteration 33511 / 39200) loss: 1.533838
(Iteration 33521 / 39200) loss: 1.506436
(Iteration 33531 / 39200) loss: 1.406007
(Iteration 33541 / 39200) loss: 1.094667
(Iteration 33551 / 39200) loss: 1.575949
(Iteration 33561 / 39200) loss: 1.054808
(Iteration 33571 / 39200) loss: 1.617766
(Iteration 33581 / 39200) loss: 1.104805
(Iteration 33591 / 39200) loss: 1.282877
(Iteration 33601 / 39200) loss: 1.368874
(Iteration 33611 / 39200) loss: 1.727540
(Iteration 33621 / 39200) loss: 1.351535
(Iteration 33631 / 39200) loss: 1.595162
(Iteration 33641 / 39200) loss: 1.262440
(Iteration 33651 / 39200) loss: 1.181607
(Iteration 33661 / 39200) loss: 1.765467
(Iteration 33671 / 39200) loss: 1.162778
(Iteration 33681 / 39200) loss: 1.164708
(Iteration 33691 / 39200) loss: 1.611934
(Iteration 33701 / 39200) loss: 1.347768
(Iteration 33711 / 39200) loss: 1.331425
(Iteration 33721 / 39200) loss: 1.576947
(Iteration 33731 / 39200) loss: 1.315464
(Iteration 33741 / 39200) loss: 1.723123
(Iteration 33751 / 39200) loss: 1.552223
(Iteration 33761 / 39200) loss: 1.200461
(Iteration 33771 / 39200) loss: 1.470509
(Iteration 33781 / 39200) loss: 1.564877
(Iteration 33791 / 39200) loss: 1.444532
(Iteration 33801 / 39200) loss: 1.536037
(Iteration 33811 / 39200) loss: 1.857402
(Iteration 33821 / 39200) loss: 2.077288
(Iteration 33831 / 39200) loss: 1.338399
(Iteration 33841 / 39200) loss: 1.407809
(Iteration 33851 / 39200) loss: 1.287553
(Iteration 33861 / 39200) loss: 1.765226
(Iteration 33871 / 39200) loss: 1.254304
(Iteration 33881 / 39200) loss: 1.240939
(Iteration 33891 / 39200) loss: 1.276626

(Iteration 33901 / 39200) loss: 1.220311
(Iteration 33911 / 39200) loss: 1.239515
(Iteration 33921 / 39200) loss: 1.603959
(Iteration 33931 / 39200) loss: 1.785638
(Iteration 33941 / 39200) loss: 1.129760
(Iteration 33951 / 39200) loss: 1.613883
(Iteration 33961 / 39200) loss: 1.956934
(Iteration 33971 / 39200) loss: 1.457364
(Iteration 33981 / 39200) loss: 1.629345
(Iteration 33991 / 39200) loss: 1.136728
(Iteration 34001 / 39200) loss: 1.421170
(Iteration 34011 / 39200) loss: 1.840021
(Iteration 34021 / 39200) loss: 1.411797
(Iteration 34031 / 39200) loss: 1.122551
(Iteration 34041 / 39200) loss: 1.570688
(Iteration 34051 / 39200) loss: 1.288669
(Iteration 34061 / 39200) loss: 1.932741
(Iteration 34071 / 39200) loss: 1.259864
(Iteration 34081 / 39200) loss: 1.413684
(Iteration 34091 / 39200) loss: 1.609715
(Iteration 34101 / 39200) loss: 1.301031
(Iteration 34111 / 39200) loss: 1.388095
(Iteration 34121 / 39200) loss: 1.478201
(Iteration 34131 / 39200) loss: 1.393414
(Iteration 34141 / 39200) loss: 1.293636
(Iteration 34151 / 39200) loss: 1.111992
(Iteration 34161 / 39200) loss: 1.513675
(Iteration 34171 / 39200) loss: 1.554869
(Iteration 34181 / 39200) loss: 1.672907
(Iteration 34191 / 39200) loss: 1.250455
(Iteration 34201 / 39200) loss: 1.370731
(Iteration 34211 / 39200) loss: 1.302020
(Iteration 34221 / 39200) loss: 1.028354
(Iteration 34231 / 39200) loss: 1.941055
(Iteration 34241 / 39200) loss: 1.354469
(Iteration 34251 / 39200) loss: 0.880108
(Iteration 34261 / 39200) loss: 1.298335
(Iteration 34271 / 39200) loss: 1.591502
(Iteration 34281 / 39200) loss: 1.113360
(Iteration 34291 / 39200) loss: 1.843741
(Iteration 34301 / 39200) loss: 1.043230
(Iteration 34311 / 39200) loss: 1.317755
(Iteration 34321 / 39200) loss: 1.682992
(Iteration 34331 / 39200) loss: 1.968398
(Iteration 34341 / 39200) loss: 1.475515
(Iteration 34351 / 39200) loss: 1.310756
(Iteration 34361 / 39200) loss: 1.509081
(Iteration 34371 / 39200) loss: 1.489485

(Iteration 34381 / 39200) loss: 1.138504
(Iteration 34391 / 39200) loss: 1.408117
(Iteration 34401 / 39200) loss: 1.278233
(Iteration 34411 / 39200) loss: 1.473269
(Iteration 34421 / 39200) loss: 1.387263
(Iteration 34431 / 39200) loss: 1.077668
(Iteration 34441 / 39200) loss: 0.992490
(Iteration 34451 / 39200) loss: 1.030528
(Iteration 34461 / 39200) loss: 1.048417
(Iteration 34471 / 39200) loss: 1.388708
(Iteration 34481 / 39200) loss: 1.656414
(Iteration 34491 / 39200) loss: 1.318843
(Iteration 34501 / 39200) loss: 1.056093
(Iteration 34511 / 39200) loss: 1.576570
(Iteration 34521 / 39200) loss: 1.534665
(Iteration 34531 / 39200) loss: 0.926592
(Iteration 34541 / 39200) loss: 1.021114
(Iteration 34551 / 39200) loss: 1.686007
(Iteration 34561 / 39200) loss: 1.047900
(Iteration 34571 / 39200) loss: 1.068219
(Iteration 34581 / 39200) loss: 0.923890
(Iteration 34591 / 39200) loss: 1.360921
(Iteration 34601 / 39200) loss: 1.224866
(Iteration 34611 / 39200) loss: 1.265494
(Iteration 34621 / 39200) loss: 1.679376
(Iteration 34631 / 39200) loss: 1.794382
(Iteration 34641 / 39200) loss: 1.479128
(Iteration 34651 / 39200) loss: 1.295554
(Iteration 34661 / 39200) loss: 1.086649
(Iteration 34671 / 39200) loss: 1.285242
(Iteration 34681 / 39200) loss: 1.224350
(Iteration 34691 / 39200) loss: 1.533330
(Iteration 34701 / 39200) loss: 1.612393
(Iteration 34711 / 39200) loss: 1.151485
(Iteration 34721 / 39200) loss: 1.246208
(Iteration 34731 / 39200) loss: 1.151357
(Iteration 34741 / 39200) loss: 1.476397
(Iteration 34751 / 39200) loss: 1.396982
(Iteration 34761 / 39200) loss: 1.398748
(Iteration 34771 / 39200) loss: 1.025490
(Iteration 34781 / 39200) loss: 1.383004
(Iteration 34791 / 39200) loss: 1.064496
(Iteration 34801 / 39200) loss: 1.263632
(Iteration 34811 / 39200) loss: 1.110429
(Iteration 34821 / 39200) loss: 1.511485
(Iteration 34831 / 39200) loss: 1.841732
(Iteration 34841 / 39200) loss: 1.356769
(Iteration 34851 / 39200) loss: 1.152702

(Iteration 34861 / 39200) loss: 1.488485
(Iteration 34871 / 39200) loss: 1.551925
(Iteration 34881 / 39200) loss: 1.474151
(Iteration 34891 / 39200) loss: 1.403958
(Iteration 34901 / 39200) loss: 1.047846
(Iteration 34911 / 39200) loss: 1.810967
(Iteration 34921 / 39200) loss: 1.243903
(Iteration 34931 / 39200) loss: 1.593680
(Iteration 34941 / 39200) loss: 1.126040
(Iteration 34951 / 39200) loss: 1.476934
(Iteration 34961 / 39200) loss: 1.578593
(Iteration 34971 / 39200) loss: 1.651688
(Iteration 34981 / 39200) loss: 1.495234
(Iteration 34991 / 39200) loss: 1.316702
(Iteration 35001 / 39200) loss: 1.228243
(Iteration 35011 / 39200) loss: 1.367636
(Iteration 35021 / 39200) loss: 1.151005
(Iteration 35031 / 39200) loss: 1.099369
(Iteration 35041 / 39200) loss: 1.318233
(Iteration 35051 / 39200) loss: 1.563269
(Iteration 35061 / 39200) loss: 1.256378
(Iteration 35071 / 39200) loss: 1.045073
(Iteration 35081 / 39200) loss: 1.080388
(Iteration 35091 / 39200) loss: 1.491918
(Iteration 35101 / 39200) loss: 1.224311
(Iteration 35111 / 39200) loss: 1.781640
(Iteration 35121 / 39200) loss: 1.164762
(Iteration 35131 / 39200) loss: 1.702069
(Iteration 35141 / 39200) loss: 1.557455
(Iteration 35151 / 39200) loss: 1.893426
(Iteration 35161 / 39200) loss: 1.138189
(Iteration 35171 / 39200) loss: 1.037242
(Iteration 35181 / 39200) loss: 1.402028
(Iteration 35191 / 39200) loss: 0.956069
(Iteration 35201 / 39200) loss: 1.321674
(Iteration 35211 / 39200) loss: 1.445114
(Iteration 35221 / 39200) loss: 1.172635
(Iteration 35231 / 39200) loss: 1.449093
(Iteration 35241 / 39200) loss: 1.295793
(Iteration 35251 / 39200) loss: 1.397078
(Iteration 35261 / 39200) loss: 1.129182
(Iteration 35271 / 39200) loss: 1.412542
(Epoch 18 / 20) train acc: 0.670000; val_acc: 0.512000
(Iteration 35281 / 39200) loss: 1.225606
(Iteration 35291 / 39200) loss: 1.895965
(Iteration 35301 / 39200) loss: 1.399398
(Iteration 35311 / 39200) loss: 1.441285
(Iteration 35321 / 39200) loss: 1.469913

(Iteration 35331 / 39200) loss: 1.445716
(Iteration 35341 / 39200) loss: 1.712362
(Iteration 35351 / 39200) loss: 1.644262
(Iteration 35361 / 39200) loss: 1.246665
(Iteration 35371 / 39200) loss: 1.236089
(Iteration 35381 / 39200) loss: 2.284403
(Iteration 35391 / 39200) loss: 1.429759
(Iteration 35401 / 39200) loss: 1.388216
(Iteration 35411 / 39200) loss: 1.506396
(Iteration 35421 / 39200) loss: 1.407712
(Iteration 35431 / 39200) loss: 1.565052
(Iteration 35441 / 39200) loss: 1.265294
(Iteration 35451 / 39200) loss: 1.058230
(Iteration 35461 / 39200) loss: 1.845396
(Iteration 35471 / 39200) loss: 1.301133
(Iteration 35481 / 39200) loss: 1.285305
(Iteration 35491 / 39200) loss: 1.657965
(Iteration 35501 / 39200) loss: 1.328694
(Iteration 35511 / 39200) loss: 1.386670
(Iteration 35521 / 39200) loss: 1.438965
(Iteration 35531 / 39200) loss: 1.623777
(Iteration 35541 / 39200) loss: 1.013342
(Iteration 35551 / 39200) loss: 0.946423
(Iteration 35561 / 39200) loss: 1.264486
(Iteration 35571 / 39200) loss: 1.292597
(Iteration 35581 / 39200) loss: 1.625782
(Iteration 35591 / 39200) loss: 1.288943
(Iteration 35601 / 39200) loss: 1.376212
(Iteration 35611 / 39200) loss: 1.149541
(Iteration 35621 / 39200) loss: 1.393844
(Iteration 35631 / 39200) loss: 1.171152
(Iteration 35641 / 39200) loss: 1.470383
(Iteration 35651 / 39200) loss: 1.628637
(Iteration 35661 / 39200) loss: 0.934352
(Iteration 35671 / 39200) loss: 1.741831
(Iteration 35681 / 39200) loss: 1.467913
(Iteration 35691 / 39200) loss: 1.336411
(Iteration 35701 / 39200) loss: 1.724660
(Iteration 35711 / 39200) loss: 1.352272
(Iteration 35721 / 39200) loss: 1.521505
(Iteration 35731 / 39200) loss: 1.408516
(Iteration 35741 / 39200) loss: 1.590162
(Iteration 35751 / 39200) loss: 1.472534
(Iteration 35761 / 39200) loss: 1.495359
(Iteration 35771 / 39200) loss: 1.567137
(Iteration 35781 / 39200) loss: 1.173989
(Iteration 35791 / 39200) loss: 1.331051
(Iteration 35801 / 39200) loss: 1.430236

(Iteration 35811 / 39200) loss: 1.505411
(Iteration 35821 / 39200) loss: 1.295095
(Iteration 35831 / 39200) loss: 1.238169
(Iteration 35841 / 39200) loss: 1.284560
(Iteration 35851 / 39200) loss: 1.174982
(Iteration 35861 / 39200) loss: 1.316138
(Iteration 35871 / 39200) loss: 1.184711
(Iteration 35881 / 39200) loss: 1.321708
(Iteration 35891 / 39200) loss: 1.781030
(Iteration 35901 / 39200) loss: 1.527835
(Iteration 35911 / 39200) loss: 2.032150
(Iteration 35921 / 39200) loss: 1.135064
(Iteration 35931 / 39200) loss: 1.090170
(Iteration 35941 / 39200) loss: 1.123720
(Iteration 35951 / 39200) loss: 1.255240
(Iteration 35961 / 39200) loss: 1.367221
(Iteration 35971 / 39200) loss: 1.040101
(Iteration 35981 / 39200) loss: 1.332315
(Iteration 35991 / 39200) loss: 1.448980
(Iteration 36001 / 39200) loss: 1.343604
(Iteration 36011 / 39200) loss: 1.633878
(Iteration 36021 / 39200) loss: 1.428155
(Iteration 36031 / 39200) loss: 1.549083
(Iteration 36041 / 39200) loss: 1.432591
(Iteration 36051 / 39200) loss: 0.902819
(Iteration 36061 / 39200) loss: 0.914946
(Iteration 36071 / 39200) loss: 1.332993
(Iteration 36081 / 39200) loss: 1.081582
(Iteration 36091 / 39200) loss: 1.141321
(Iteration 36101 / 39200) loss: 1.341337
(Iteration 36111 / 39200) loss: 1.192941
(Iteration 36121 / 39200) loss: 1.099135
(Iteration 36131 / 39200) loss: 1.149854
(Iteration 36141 / 39200) loss: 0.990641
(Iteration 36151 / 39200) loss: 1.208458
(Iteration 36161 / 39200) loss: 1.817471
(Iteration 36171 / 39200) loss: 1.514124
(Iteration 36181 / 39200) loss: 1.367647
(Iteration 36191 / 39200) loss: 1.507466
(Iteration 36201 / 39200) loss: 0.852818
(Iteration 36211 / 39200) loss: 1.272737
(Iteration 36221 / 39200) loss: 1.429023
(Iteration 36231 / 39200) loss: 1.097122
(Iteration 36241 / 39200) loss: 1.014635
(Iteration 36251 / 39200) loss: 1.330163
(Iteration 36261 / 39200) loss: 1.472393
(Iteration 36271 / 39200) loss: 1.112441
(Iteration 36281 / 39200) loss: 1.211574

(Iteration 36291 / 39200) loss: 1.111174
(Iteration 36301 / 39200) loss: 0.968888
(Iteration 36311 / 39200) loss: 1.734837
(Iteration 36321 / 39200) loss: 1.170549
(Iteration 36331 / 39200) loss: 1.436227
(Iteration 36341 / 39200) loss: 1.289532
(Iteration 36351 / 39200) loss: 1.306724
(Iteration 36361 / 39200) loss: 1.304257
(Iteration 36371 / 39200) loss: 1.386643
(Iteration 36381 / 39200) loss: 1.520250
(Iteration 36391 / 39200) loss: 1.035260
(Iteration 36401 / 39200) loss: 0.953589
(Iteration 36411 / 39200) loss: 1.218262
(Iteration 36421 / 39200) loss: 1.495975
(Iteration 36431 / 39200) loss: 1.186196
(Iteration 36441 / 39200) loss: 1.499923
(Iteration 36451 / 39200) loss: 1.016640
(Iteration 36461 / 39200) loss: 1.378885
(Iteration 36471 / 39200) loss: 1.388469
(Iteration 36481 / 39200) loss: 1.421684
(Iteration 36491 / 39200) loss: 1.130094
(Iteration 36501 / 39200) loss: 1.242169
(Iteration 36511 / 39200) loss: 1.371422
(Iteration 36521 / 39200) loss: 1.142581
(Iteration 36531 / 39200) loss: 1.480327
(Iteration 36541 / 39200) loss: 1.403910
(Iteration 36551 / 39200) loss: 1.349022
(Iteration 36561 / 39200) loss: 1.234883
(Iteration 36571 / 39200) loss: 1.185143
(Iteration 36581 / 39200) loss: 1.247302
(Iteration 36591 / 39200) loss: 1.545919
(Iteration 36601 / 39200) loss: 1.229056
(Iteration 36611 / 39200) loss: 1.268782
(Iteration 36621 / 39200) loss: 1.372575
(Iteration 36631 / 39200) loss: 1.034298
(Iteration 36641 / 39200) loss: 1.580215
(Iteration 36651 / 39200) loss: 1.464553
(Iteration 36661 / 39200) loss: 1.330021
(Iteration 36671 / 39200) loss: 1.427573
(Iteration 36681 / 39200) loss: 1.260671
(Iteration 36691 / 39200) loss: 1.325370
(Iteration 36701 / 39200) loss: 1.391392
(Iteration 36711 / 39200) loss: 1.728631
(Iteration 36721 / 39200) loss: 1.475669
(Iteration 36731 / 39200) loss: 1.210308
(Iteration 36741 / 39200) loss: 1.492148
(Iteration 36751 / 39200) loss: 1.545142
(Iteration 36761 / 39200) loss: 1.569428

(Iteration 36771 / 39200) loss: 1.235574
(Iteration 36781 / 39200) loss: 0.952038
(Iteration 36791 / 39200) loss: 1.446102
(Iteration 36801 / 39200) loss: 1.004139
(Iteration 36811 / 39200) loss: 1.241169
(Iteration 36821 / 39200) loss: 1.309179
(Iteration 36831 / 39200) loss: 1.132684
(Iteration 36841 / 39200) loss: 1.465394
(Iteration 36851 / 39200) loss: 1.998687
(Iteration 36861 / 39200) loss: 1.182220
(Iteration 36871 / 39200) loss: 1.436514
(Iteration 36881 / 39200) loss: 1.796593
(Iteration 36891 / 39200) loss: 1.548816
(Iteration 36901 / 39200) loss: 1.230543
(Iteration 36911 / 39200) loss: 1.522822
(Iteration 36921 / 39200) loss: 1.380526
(Iteration 36931 / 39200) loss: 0.996776
(Iteration 36941 / 39200) loss: 1.421419
(Iteration 36951 / 39200) loss: 1.364397
(Iteration 36961 / 39200) loss: 1.397762
(Iteration 36971 / 39200) loss: 1.437129
(Iteration 36981 / 39200) loss: 1.491834
(Iteration 36991 / 39200) loss: 1.228712
(Iteration 37001 / 39200) loss: 1.218275
(Iteration 37011 / 39200) loss: 1.490757
(Iteration 37021 / 39200) loss: 1.505741
(Iteration 37031 / 39200) loss: 1.319781
(Iteration 37041 / 39200) loss: 1.280337
(Iteration 37051 / 39200) loss: 1.225162
(Iteration 37061 / 39200) loss: 1.617855
(Iteration 37071 / 39200) loss: 1.138075
(Iteration 37081 / 39200) loss: 1.548997
(Iteration 37091 / 39200) loss: 1.429580
(Iteration 37101 / 39200) loss: 1.124383
(Iteration 37111 / 39200) loss: 1.081517
(Iteration 37121 / 39200) loss: 1.104256
(Iteration 37131 / 39200) loss: 1.317160
(Iteration 37141 / 39200) loss: 1.300919
(Iteration 37151 / 39200) loss: 1.139385
(Iteration 37161 / 39200) loss: 1.389987
(Iteration 37171 / 39200) loss: 1.267207
(Iteration 37181 / 39200) loss: 0.892691
(Iteration 37191 / 39200) loss: 1.398945
(Iteration 37201 / 39200) loss: 1.098181
(Iteration 37211 / 39200) loss: 1.274403
(Iteration 37221 / 39200) loss: 1.383749
(Iteration 37231 / 39200) loss: 1.148965
(Epoch 19 / 20) train acc: 0.689000; val_acc: 0.523000

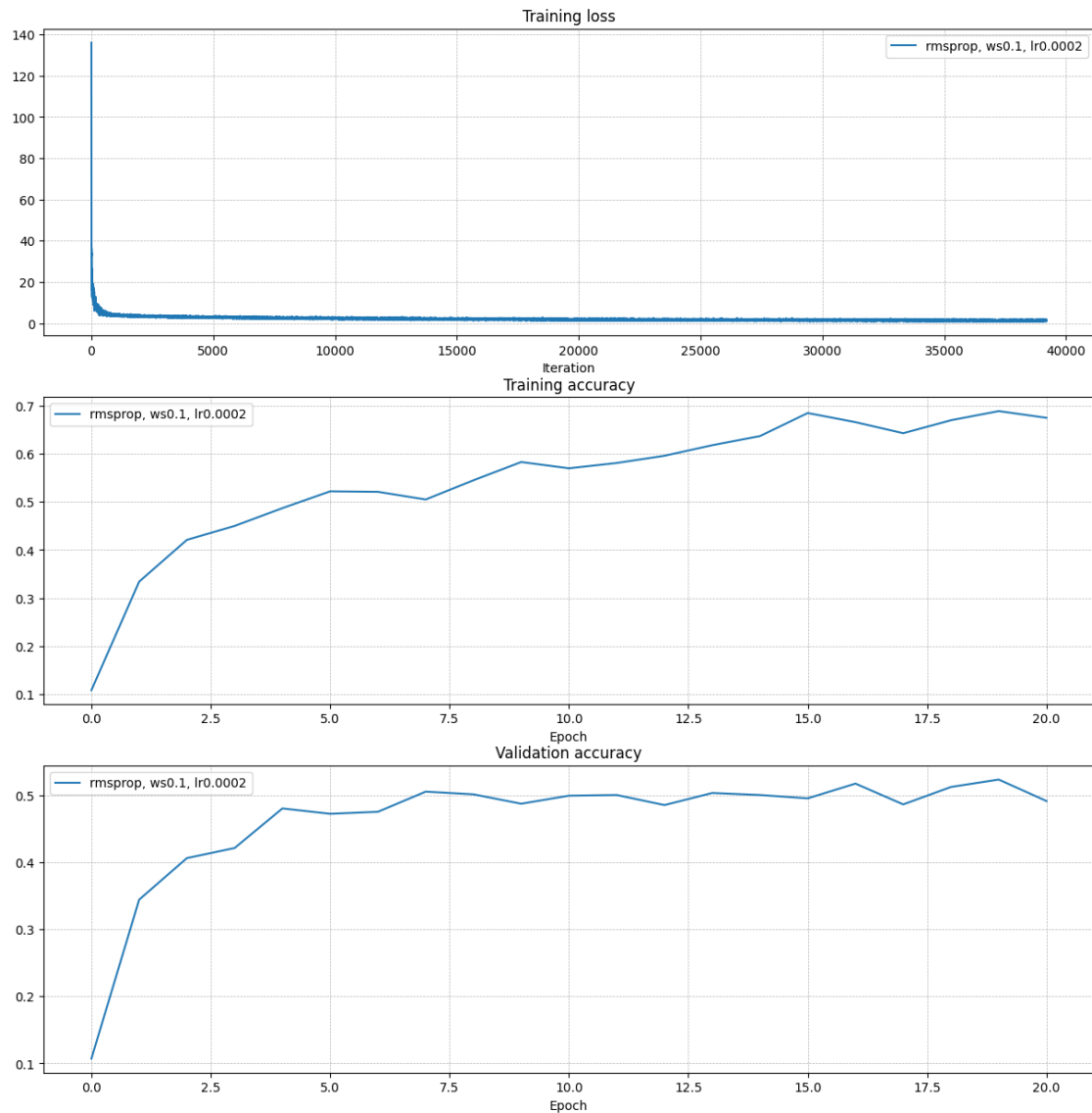
(Iteration 37241 / 39200) loss: 1.540813
(Iteration 37251 / 39200) loss: 1.404514
(Iteration 37261 / 39200) loss: 1.429719
(Iteration 37271 / 39200) loss: 1.248999
(Iteration 37281 / 39200) loss: 1.420072
(Iteration 37291 / 39200) loss: 1.369141
(Iteration 37301 / 39200) loss: 0.982231
(Iteration 37311 / 39200) loss: 1.679983
(Iteration 37321 / 39200) loss: 1.139177
(Iteration 37331 / 39200) loss: 1.480757
(Iteration 37341 / 39200) loss: 1.307474
(Iteration 37351 / 39200) loss: 1.287505
(Iteration 37361 / 39200) loss: 1.176540
(Iteration 37371 / 39200) loss: 1.283091
(Iteration 37381 / 39200) loss: 1.123834
(Iteration 37391 / 39200) loss: 1.190769
(Iteration 37401 / 39200) loss: 1.177931
(Iteration 37411 / 39200) loss: 1.206836
(Iteration 37421 / 39200) loss: 1.280327
(Iteration 37431 / 39200) loss: 1.427111
(Iteration 37441 / 39200) loss: 1.332148
(Iteration 37451 / 39200) loss: 1.804183
(Iteration 37461 / 39200) loss: 0.820270
(Iteration 37471 / 39200) loss: 1.395304
(Iteration 37481 / 39200) loss: 1.202187
(Iteration 37491 / 39200) loss: 1.761382
(Iteration 37501 / 39200) loss: 1.502791
(Iteration 37511 / 39200) loss: 1.442009
(Iteration 37521 / 39200) loss: 1.049887
(Iteration 37531 / 39200) loss: 1.220101
(Iteration 37541 / 39200) loss: 1.327520
(Iteration 37551 / 39200) loss: 1.107370
(Iteration 37561 / 39200) loss: 1.307099
(Iteration 37571 / 39200) loss: 1.427611
(Iteration 37581 / 39200) loss: 1.454841
(Iteration 37591 / 39200) loss: 1.310789
(Iteration 37601 / 39200) loss: 1.269976
(Iteration 37611 / 39200) loss: 1.433239
(Iteration 37621 / 39200) loss: 1.200975
(Iteration 37631 / 39200) loss: 1.425364
(Iteration 37641 / 39200) loss: 1.419344
(Iteration 37651 / 39200) loss: 0.954258
(Iteration 37661 / 39200) loss: 1.027040
(Iteration 37671 / 39200) loss: 1.425516
(Iteration 37681 / 39200) loss: 1.015643
(Iteration 37691 / 39200) loss: 1.478010
(Iteration 37701 / 39200) loss: 1.391279
(Iteration 37711 / 39200) loss: 1.595598

(Iteration 37721 / 39200) loss: 1.325682
(Iteration 37731 / 39200) loss: 1.574496
(Iteration 37741 / 39200) loss: 1.198783
(Iteration 37751 / 39200) loss: 1.438778
(Iteration 37761 / 39200) loss: 0.939028
(Iteration 37771 / 39200) loss: 1.555548
(Iteration 37781 / 39200) loss: 1.204081
(Iteration 37791 / 39200) loss: 1.446077
(Iteration 37801 / 39200) loss: 0.879029
(Iteration 37811 / 39200) loss: 1.521450
(Iteration 37821 / 39200) loss: 1.309468
(Iteration 37831 / 39200) loss: 1.488985
(Iteration 37841 / 39200) loss: 1.152506
(Iteration 37851 / 39200) loss: 1.170340
(Iteration 37861 / 39200) loss: 1.297241
(Iteration 37871 / 39200) loss: 1.483188
(Iteration 37881 / 39200) loss: 0.939008
(Iteration 37891 / 39200) loss: 1.063457
(Iteration 37901 / 39200) loss: 1.297979
(Iteration 37911 / 39200) loss: 1.261515
(Iteration 37921 / 39200) loss: 1.434584
(Iteration 37931 / 39200) loss: 1.217631
(Iteration 37941 / 39200) loss: 1.345131
(Iteration 37951 / 39200) loss: 1.463569
(Iteration 37961 / 39200) loss: 1.510212
(Iteration 37971 / 39200) loss: 0.913550
(Iteration 37981 / 39200) loss: 1.110152
(Iteration 37991 / 39200) loss: 1.242765
(Iteration 38001 / 39200) loss: 1.147283
(Iteration 38011 / 39200) loss: 1.463941
(Iteration 38021 / 39200) loss: 0.934749
(Iteration 38031 / 39200) loss: 1.583440
(Iteration 38041 / 39200) loss: 1.103662
(Iteration 38051 / 39200) loss: 1.231475
(Iteration 38061 / 39200) loss: 1.152097
(Iteration 38071 / 39200) loss: 1.284701
(Iteration 38081 / 39200) loss: 0.989990
(Iteration 38091 / 39200) loss: 1.548155
(Iteration 38101 / 39200) loss: 1.498939
(Iteration 38111 / 39200) loss: 1.372524
(Iteration 38121 / 39200) loss: 1.152029
(Iteration 38131 / 39200) loss: 1.195819
(Iteration 38141 / 39200) loss: 1.446445
(Iteration 38151 / 39200) loss: 1.152848
(Iteration 38161 / 39200) loss: 1.217229
(Iteration 38171 / 39200) loss: 1.228378
(Iteration 38181 / 39200) loss: 1.821883
(Iteration 38191 / 39200) loss: 1.216294

(Iteration 38201 / 39200) loss: 1.602858
(Iteration 38211 / 39200) loss: 1.497350
(Iteration 38221 / 39200) loss: 1.081050
(Iteration 38231 / 39200) loss: 1.522677
(Iteration 38241 / 39200) loss: 1.296272
(Iteration 38251 / 39200) loss: 1.381723
(Iteration 38261 / 39200) loss: 1.184145
(Iteration 38271 / 39200) loss: 0.994404
(Iteration 38281 / 39200) loss: 1.289977
(Iteration 38291 / 39200) loss: 1.455169
(Iteration 38301 / 39200) loss: 1.274690
(Iteration 38311 / 39200) loss: 1.318350
(Iteration 38321 / 39200) loss: 1.094901
(Iteration 38331 / 39200) loss: 1.081774
(Iteration 38341 / 39200) loss: 1.414570
(Iteration 38351 / 39200) loss: 1.072582
(Iteration 38361 / 39200) loss: 1.308743
(Iteration 38371 / 39200) loss: 1.569375
(Iteration 38381 / 39200) loss: 1.438878
(Iteration 38391 / 39200) loss: 1.283752
(Iteration 38401 / 39200) loss: 1.765718
(Iteration 38411 / 39200) loss: 0.970415
(Iteration 38421 / 39200) loss: 1.096267
(Iteration 38431 / 39200) loss: 1.178346
(Iteration 38441 / 39200) loss: 1.706610
(Iteration 38451 / 39200) loss: 1.352950
(Iteration 38461 / 39200) loss: 1.171170
(Iteration 38471 / 39200) loss: 0.944163
(Iteration 38481 / 39200) loss: 1.367759
(Iteration 38491 / 39200) loss: 1.194295
(Iteration 38501 / 39200) loss: 1.440167
(Iteration 38511 / 39200) loss: 1.377639
(Iteration 38521 / 39200) loss: 1.181505
(Iteration 38531 / 39200) loss: 1.113875
(Iteration 38541 / 39200) loss: 1.322884
(Iteration 38551 / 39200) loss: 1.562516
(Iteration 38561 / 39200) loss: 1.316692
(Iteration 38571 / 39200) loss: 1.607765
(Iteration 38581 / 39200) loss: 1.242269
(Iteration 38591 / 39200) loss: 1.512949
(Iteration 38601 / 39200) loss: 2.114089
(Iteration 38611 / 39200) loss: 1.377931
(Iteration 38621 / 39200) loss: 1.203711
(Iteration 38631 / 39200) loss: 1.131229
(Iteration 38641 / 39200) loss: 1.478749
(Iteration 38651 / 39200) loss: 1.307594
(Iteration 38661 / 39200) loss: 1.504517
(Iteration 38671 / 39200) loss: 1.219543

(Iteration 38681 / 39200) loss: 1.482611
(Iteration 38691 / 39200) loss: 1.226932
(Iteration 38701 / 39200) loss: 1.197373
(Iteration 38711 / 39200) loss: 1.339526
(Iteration 38721 / 39200) loss: 0.823320
(Iteration 38731 / 39200) loss: 1.340667
(Iteration 38741 / 39200) loss: 1.121909
(Iteration 38751 / 39200) loss: 1.802579
(Iteration 38761 / 39200) loss: 1.089196
(Iteration 38771 / 39200) loss: 1.622834
(Iteration 38781 / 39200) loss: 1.005175
(Iteration 38791 / 39200) loss: 1.418469
(Iteration 38801 / 39200) loss: 1.651677
(Iteration 38811 / 39200) loss: 1.473787
(Iteration 38821 / 39200) loss: 1.127300
(Iteration 38831 / 39200) loss: 1.267822
(Iteration 38841 / 39200) loss: 1.376162
(Iteration 38851 / 39200) loss: 1.010704
(Iteration 38861 / 39200) loss: 1.341170
(Iteration 38871 / 39200) loss: 1.323551
(Iteration 38881 / 39200) loss: 1.557164
(Iteration 38891 / 39200) loss: 1.272004
(Iteration 38901 / 39200) loss: 1.162999
(Iteration 38911 / 39200) loss: 1.193100
(Iteration 38921 / 39200) loss: 1.372161
(Iteration 38931 / 39200) loss: 1.377671
(Iteration 38941 / 39200) loss: 1.732649
(Iteration 38951 / 39200) loss: 1.151335
(Iteration 38961 / 39200) loss: 1.573195
(Iteration 38971 / 39200) loss: 1.354406
(Iteration 38981 / 39200) loss: 1.326339
(Iteration 38991 / 39200) loss: 1.537208
(Iteration 39001 / 39200) loss: 1.499543
(Iteration 39011 / 39200) loss: 1.080217
(Iteration 39021 / 39200) loss: 1.198469
(Iteration 39031 / 39200) loss: 1.218066
(Iteration 39041 / 39200) loss: 1.214402
(Iteration 39051 / 39200) loss: 1.403896
(Iteration 39061 / 39200) loss: 1.361195
(Iteration 39071 / 39200) loss: 1.429918
(Iteration 39081 / 39200) loss: 1.399648
(Iteration 39091 / 39200) loss: 1.210354
(Iteration 39101 / 39200) loss: 1.196619
(Iteration 39111 / 39200) loss: 1.036373
(Iteration 39121 / 39200) loss: 1.230140
(Iteration 39131 / 39200) loss: 1.620899
(Iteration 39141 / 39200) loss: 1.259165
(Iteration 39151 / 39200) loss: 1.412158

(Iteration 39161 / 39200) loss: 1.391319
(Iteration 39171 / 39200) loss: 1.048255
(Iteration 39181 / 39200) loss: 1.232175
(Iteration 39191 / 39200) loss: 1.387487
(Epoch 20 / 20) train acc: 0.675000; val_acc: 0.491000
weight_scale 1.000000e-01 lr 2.000000e-04 rule rmsprop train accuracy: 0.675000
val accuracy: 0.523000
best val accuracy achieved: 0.523000
weight_scale 1.000000e-01 lr 2.000000e-04 rule rmsprop train accuracy: 0.675000
val accuracy: 0.523000
best val accuracy achieved: 0.523000



4 Evaluate baseline model from A2

Evaluate above baseline model.

```
[10]: best_model_basic = best_model

[11]: y_test_pred = np.argmax(best_model_basic.loss(data['X_test']), axis=1)
      y_val_pred = np.argmax(best_model_basic.loss(data['X_val']), axis=1)
      print('Validation set accuracy: ', (y_val_pred == data['y_val']).mean())
      print('Test set accuracy: ', (y_test_pred == data['y_test']).mean())
```

Validation set accuracy: 0.523

Test set accuracy: 0.504

5 Train improved model

Design a new model in `FullyConnectedNetImproved` in the file `cse493g1/classifiers/fc_net.py`. You can start by having `FullyConnectedNetImproved` be the same design as `FullyConnectedNetBasic`. Next, complete the `BatchNormalization.ipynb` and `Dropout.ipynb` notebooks. Then return to this notebook and complete `FullyConnectedNetImproved` by adding in batchnorm and dropout. Try to beat the accuracy of your baseline model! You may have to adjust your hyperparameters.

```
[20]: best_model_improved = None

#####
# TODO: Train the best FullyConnectedNetImproved that you can on CIFAR-10. You
# might #
# find batch/layer normalization and dropout useful. Store your best model in #
# the best_model_improved variable.
# #
#####
# *****START OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****

weight_scale = [1e-1]
learning_rate = [1e-3]
update_rule = ['adam']
reg = [0]

results = {}
best_val = 0
best_solver = None
solvers = {}

for i in weight_scale:
    for j in learning_rate:
        for rule in update_rule:
            model = FullyConnectedNetImproved(
```



```

        [100, 100, 100, 100, 100],
        input_dim=3 * 32 * 32,
        num_classes=10,
        dropout_keep_ratio=0.75,
        normalization='batchnorm',
        reg=reg[0],
        weight_scale=i,
        dtype=np.float32,
        seed=None,
    )
    solver = Solver(
        model,
        data,
        print_every=10,
        num_epochs=20,
        batch_size=25,
        update_rule= rule,
        optim_config={"learning_rate": j},
    )
    solver.train()

    solvers[(i,j,rule)] = solver

    if solver.best_val_acc > best_val:
        best_model_improved = model
        best_solver = solver
        best_val = solver.best_val_acc

    results[(i,j,rule)] = (solver.train_acc_history[len(solver.
↪train_acc_history) - 1], solver.best_val_acc)

# Print out results.
for i, j, rule in sorted(results):
    train_accuracy, val_accuracy = results[(i,j,rule)]
    print('weight_scale %e lr %e rule %s train accuracy: %f val accuracy: %f'↪
↪% (
        i, j, rule, train_accuracy, val_accuracy))

print('best val accuracy achieved: %f' % best_val)

fig, axes = plt.subplots(3, 1, figsize=(15, 15))

axes[0].set_title('Training loss')
axes[0].set_xlabel('Iteration')
axes[1].set_title('Training accuracy')
axes[1].set_xlabel('Epoch')

```

```

axes[2].set_title('Validation accuracy')
axes[2].set_xlabel('Epoch')

for (i,j, update_rule), solver in solvers.items():
    axes[0].plot(solver.loss_history, label=f"{update_rule}, ws{i}, lr{j}")
    axes[1].plot(solver.train_acc_history, label=f"{update_rule}, ws{i}, lr{j}")
    axes[2].plot(solver.val_acc_history, label=f"{update_rule}, ws{i}, lr{j}")

for ax in axes:
    ax.legend(loc='best', ncol=4)
    ax.grid(linestyle='--', linewidth=0.5)

plt.show()

# *****END OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****
#####
#                                     END OF YOUR CODE                                     #
#####

```

```

(Iteration 1 / 39200) loss: 2.525122
(Epoch 0 / 20) train acc: 0.122000; val_acc: 0.089000
(Iteration 11 / 39200) loss: 2.467785
(Iteration 21 / 39200) loss: 2.616635
(Iteration 31 / 39200) loss: 2.337299
(Iteration 41 / 39200) loss: 2.354122
(Iteration 51 / 39200) loss: 2.346690
(Iteration 61 / 39200) loss: 2.435356
(Iteration 71 / 39200) loss: 2.226873
(Iteration 81 / 39200) loss: 2.168396
(Iteration 91 / 39200) loss: 2.203394
(Iteration 101 / 39200) loss: 2.301686
(Iteration 111 / 39200) loss: 2.002435
(Iteration 121 / 39200) loss: 2.528594
(Iteration 131 / 39200) loss: 2.359258
(Iteration 141 / 39200) loss: 2.417357
(Iteration 151 / 39200) loss: 2.048365
(Iteration 161 / 39200) loss: 2.222106
(Iteration 171 / 39200) loss: 2.155491
(Iteration 181 / 39200) loss: 2.210556
(Iteration 191 / 39200) loss: 2.106343
(Iteration 201 / 39200) loss: 2.336861
(Iteration 211 / 39200) loss: 2.116210
(Iteration 221 / 39200) loss: 2.276643
(Iteration 231 / 39200) loss: 2.143781
(Iteration 241 / 39200) loss: 2.310002
(Iteration 251 / 39200) loss: 2.466693

```

(Iteration 261 / 39200) loss: 2.266823
(Iteration 271 / 39200) loss: 2.073455
(Iteration 281 / 39200) loss: 2.066790
(Iteration 291 / 39200) loss: 2.050672
(Iteration 301 / 39200) loss: 2.344767
(Iteration 311 / 39200) loss: 2.114384
(Iteration 321 / 39200) loss: 2.206529
(Iteration 331 / 39200) loss: 2.255680
(Iteration 341 / 39200) loss: 2.027948
(Iteration 351 / 39200) loss: 2.254711
(Iteration 361 / 39200) loss: 2.474295
(Iteration 371 / 39200) loss: 2.165783
(Iteration 381 / 39200) loss: 2.291080
(Iteration 391 / 39200) loss: 2.152227
(Iteration 401 / 39200) loss: 1.802373
(Iteration 411 / 39200) loss: 2.270864
(Iteration 421 / 39200) loss: 1.996380
(Iteration 431 / 39200) loss: 1.860764
(Iteration 441 / 39200) loss: 2.235544
(Iteration 451 / 39200) loss: 2.024169
(Iteration 461 / 39200) loss: 2.062680
(Iteration 471 / 39200) loss: 2.311384
(Iteration 481 / 39200) loss: 2.170448
(Iteration 491 / 39200) loss: 1.965757
(Iteration 501 / 39200) loss: 1.911707
(Iteration 511 / 39200) loss: 2.194529
(Iteration 521 / 39200) loss: 1.885085
(Iteration 531 / 39200) loss: 1.975254
(Iteration 541 / 39200) loss: 1.871400
(Iteration 551 / 39200) loss: 1.887640
(Iteration 561 / 39200) loss: 1.769483
(Iteration 571 / 39200) loss: 1.989507
(Iteration 581 / 39200) loss: 2.131859
(Iteration 591 / 39200) loss: 1.998037
(Iteration 601 / 39200) loss: 2.043694
(Iteration 611 / 39200) loss: 1.982306
(Iteration 621 / 39200) loss: 2.021171
(Iteration 631 / 39200) loss: 2.124259
(Iteration 641 / 39200) loss: 1.967371
(Iteration 651 / 39200) loss: 1.728728
(Iteration 661 / 39200) loss: 1.987389
(Iteration 671 / 39200) loss: 1.905807
(Iteration 681 / 39200) loss: 1.865888
(Iteration 691 / 39200) loss: 1.710499
(Iteration 701 / 39200) loss: 1.798513
(Iteration 711 / 39200) loss: 1.820674
(Iteration 721 / 39200) loss: 1.967337
(Iteration 731 / 39200) loss: 2.187136

(Iteration 741 / 39200) loss: 1.633906
(Iteration 751 / 39200) loss: 2.208981
(Iteration 761 / 39200) loss: 1.743122
(Iteration 771 / 39200) loss: 1.876759
(Iteration 781 / 39200) loss: 1.701047
(Iteration 791 / 39200) loss: 1.742663
(Iteration 801 / 39200) loss: 2.432476
(Iteration 811 / 39200) loss: 1.971093
(Iteration 821 / 39200) loss: 2.225596
(Iteration 831 / 39200) loss: 1.984797
(Iteration 841 / 39200) loss: 1.860005
(Iteration 851 / 39200) loss: 1.770427
(Iteration 861 / 39200) loss: 2.111076
(Iteration 871 / 39200) loss: 1.936362
(Iteration 881 / 39200) loss: 2.085451
(Iteration 891 / 39200) loss: 2.051184
(Iteration 901 / 39200) loss: 1.764230
(Iteration 911 / 39200) loss: 1.942426
(Iteration 921 / 39200) loss: 1.976438
(Iteration 931 / 39200) loss: 1.861229
(Iteration 941 / 39200) loss: 1.918548
(Iteration 951 / 39200) loss: 1.810389
(Iteration 961 / 39200) loss: 2.058438
(Iteration 971 / 39200) loss: 1.793838
(Iteration 981 / 39200) loss: 1.796357
(Iteration 991 / 39200) loss: 1.904686
(Iteration 1001 / 39200) loss: 1.952474
(Iteration 1011 / 39200) loss: 2.007788
(Iteration 1021 / 39200) loss: 1.921345
(Iteration 1031 / 39200) loss: 1.869017
(Iteration 1041 / 39200) loss: 1.755141
(Iteration 1051 / 39200) loss: 2.269666
(Iteration 1061 / 39200) loss: 1.663638
(Iteration 1071 / 39200) loss: 2.019454
(Iteration 1081 / 39200) loss: 1.807202
(Iteration 1091 / 39200) loss: 1.735851
(Iteration 1101 / 39200) loss: 1.638167
(Iteration 1111 / 39200) loss: 1.756730
(Iteration 1121 / 39200) loss: 1.998030
(Iteration 1131 / 39200) loss: 1.964845
(Iteration 1141 / 39200) loss: 1.736503
(Iteration 1151 / 39200) loss: 1.923231
(Iteration 1161 / 39200) loss: 2.148853
(Iteration 1171 / 39200) loss: 1.708484
(Iteration 1181 / 39200) loss: 1.924345
(Iteration 1191 / 39200) loss: 1.514818
(Iteration 1201 / 39200) loss: 1.971932
(Iteration 1211 / 39200) loss: 2.265229

(Iteration 1221 / 39200) loss: 2.148307
(Iteration 1231 / 39200) loss: 1.880980
(Iteration 1241 / 39200) loss: 1.876430
(Iteration 1251 / 39200) loss: 1.905483
(Iteration 1261 / 39200) loss: 1.783514
(Iteration 1271 / 39200) loss: 2.082941
(Iteration 1281 / 39200) loss: 1.928526
(Iteration 1291 / 39200) loss: 1.659647
(Iteration 1301 / 39200) loss: 1.834555
(Iteration 1311 / 39200) loss: 1.974971
(Iteration 1321 / 39200) loss: 1.997263
(Iteration 1331 / 39200) loss: 1.818496
(Iteration 1341 / 39200) loss: 2.188172
(Iteration 1351 / 39200) loss: 2.257936
(Iteration 1361 / 39200) loss: 2.187794
(Iteration 1371 / 39200) loss: 1.916799
(Iteration 1381 / 39200) loss: 1.813109
(Iteration 1391 / 39200) loss: 2.062154
(Iteration 1401 / 39200) loss: 2.032507
(Iteration 1411 / 39200) loss: 1.959909
(Iteration 1421 / 39200) loss: 2.008422
(Iteration 1431 / 39200) loss: 1.908402
(Iteration 1441 / 39200) loss: 1.605718
(Iteration 1451 / 39200) loss: 2.103060
(Iteration 1461 / 39200) loss: 2.041337
(Iteration 1471 / 39200) loss: 1.950203
(Iteration 1481 / 39200) loss: 1.939719
(Iteration 1491 / 39200) loss: 1.868248
(Iteration 1501 / 39200) loss: 1.799289
(Iteration 1511 / 39200) loss: 2.008444
(Iteration 1521 / 39200) loss: 1.802304
(Iteration 1531 / 39200) loss: 1.927966
(Iteration 1541 / 39200) loss: 1.655177
(Iteration 1551 / 39200) loss: 2.018063
(Iteration 1561 / 39200) loss: 2.124274
(Iteration 1571 / 39200) loss: 1.617355
(Iteration 1581 / 39200) loss: 1.942204
(Iteration 1591 / 39200) loss: 2.019558
(Iteration 1601 / 39200) loss: 1.819257
(Iteration 1611 / 39200) loss: 1.752061
(Iteration 1621 / 39200) loss: 1.774109
(Iteration 1631 / 39200) loss: 1.806050
(Iteration 1641 / 39200) loss: 1.712379
(Iteration 1651 / 39200) loss: 2.009524
(Iteration 1661 / 39200) loss: 1.699480
(Iteration 1671 / 39200) loss: 2.207040
(Iteration 1681 / 39200) loss: 2.002931
(Iteration 1691 / 39200) loss: 1.688414

(Iteration 1701 / 39200) loss: 1.880068
(Iteration 1711 / 39200) loss: 1.990169
(Iteration 1721 / 39200) loss: 2.127602
(Iteration 1731 / 39200) loss: 1.894949
(Iteration 1741 / 39200) loss: 1.744735
(Iteration 1751 / 39200) loss: 1.879007
(Iteration 1761 / 39200) loss: 2.229259
(Iteration 1771 / 39200) loss: 1.702454
(Iteration 1781 / 39200) loss: 1.859544
(Iteration 1791 / 39200) loss: 2.104332
(Iteration 1801 / 39200) loss: 1.813196
(Iteration 1811 / 39200) loss: 1.875510
(Iteration 1821 / 39200) loss: 1.661172
(Iteration 1831 / 39200) loss: 2.130069
(Iteration 1841 / 39200) loss: 1.679051
(Iteration 1851 / 39200) loss: 1.498203
(Iteration 1861 / 39200) loss: 1.706867
(Iteration 1871 / 39200) loss: 1.772197
(Iteration 1881 / 39200) loss: 1.879655
(Iteration 1891 / 39200) loss: 1.571259
(Iteration 1901 / 39200) loss: 1.978036
(Iteration 1911 / 39200) loss: 1.717466
(Iteration 1921 / 39200) loss: 1.720818
(Iteration 1931 / 39200) loss: 1.955629
(Iteration 1941 / 39200) loss: 2.130352
(Iteration 1951 / 39200) loss: 1.794768
(Epoch 1 / 20) train acc: 0.379000; val_acc: 0.416000
(Iteration 1961 / 39200) loss: 1.746854
(Iteration 1971 / 39200) loss: 1.969212
(Iteration 1981 / 39200) loss: 2.152475
(Iteration 1991 / 39200) loss: 2.101151
(Iteration 2001 / 39200) loss: 1.733672
(Iteration 2011 / 39200) loss: 1.769659
(Iteration 2021 / 39200) loss: 1.798009
(Iteration 2031 / 39200) loss: 1.757904
(Iteration 2041 / 39200) loss: 1.849163
(Iteration 2051 / 39200) loss: 2.177884
(Iteration 2061 / 39200) loss: 1.961398
(Iteration 2071 / 39200) loss: 1.864764
(Iteration 2081 / 39200) loss: 1.554389
(Iteration 2091 / 39200) loss: 1.807582
(Iteration 2101 / 39200) loss: 1.811121
(Iteration 2111 / 39200) loss: 1.822951
(Iteration 2121 / 39200) loss: 1.694800
(Iteration 2131 / 39200) loss: 2.181775
(Iteration 2141 / 39200) loss: 1.884159
(Iteration 2151 / 39200) loss: 1.655433
(Iteration 2161 / 39200) loss: 1.596759

(Iteration 2171 / 39200) loss: 1.754040
(Iteration 2181 / 39200) loss: 1.769566
(Iteration 2191 / 39200) loss: 1.782154
(Iteration 2201 / 39200) loss: 1.821051
(Iteration 2211 / 39200) loss: 1.656369
(Iteration 2221 / 39200) loss: 1.845171
(Iteration 2231 / 39200) loss: 1.596701
(Iteration 2241 / 39200) loss: 1.793200
(Iteration 2251 / 39200) loss: 1.901322
(Iteration 2261 / 39200) loss: 2.088493
(Iteration 2271 / 39200) loss: 1.643642
(Iteration 2281 / 39200) loss: 1.702491
(Iteration 2291 / 39200) loss: 1.693007
(Iteration 2301 / 39200) loss: 1.524824
(Iteration 2311 / 39200) loss: 1.674742
(Iteration 2321 / 39200) loss: 1.934575
(Iteration 2331 / 39200) loss: 1.864551
(Iteration 2341 / 39200) loss: 1.731801
(Iteration 2351 / 39200) loss: 2.253600
(Iteration 2361 / 39200) loss: 2.005740
(Iteration 2371 / 39200) loss: 1.991589
(Iteration 2381 / 39200) loss: 1.740878
(Iteration 2391 / 39200) loss: 1.966752
(Iteration 2401 / 39200) loss: 1.758412
(Iteration 2411 / 39200) loss: 1.549352
(Iteration 2421 / 39200) loss: 1.634964
(Iteration 2431 / 39200) loss: 1.708960
(Iteration 2441 / 39200) loss: 1.649057
(Iteration 2451 / 39200) loss: 1.627893
(Iteration 2461 / 39200) loss: 1.693171
(Iteration 2471 / 39200) loss: 1.776465
(Iteration 2481 / 39200) loss: 1.592797
(Iteration 2491 / 39200) loss: 1.651105
(Iteration 2501 / 39200) loss: 1.775211
(Iteration 2511 / 39200) loss: 1.693261
(Iteration 2521 / 39200) loss: 1.849581
(Iteration 2531 / 39200) loss: 1.600874
(Iteration 2541 / 39200) loss: 1.653491
(Iteration 2551 / 39200) loss: 1.622342
(Iteration 2561 / 39200) loss: 1.556165
(Iteration 2571 / 39200) loss: 1.710065
(Iteration 2581 / 39200) loss: 2.008397
(Iteration 2591 / 39200) loss: 1.494379
(Iteration 2601 / 39200) loss: 1.689025
(Iteration 2611 / 39200) loss: 1.621431
(Iteration 2621 / 39200) loss: 1.750084
(Iteration 2631 / 39200) loss: 1.819671
(Iteration 2641 / 39200) loss: 1.603989

(Iteration 2651 / 39200) loss: 1.727191
(Iteration 2661 / 39200) loss: 2.135347
(Iteration 2671 / 39200) loss: 2.205488
(Iteration 2681 / 39200) loss: 1.693212
(Iteration 2691 / 39200) loss: 2.117985
(Iteration 2701 / 39200) loss: 1.849412
(Iteration 2711 / 39200) loss: 1.419887
(Iteration 2721 / 39200) loss: 1.874545
(Iteration 2731 / 39200) loss: 1.916249
(Iteration 2741 / 39200) loss: 1.774395
(Iteration 2751 / 39200) loss: 2.021962
(Iteration 2761 / 39200) loss: 1.797255
(Iteration 2771 / 39200) loss: 1.889316
(Iteration 2781 / 39200) loss: 1.832752
(Iteration 2791 / 39200) loss: 1.696471
(Iteration 2801 / 39200) loss: 1.614870
(Iteration 2811 / 39200) loss: 2.038165
(Iteration 2821 / 39200) loss: 2.006033
(Iteration 2831 / 39200) loss: 1.693869
(Iteration 2841 / 39200) loss: 1.700739
(Iteration 2851 / 39200) loss: 1.552532
(Iteration 2861 / 39200) loss: 1.758433
(Iteration 2871 / 39200) loss: 1.908137
(Iteration 2881 / 39200) loss: 1.550467
(Iteration 2891 / 39200) loss: 1.738892
(Iteration 2901 / 39200) loss: 1.847832
(Iteration 2911 / 39200) loss: 1.723861
(Iteration 2921 / 39200) loss: 1.918199
(Iteration 2931 / 39200) loss: 1.458919
(Iteration 2941 / 39200) loss: 1.779878
(Iteration 2951 / 39200) loss: 1.477868
(Iteration 2961 / 39200) loss: 1.576389
(Iteration 2971 / 39200) loss: 1.570321
(Iteration 2981 / 39200) loss: 1.831275
(Iteration 2991 / 39200) loss: 1.697827
(Iteration 3001 / 39200) loss: 1.586197
(Iteration 3011 / 39200) loss: 1.841208
(Iteration 3021 / 39200) loss: 1.917179
(Iteration 3031 / 39200) loss: 1.686247
(Iteration 3041 / 39200) loss: 1.623737
(Iteration 3051 / 39200) loss: 1.594569
(Iteration 3061 / 39200) loss: 1.762163
(Iteration 3071 / 39200) loss: 1.946923
(Iteration 3081 / 39200) loss: 1.860225
(Iteration 3091 / 39200) loss: 1.644382
(Iteration 3101 / 39200) loss: 1.716858
(Iteration 3111 / 39200) loss: 2.108994
(Iteration 3121 / 39200) loss: 1.634190

(Iteration 3131 / 39200) loss: 1.666594
(Iteration 3141 / 39200) loss: 1.713882
(Iteration 3151 / 39200) loss: 1.359702
(Iteration 3161 / 39200) loss: 2.000374
(Iteration 3171 / 39200) loss: 1.768168
(Iteration 3181 / 39200) loss: 1.966067
(Iteration 3191 / 39200) loss: 1.898376
(Iteration 3201 / 39200) loss: 1.902131
(Iteration 3211 / 39200) loss: 1.842470
(Iteration 3221 / 39200) loss: 1.806449
(Iteration 3231 / 39200) loss: 1.733430
(Iteration 3241 / 39200) loss: 1.829955
(Iteration 3251 / 39200) loss: 1.683682
(Iteration 3261 / 39200) loss: 1.989813
(Iteration 3271 / 39200) loss: 1.852714
(Iteration 3281 / 39200) loss: 1.759622
(Iteration 3291 / 39200) loss: 1.532214
(Iteration 3301 / 39200) loss: 1.812507
(Iteration 3311 / 39200) loss: 2.032373
(Iteration 3321 / 39200) loss: 1.659637
(Iteration 3331 / 39200) loss: 1.915039
(Iteration 3341 / 39200) loss: 1.761832
(Iteration 3351 / 39200) loss: 1.689066
(Iteration 3361 / 39200) loss: 1.925475
(Iteration 3371 / 39200) loss: 1.729478
(Iteration 3381 / 39200) loss: 1.826847
(Iteration 3391 / 39200) loss: 1.685297
(Iteration 3401 / 39200) loss: 1.609914
(Iteration 3411 / 39200) loss: 1.549685
(Iteration 3421 / 39200) loss: 1.767749
(Iteration 3431 / 39200) loss: 1.842609
(Iteration 3441 / 39200) loss: 1.662461
(Iteration 3451 / 39200) loss: 1.460567
(Iteration 3461 / 39200) loss: 1.896562
(Iteration 3471 / 39200) loss: 1.988663
(Iteration 3481 / 39200) loss: 1.488705
(Iteration 3491 / 39200) loss: 1.661979
(Iteration 3501 / 39200) loss: 1.904504
(Iteration 3511 / 39200) loss: 1.663872
(Iteration 3521 / 39200) loss: 2.023008
(Iteration 3531 / 39200) loss: 1.731338
(Iteration 3541 / 39200) loss: 1.643755
(Iteration 3551 / 39200) loss: 1.514500
(Iteration 3561 / 39200) loss: 1.762016
(Iteration 3571 / 39200) loss: 2.237239
(Iteration 3581 / 39200) loss: 1.419554
(Iteration 3591 / 39200) loss: 1.561184
(Iteration 3601 / 39200) loss: 1.616574

(Iteration 3611 / 39200) loss: 1.860484
(Iteration 3621 / 39200) loss: 2.094872
(Iteration 3631 / 39200) loss: 1.802189
(Iteration 3641 / 39200) loss: 1.606201
(Iteration 3651 / 39200) loss: 1.706260
(Iteration 3661 / 39200) loss: 1.934194
(Iteration 3671 / 39200) loss: 1.942570
(Iteration 3681 / 39200) loss: 1.713429
(Iteration 3691 / 39200) loss: 1.862687
(Iteration 3701 / 39200) loss: 1.627410
(Iteration 3711 / 39200) loss: 2.064437
(Iteration 3721 / 39200) loss: 1.979304
(Iteration 3731 / 39200) loss: 1.842282
(Iteration 3741 / 39200) loss: 1.956436
(Iteration 3751 / 39200) loss: 1.606818
(Iteration 3761 / 39200) loss: 1.748341
(Iteration 3771 / 39200) loss: 1.817639
(Iteration 3781 / 39200) loss: 1.399363
(Iteration 3791 / 39200) loss: 1.996377
(Iteration 3801 / 39200) loss: 1.486783
(Iteration 3811 / 39200) loss: 1.360591
(Iteration 3821 / 39200) loss: 1.489881
(Iteration 3831 / 39200) loss: 1.609036
(Iteration 3841 / 39200) loss: 1.512601
(Iteration 3851 / 39200) loss: 1.625218
(Iteration 3861 / 39200) loss: 1.897696
(Iteration 3871 / 39200) loss: 2.164587
(Iteration 3881 / 39200) loss: 1.920236
(Iteration 3891 / 39200) loss: 1.823032
(Iteration 3901 / 39200) loss: 1.696412
(Iteration 3911 / 39200) loss: 1.970563
(Epoch 2 / 20) train acc: 0.432000; val_acc: 0.443000
(Iteration 3921 / 39200) loss: 1.595321
(Iteration 3931 / 39200) loss: 2.112098
(Iteration 3941 / 39200) loss: 1.462200
(Iteration 3951 / 39200) loss: 1.975209
(Iteration 3961 / 39200) loss: 1.904154
(Iteration 3971 / 39200) loss: 1.994571
(Iteration 3981 / 39200) loss: 1.570009
(Iteration 3991 / 39200) loss: 1.347252
(Iteration 4001 / 39200) loss: 1.197513
(Iteration 4011 / 39200) loss: 1.746233
(Iteration 4021 / 39200) loss: 1.197957
(Iteration 4031 / 39200) loss: 1.558314
(Iteration 4041 / 39200) loss: 1.791444
(Iteration 4051 / 39200) loss: 1.960083
(Iteration 4061 / 39200) loss: 1.899397
(Iteration 4071 / 39200) loss: 1.619581

(Iteration 4081 / 39200) loss: 1.964104
(Iteration 4091 / 39200) loss: 1.804298
(Iteration 4101 / 39200) loss: 1.818104
(Iteration 4111 / 39200) loss: 1.757821
(Iteration 4121 / 39200) loss: 1.911975
(Iteration 4131 / 39200) loss: 1.836970
(Iteration 4141 / 39200) loss: 1.821051
(Iteration 4151 / 39200) loss: 1.614229
(Iteration 4161 / 39200) loss: 1.584167
(Iteration 4171 / 39200) loss: 1.359986
(Iteration 4181 / 39200) loss: 1.352210
(Iteration 4191 / 39200) loss: 1.963908
(Iteration 4201 / 39200) loss: 1.537014
(Iteration 4211 / 39200) loss: 1.501054
(Iteration 4221 / 39200) loss: 1.696673
(Iteration 4231 / 39200) loss: 2.154088
(Iteration 4241 / 39200) loss: 1.430911
(Iteration 4251 / 39200) loss: 1.808571
(Iteration 4261 / 39200) loss: 1.859296
(Iteration 4271 / 39200) loss: 1.591599
(Iteration 4281 / 39200) loss: 1.583388
(Iteration 4291 / 39200) loss: 1.777109
(Iteration 4301 / 39200) loss: 1.832111
(Iteration 4311 / 39200) loss: 1.538289
(Iteration 4321 / 39200) loss: 2.024485
(Iteration 4331 / 39200) loss: 1.914107
(Iteration 4341 / 39200) loss: 1.661801
(Iteration 4351 / 39200) loss: 1.674253
(Iteration 4361 / 39200) loss: 1.628801
(Iteration 4371 / 39200) loss: 1.925448
(Iteration 4381 / 39200) loss: 1.799167
(Iteration 4391 / 39200) loss: 1.780549
(Iteration 4401 / 39200) loss: 1.878803
(Iteration 4411 / 39200) loss: 1.749354
(Iteration 4421 / 39200) loss: 1.668603
(Iteration 4431 / 39200) loss: 1.559712
(Iteration 4441 / 39200) loss: 1.840593
(Iteration 4451 / 39200) loss: 2.036278
(Iteration 4461 / 39200) loss: 1.876353
(Iteration 4471 / 39200) loss: 2.043680
(Iteration 4481 / 39200) loss: 2.081886
(Iteration 4491 / 39200) loss: 1.719711
(Iteration 4501 / 39200) loss: 1.543954
(Iteration 4511 / 39200) loss: 1.552562
(Iteration 4521 / 39200) loss: 1.479005
(Iteration 4531 / 39200) loss: 1.685675
(Iteration 4541 / 39200) loss: 1.880607
(Iteration 4551 / 39200) loss: 1.759124

(Iteration 4561 / 39200) loss: 1.571191
(Iteration 4571 / 39200) loss: 1.774980
(Iteration 4581 / 39200) loss: 1.605489
(Iteration 4591 / 39200) loss: 1.615291
(Iteration 4601 / 39200) loss: 1.734711
(Iteration 4611 / 39200) loss: 1.660989
(Iteration 4621 / 39200) loss: 1.659748
(Iteration 4631 / 39200) loss: 1.950349
(Iteration 4641 / 39200) loss: 1.667657
(Iteration 4651 / 39200) loss: 1.663833
(Iteration 4661 / 39200) loss: 1.971418
(Iteration 4671 / 39200) loss: 1.511459
(Iteration 4681 / 39200) loss: 1.692894
(Iteration 4691 / 39200) loss: 1.666461
(Iteration 4701 / 39200) loss: 1.866939
(Iteration 4711 / 39200) loss: 1.814454
(Iteration 4721 / 39200) loss: 1.868205
(Iteration 4731 / 39200) loss: 1.481072
(Iteration 4741 / 39200) loss: 1.827236
(Iteration 4751 / 39200) loss: 1.416675
(Iteration 4761 / 39200) loss: 1.688719
(Iteration 4771 / 39200) loss: 1.540326
(Iteration 4781 / 39200) loss: 1.801927
(Iteration 4791 / 39200) loss: 1.737021
(Iteration 4801 / 39200) loss: 1.412084
(Iteration 4811 / 39200) loss: 1.653138
(Iteration 4821 / 39200) loss: 2.043015
(Iteration 4831 / 39200) loss: 1.491364
(Iteration 4841 / 39200) loss: 1.671928
(Iteration 4851 / 39200) loss: 1.718878
(Iteration 4861 / 39200) loss: 1.794730
(Iteration 4871 / 39200) loss: 1.809443
(Iteration 4881 / 39200) loss: 1.788416
(Iteration 4891 / 39200) loss: 1.524701
(Iteration 4901 / 39200) loss: 1.803947
(Iteration 4911 / 39200) loss: 1.635392
(Iteration 4921 / 39200) loss: 1.403926
(Iteration 4931 / 39200) loss: 1.995594
(Iteration 4941 / 39200) loss: 1.821179
(Iteration 4951 / 39200) loss: 1.765652
(Iteration 4961 / 39200) loss: 1.980786
(Iteration 4971 / 39200) loss: 1.680055
(Iteration 4981 / 39200) loss: 1.802886
(Iteration 4991 / 39200) loss: 1.766609
(Iteration 5001 / 39200) loss: 2.119667
(Iteration 5011 / 39200) loss: 1.601129
(Iteration 5021 / 39200) loss: 1.775945
(Iteration 5031 / 39200) loss: 1.889682

(Iteration 5041 / 39200) loss: 1.291764
(Iteration 5051 / 39200) loss: 1.853857
(Iteration 5061 / 39200) loss: 1.800444
(Iteration 5071 / 39200) loss: 1.795303
(Iteration 5081 / 39200) loss: 1.813965
(Iteration 5091 / 39200) loss: 1.657444
(Iteration 5101 / 39200) loss: 2.263003
(Iteration 5111 / 39200) loss: 1.508231
(Iteration 5121 / 39200) loss: 1.753808
(Iteration 5131 / 39200) loss: 1.891878
(Iteration 5141 / 39200) loss: 1.885837
(Iteration 5151 / 39200) loss: 1.788198
(Iteration 5161 / 39200) loss: 2.091331
(Iteration 5171 / 39200) loss: 1.735600
(Iteration 5181 / 39200) loss: 1.678950
(Iteration 5191 / 39200) loss: 1.929931
(Iteration 5201 / 39200) loss: 1.565188
(Iteration 5211 / 39200) loss: 1.630443
(Iteration 5221 / 39200) loss: 1.623076
(Iteration 5231 / 39200) loss: 1.774197
(Iteration 5241 / 39200) loss: 2.017349
(Iteration 5251 / 39200) loss: 1.633224
(Iteration 5261 / 39200) loss: 1.912602
(Iteration 5271 / 39200) loss: 1.454694
(Iteration 5281 / 39200) loss: 1.533341
(Iteration 5291 / 39200) loss: 1.647497
(Iteration 5301 / 39200) loss: 1.734177
(Iteration 5311 / 39200) loss: 1.922223
(Iteration 5321 / 39200) loss: 1.312485
(Iteration 5331 / 39200) loss: 1.595964
(Iteration 5341 / 39200) loss: 1.753501
(Iteration 5351 / 39200) loss: 1.696541
(Iteration 5361 / 39200) loss: 1.825727
(Iteration 5371 / 39200) loss: 1.565504
(Iteration 5381 / 39200) loss: 2.014568
(Iteration 5391 / 39200) loss: 1.620370
(Iteration 5401 / 39200) loss: 1.707508
(Iteration 5411 / 39200) loss: 2.024210
(Iteration 5421 / 39200) loss: 1.797527
(Iteration 5431 / 39200) loss: 1.996694
(Iteration 5441 / 39200) loss: 1.862654
(Iteration 5451 / 39200) loss: 1.790490
(Iteration 5461 / 39200) loss: 1.604370
(Iteration 5471 / 39200) loss: 1.840135
(Iteration 5481 / 39200) loss: 1.501112
(Iteration 5491 / 39200) loss: 1.523846
(Iteration 5501 / 39200) loss: 1.724997
(Iteration 5511 / 39200) loss: 1.761815

(Iteration 5521 / 39200) loss: 1.655163
(Iteration 5531 / 39200) loss: 1.678318
(Iteration 5541 / 39200) loss: 1.923934
(Iteration 5551 / 39200) loss: 1.826719
(Iteration 5561 / 39200) loss: 1.614647
(Iteration 5571 / 39200) loss: 1.672796
(Iteration 5581 / 39200) loss: 1.822332
(Iteration 5591 / 39200) loss: 1.891118
(Iteration 5601 / 39200) loss: 1.876264
(Iteration 5611 / 39200) loss: 1.585410
(Iteration 5621 / 39200) loss: 2.004550
(Iteration 5631 / 39200) loss: 1.847695
(Iteration 5641 / 39200) loss: 1.848167
(Iteration 5651 / 39200) loss: 1.458387
(Iteration 5661 / 39200) loss: 1.724198
(Iteration 5671 / 39200) loss: 1.715982
(Iteration 5681 / 39200) loss: 1.731864
(Iteration 5691 / 39200) loss: 1.744210
(Iteration 5701 / 39200) loss: 1.889773
(Iteration 5711 / 39200) loss: 1.620582
(Iteration 5721 / 39200) loss: 2.008225
(Iteration 5731 / 39200) loss: 1.716173
(Iteration 5741 / 39200) loss: 1.703477
(Iteration 5751 / 39200) loss: 1.670683
(Iteration 5761 / 39200) loss: 1.798282
(Iteration 5771 / 39200) loss: 1.440329
(Iteration 5781 / 39200) loss: 1.767882
(Iteration 5791 / 39200) loss: 1.683029
(Iteration 5801 / 39200) loss: 1.676076
(Iteration 5811 / 39200) loss: 1.832878
(Iteration 5821 / 39200) loss: 1.487141
(Iteration 5831 / 39200) loss: 2.037383
(Iteration 5841 / 39200) loss: 1.664656
(Iteration 5851 / 39200) loss: 1.861401
(Iteration 5861 / 39200) loss: 1.207743
(Iteration 5871 / 39200) loss: 1.730303
(Epoch 3 / 20) train acc: 0.477000; val_acc: 0.463000
(Iteration 5881 / 39200) loss: 1.857237
(Iteration 5891 / 39200) loss: 2.088953
(Iteration 5901 / 39200) loss: 1.662834
(Iteration 5911 / 39200) loss: 2.091253
(Iteration 5921 / 39200) loss: 1.976961
(Iteration 5931 / 39200) loss: 1.521523
(Iteration 5941 / 39200) loss: 1.637845
(Iteration 5951 / 39200) loss: 2.101026
(Iteration 5961 / 39200) loss: 1.332115
(Iteration 5971 / 39200) loss: 1.821405
(Iteration 5981 / 39200) loss: 1.533848

(Iteration 5991 / 39200) loss: 1.500452
(Iteration 6001 / 39200) loss: 2.156840
(Iteration 6011 / 39200) loss: 1.842925
(Iteration 6021 / 39200) loss: 2.077616
(Iteration 6031 / 39200) loss: 1.813980
(Iteration 6041 / 39200) loss: 1.819634
(Iteration 6051 / 39200) loss: 1.539370
(Iteration 6061 / 39200) loss: 1.998718
(Iteration 6071 / 39200) loss: 2.039273
(Iteration 6081 / 39200) loss: 1.874041
(Iteration 6091 / 39200) loss: 1.656431
(Iteration 6101 / 39200) loss: 1.700865
(Iteration 6111 / 39200) loss: 1.488687
(Iteration 6121 / 39200) loss: 1.595909
(Iteration 6131 / 39200) loss: 1.790814
(Iteration 6141 / 39200) loss: 1.707116
(Iteration 6151 / 39200) loss: 2.255435
(Iteration 6161 / 39200) loss: 1.392185
(Iteration 6171 / 39200) loss: 1.427461
(Iteration 6181 / 39200) loss: 1.784008
(Iteration 6191 / 39200) loss: 1.633824
(Iteration 6201 / 39200) loss: 1.701548
(Iteration 6211 / 39200) loss: 1.671459
(Iteration 6221 / 39200) loss: 1.673821
(Iteration 6231 / 39200) loss: 1.732620
(Iteration 6241 / 39200) loss: 1.767985
(Iteration 6251 / 39200) loss: 1.655023
(Iteration 6261 / 39200) loss: 1.802986
(Iteration 6271 / 39200) loss: 1.428025
(Iteration 6281 / 39200) loss: 2.015703
(Iteration 6291 / 39200) loss: 2.044223
(Iteration 6301 / 39200) loss: 1.173905
(Iteration 6311 / 39200) loss: 1.481580
(Iteration 6321 / 39200) loss: 1.540388
(Iteration 6331 / 39200) loss: 1.744547
(Iteration 6341 / 39200) loss: 1.617861
(Iteration 6351 / 39200) loss: 1.699195
(Iteration 6361 / 39200) loss: 1.768542
(Iteration 6371 / 39200) loss: 1.736244
(Iteration 6381 / 39200) loss: 1.697209
(Iteration 6391 / 39200) loss: 1.734730
(Iteration 6401 / 39200) loss: 1.627478
(Iteration 6411 / 39200) loss: 1.635304
(Iteration 6421 / 39200) loss: 1.537948
(Iteration 6431 / 39200) loss: 1.438772
(Iteration 6441 / 39200) loss: 1.868237
(Iteration 6451 / 39200) loss: 2.233099
(Iteration 6461 / 39200) loss: 1.747152

(Iteration 6471 / 39200) loss: 1.720535
(Iteration 6481 / 39200) loss: 1.855634
(Iteration 6491 / 39200) loss: 1.764890
(Iteration 6501 / 39200) loss: 1.831401
(Iteration 6511 / 39200) loss: 1.625651
(Iteration 6521 / 39200) loss: 1.725347
(Iteration 6531 / 39200) loss: 1.814501
(Iteration 6541 / 39200) loss: 1.661790
(Iteration 6551 / 39200) loss: 1.532124
(Iteration 6561 / 39200) loss: 1.553863
(Iteration 6571 / 39200) loss: 1.703089
(Iteration 6581 / 39200) loss: 1.647682
(Iteration 6591 / 39200) loss: 1.615421
(Iteration 6601 / 39200) loss: 1.540969
(Iteration 6611 / 39200) loss: 1.676659
(Iteration 6621 / 39200) loss: 1.372078
(Iteration 6631 / 39200) loss: 1.367587
(Iteration 6641 / 39200) loss: 1.857574
(Iteration 6651 / 39200) loss: 1.687730
(Iteration 6661 / 39200) loss: 1.640030
(Iteration 6671 / 39200) loss: 1.588143
(Iteration 6681 / 39200) loss: 1.778758
(Iteration 6691 / 39200) loss: 1.395865
(Iteration 6701 / 39200) loss: 1.675175
(Iteration 6711 / 39200) loss: 2.051380
(Iteration 6721 / 39200) loss: 1.295704
(Iteration 6731 / 39200) loss: 1.501946
(Iteration 6741 / 39200) loss: 1.882881
(Iteration 6751 / 39200) loss: 1.417844
(Iteration 6761 / 39200) loss: 1.504022
(Iteration 6771 / 39200) loss: 1.716528
(Iteration 6781 / 39200) loss: 1.783953
(Iteration 6791 / 39200) loss: 1.571013
(Iteration 6801 / 39200) loss: 1.514065
(Iteration 6811 / 39200) loss: 1.589624
(Iteration 6821 / 39200) loss: 1.648640
(Iteration 6831 / 39200) loss: 1.626128
(Iteration 6841 / 39200) loss: 1.538344
(Iteration 6851 / 39200) loss: 1.943011
(Iteration 6861 / 39200) loss: 1.731529
(Iteration 6871 / 39200) loss: 1.636417
(Iteration 6881 / 39200) loss: 2.082206
(Iteration 6891 / 39200) loss: 1.668546
(Iteration 6901 / 39200) loss: 1.700188
(Iteration 6911 / 39200) loss: 1.497786
(Iteration 6921 / 39200) loss: 1.663698
(Iteration 6931 / 39200) loss: 1.631068
(Iteration 6941 / 39200) loss: 1.741737

(Iteration 6951 / 39200) loss: 1.635378
(Iteration 6961 / 39200) loss: 2.062575
(Iteration 6971 / 39200) loss: 1.826679
(Iteration 6981 / 39200) loss: 1.657983
(Iteration 6991 / 39200) loss: 1.664651
(Iteration 7001 / 39200) loss: 1.777015
(Iteration 7011 / 39200) loss: 1.435114
(Iteration 7021 / 39200) loss: 1.675456
(Iteration 7031 / 39200) loss: 1.781036
(Iteration 7041 / 39200) loss: 1.906368
(Iteration 7051 / 39200) loss: 1.317939
(Iteration 7061 / 39200) loss: 1.828804
(Iteration 7071 / 39200) loss: 1.692868
(Iteration 7081 / 39200) loss: 1.681318
(Iteration 7091 / 39200) loss: 1.560652
(Iteration 7101 / 39200) loss: 1.624005
(Iteration 7111 / 39200) loss: 1.623068
(Iteration 7121 / 39200) loss: 1.898455
(Iteration 7131 / 39200) loss: 1.306563
(Iteration 7141 / 39200) loss: 1.260912
(Iteration 7151 / 39200) loss: 1.624753
(Iteration 7161 / 39200) loss: 1.453661
(Iteration 7171 / 39200) loss: 1.537710
(Iteration 7181 / 39200) loss: 1.566140
(Iteration 7191 / 39200) loss: 1.503192
(Iteration 7201 / 39200) loss: 1.626511
(Iteration 7211 / 39200) loss: 1.823610
(Iteration 7221 / 39200) loss: 1.548729
(Iteration 7231 / 39200) loss: 1.932905
(Iteration 7241 / 39200) loss: 1.724802
(Iteration 7251 / 39200) loss: 1.738879
(Iteration 7261 / 39200) loss: 1.880360
(Iteration 7271 / 39200) loss: 1.790313
(Iteration 7281 / 39200) loss: 1.800684
(Iteration 7291 / 39200) loss: 1.328311
(Iteration 7301 / 39200) loss: 1.487863
(Iteration 7311 / 39200) loss: 1.588098
(Iteration 7321 / 39200) loss: 1.730467
(Iteration 7331 / 39200) loss: 1.574282
(Iteration 7341 / 39200) loss: 1.622465
(Iteration 7351 / 39200) loss: 1.899556
(Iteration 7361 / 39200) loss: 1.828170
(Iteration 7371 / 39200) loss: 1.703479
(Iteration 7381 / 39200) loss: 1.718206
(Iteration 7391 / 39200) loss: 1.538493
(Iteration 7401 / 39200) loss: 2.230060
(Iteration 7411 / 39200) loss: 1.519118
(Iteration 7421 / 39200) loss: 1.549434

(Iteration 7431 / 39200) loss: 1.540292
(Iteration 7441 / 39200) loss: 1.968076
(Iteration 7451 / 39200) loss: 1.526822
(Iteration 7461 / 39200) loss: 1.638367
(Iteration 7471 / 39200) loss: 1.560075
(Iteration 7481 / 39200) loss: 1.739682
(Iteration 7491 / 39200) loss: 1.751053
(Iteration 7501 / 39200) loss: 1.473221
(Iteration 7511 / 39200) loss: 2.067597
(Iteration 7521 / 39200) loss: 1.713887
(Iteration 7531 / 39200) loss: 1.859483
(Iteration 7541 / 39200) loss: 1.560787
(Iteration 7551 / 39200) loss: 1.503037
(Iteration 7561 / 39200) loss: 1.581905
(Iteration 7571 / 39200) loss: 1.800407
(Iteration 7581 / 39200) loss: 1.733301
(Iteration 7591 / 39200) loss: 2.042111
(Iteration 7601 / 39200) loss: 1.276870
(Iteration 7611 / 39200) loss: 1.508971
(Iteration 7621 / 39200) loss: 1.816977
(Iteration 7631 / 39200) loss: 1.519897
(Iteration 7641 / 39200) loss: 1.404633
(Iteration 7651 / 39200) loss: 1.772173
(Iteration 7661 / 39200) loss: 1.706074
(Iteration 7671 / 39200) loss: 1.542434
(Iteration 7681 / 39200) loss: 2.051346
(Iteration 7691 / 39200) loss: 1.283696
(Iteration 7701 / 39200) loss: 1.739123
(Iteration 7711 / 39200) loss: 1.833348
(Iteration 7721 / 39200) loss: 1.537882
(Iteration 7731 / 39200) loss: 1.563024
(Iteration 7741 / 39200) loss: 1.635728
(Iteration 7751 / 39200) loss: 1.687251
(Iteration 7761 / 39200) loss: 1.817379
(Iteration 7771 / 39200) loss: 1.556292
(Iteration 7781 / 39200) loss: 1.878299
(Iteration 7791 / 39200) loss: 2.195210
(Iteration 7801 / 39200) loss: 1.440324
(Iteration 7811 / 39200) loss: 2.123186
(Iteration 7821 / 39200) loss: 2.009198
(Iteration 7831 / 39200) loss: 2.007013
(Epoch 4 / 20) train acc: 0.516000; val_acc: 0.488000
(Iteration 7841 / 39200) loss: 1.760856
(Iteration 7851 / 39200) loss: 1.817385
(Iteration 7861 / 39200) loss: 1.772466
(Iteration 7871 / 39200) loss: 1.883135
(Iteration 7881 / 39200) loss: 1.378930
(Iteration 7891 / 39200) loss: 1.774846

(Iteration 7901 / 39200) loss: 1.769223
(Iteration 7911 / 39200) loss: 1.734680
(Iteration 7921 / 39200) loss: 1.847188
(Iteration 7931 / 39200) loss: 1.655003
(Iteration 7941 / 39200) loss: 1.663468
(Iteration 7951 / 39200) loss: 1.506675
(Iteration 7961 / 39200) loss: 1.670302
(Iteration 7971 / 39200) loss: 1.644248
(Iteration 7981 / 39200) loss: 1.892125
(Iteration 7991 / 39200) loss: 1.652155
(Iteration 8001 / 39200) loss: 1.854700
(Iteration 8011 / 39200) loss: 1.719705
(Iteration 8021 / 39200) loss: 1.536374
(Iteration 8031 / 39200) loss: 1.891396
(Iteration 8041 / 39200) loss: 1.910721
(Iteration 8051 / 39200) loss: 1.372806
(Iteration 8061 / 39200) loss: 1.182192
(Iteration 8071 / 39200) loss: 1.499579
(Iteration 8081 / 39200) loss: 1.277263
(Iteration 8091 / 39200) loss: 1.594770
(Iteration 8101 / 39200) loss: 1.943514
(Iteration 8111 / 39200) loss: 1.821150
(Iteration 8121 / 39200) loss: 1.603630
(Iteration 8131 / 39200) loss: 1.552965
(Iteration 8141 / 39200) loss: 1.636753
(Iteration 8151 / 39200) loss: 1.827061
(Iteration 8161 / 39200) loss: 1.523282
(Iteration 8171 / 39200) loss: 1.881449
(Iteration 8181 / 39200) loss: 1.730706
(Iteration 8191 / 39200) loss: 1.814096
(Iteration 8201 / 39200) loss: 1.186597
(Iteration 8211 / 39200) loss: 2.000311
(Iteration 8221 / 39200) loss: 1.685852
(Iteration 8231 / 39200) loss: 2.119424
(Iteration 8241 / 39200) loss: 1.858656
(Iteration 8251 / 39200) loss: 1.910005
(Iteration 8261 / 39200) loss: 1.768730
(Iteration 8271 / 39200) loss: 1.592876
(Iteration 8281 / 39200) loss: 1.634989
(Iteration 8291 / 39200) loss: 1.550677
(Iteration 8301 / 39200) loss: 1.542021
(Iteration 8311 / 39200) loss: 1.958296
(Iteration 8321 / 39200) loss: 1.788286
(Iteration 8331 / 39200) loss: 2.192223
(Iteration 8341 / 39200) loss: 1.775057
(Iteration 8351 / 39200) loss: 2.000203
(Iteration 8361 / 39200) loss: 1.470177
(Iteration 8371 / 39200) loss: 1.934313

(Iteration 8381 / 39200) loss: 1.608581
(Iteration 8391 / 39200) loss: 1.810137
(Iteration 8401 / 39200) loss: 1.668903
(Iteration 8411 / 39200) loss: 2.003023
(Iteration 8421 / 39200) loss: 1.587015
(Iteration 8431 / 39200) loss: 1.559164
(Iteration 8441 / 39200) loss: 1.754961
(Iteration 8451 / 39200) loss: 1.884061
(Iteration 8461 / 39200) loss: 2.052478
(Iteration 8471 / 39200) loss: 1.821937
(Iteration 8481 / 39200) loss: 1.623264
(Iteration 8491 / 39200) loss: 1.457130
(Iteration 8501 / 39200) loss: 1.689142
(Iteration 8511 / 39200) loss: 1.511180
(Iteration 8521 / 39200) loss: 1.612369
(Iteration 8531 / 39200) loss: 1.756674
(Iteration 8541 / 39200) loss: 1.393683
(Iteration 8551 / 39200) loss: 1.783718
(Iteration 8561 / 39200) loss: 1.364011
(Iteration 8571 / 39200) loss: 1.756699
(Iteration 8581 / 39200) loss: 1.208794
(Iteration 8591 / 39200) loss: 1.595813
(Iteration 8601 / 39200) loss: 1.757959
(Iteration 8611 / 39200) loss: 1.793331
(Iteration 8621 / 39200) loss: 1.736235
(Iteration 8631 / 39200) loss: 1.560878
(Iteration 8641 / 39200) loss: 1.609898
(Iteration 8651 / 39200) loss: 1.860445
(Iteration 8661 / 39200) loss: 1.564806
(Iteration 8671 / 39200) loss: 1.538872
(Iteration 8681 / 39200) loss: 1.574993
(Iteration 8691 / 39200) loss: 1.636717
(Iteration 8701 / 39200) loss: 1.571569
(Iteration 8711 / 39200) loss: 1.583807
(Iteration 8721 / 39200) loss: 1.382735
(Iteration 8731 / 39200) loss: 1.933459
(Iteration 8741 / 39200) loss: 1.488002
(Iteration 8751 / 39200) loss: 1.808528
(Iteration 8761 / 39200) loss: 1.793192
(Iteration 8771 / 39200) loss: 1.736849
(Iteration 8781 / 39200) loss: 1.735137
(Iteration 8791 / 39200) loss: 1.736230
(Iteration 8801 / 39200) loss: 1.687962
(Iteration 8811 / 39200) loss: 1.729842
(Iteration 8821 / 39200) loss: 1.638633
(Iteration 8831 / 39200) loss: 1.849539
(Iteration 8841 / 39200) loss: 1.810648
(Iteration 8851 / 39200) loss: 1.612319

(Iteration 8861 / 39200) loss: 1.356279
(Iteration 8871 / 39200) loss: 1.708561
(Iteration 8881 / 39200) loss: 1.637691
(Iteration 8891 / 39200) loss: 1.565932
(Iteration 8901 / 39200) loss: 1.583187
(Iteration 8911 / 39200) loss: 1.565264
(Iteration 8921 / 39200) loss: 1.659378
(Iteration 8931 / 39200) loss: 1.458633
(Iteration 8941 / 39200) loss: 1.592036
(Iteration 8951 / 39200) loss: 1.385693
(Iteration 8961 / 39200) loss: 1.404033
(Iteration 8971 / 39200) loss: 1.835064
(Iteration 8981 / 39200) loss: 1.728044
(Iteration 8991 / 39200) loss: 1.779606
(Iteration 9001 / 39200) loss: 1.505568
(Iteration 9011 / 39200) loss: 1.463723
(Iteration 9021 / 39200) loss: 1.472168
(Iteration 9031 / 39200) loss: 1.689421
(Iteration 9041 / 39200) loss: 1.836313
(Iteration 9051 / 39200) loss: 1.578455
(Iteration 9061 / 39200) loss: 1.547479
(Iteration 9071 / 39200) loss: 1.813644
(Iteration 9081 / 39200) loss: 1.855458
(Iteration 9091 / 39200) loss: 1.872103
(Iteration 9101 / 39200) loss: 1.827670
(Iteration 9111 / 39200) loss: 1.360639
(Iteration 9121 / 39200) loss: 1.589711
(Iteration 9131 / 39200) loss: 1.416607
(Iteration 9141 / 39200) loss: 1.879158
(Iteration 9151 / 39200) loss: 1.579849
(Iteration 9161 / 39200) loss: 1.490712
(Iteration 9171 / 39200) loss: 1.981008
(Iteration 9181 / 39200) loss: 1.630072
(Iteration 9191 / 39200) loss: 1.719325
(Iteration 9201 / 39200) loss: 1.509168
(Iteration 9211 / 39200) loss: 1.775000
(Iteration 9221 / 39200) loss: 2.049556
(Iteration 9231 / 39200) loss: 1.761724
(Iteration 9241 / 39200) loss: 1.942415
(Iteration 9251 / 39200) loss: 1.330922
(Iteration 9261 / 39200) loss: 1.805869
(Iteration 9271 / 39200) loss: 1.578241
(Iteration 9281 / 39200) loss: 1.197558
(Iteration 9291 / 39200) loss: 1.406123
(Iteration 9301 / 39200) loss: 1.635811
(Iteration 9311 / 39200) loss: 1.602601
(Iteration 9321 / 39200) loss: 1.601544
(Iteration 9331 / 39200) loss: 1.411507

(Iteration 9341 / 39200) loss: 1.786330
(Iteration 9351 / 39200) loss: 1.428743
(Iteration 9361 / 39200) loss: 1.934739
(Iteration 9371 / 39200) loss: 2.037704
(Iteration 9381 / 39200) loss: 1.262028
(Iteration 9391 / 39200) loss: 1.493827
(Iteration 9401 / 39200) loss: 1.681440
(Iteration 9411 / 39200) loss: 1.618453
(Iteration 9421 / 39200) loss: 2.136620
(Iteration 9431 / 39200) loss: 1.339525
(Iteration 9441 / 39200) loss: 1.347698
(Iteration 9451 / 39200) loss: 1.258816
(Iteration 9461 / 39200) loss: 1.362542
(Iteration 9471 / 39200) loss: 1.458872
(Iteration 9481 / 39200) loss: 1.668569
(Iteration 9491 / 39200) loss: 1.452291
(Iteration 9501 / 39200) loss: 1.797083
(Iteration 9511 / 39200) loss: 1.940519
(Iteration 9521 / 39200) loss: 1.564759
(Iteration 9531 / 39200) loss: 1.587894
(Iteration 9541 / 39200) loss: 1.669337
(Iteration 9551 / 39200) loss: 1.630057
(Iteration 9561 / 39200) loss: 1.773317
(Iteration 9571 / 39200) loss: 1.377098
(Iteration 9581 / 39200) loss: 1.690831
(Iteration 9591 / 39200) loss: 1.554654
(Iteration 9601 / 39200) loss: 1.576396
(Iteration 9611 / 39200) loss: 1.781341
(Iteration 9621 / 39200) loss: 1.639418
(Iteration 9631 / 39200) loss: 1.513548
(Iteration 9641 / 39200) loss: 1.689647
(Iteration 9651 / 39200) loss: 1.722344
(Iteration 9661 / 39200) loss: 1.698404
(Iteration 9671 / 39200) loss: 1.732178
(Iteration 9681 / 39200) loss: 1.857417
(Iteration 9691 / 39200) loss: 1.809153
(Iteration 9701 / 39200) loss: 1.595765
(Iteration 9711 / 39200) loss: 1.426872
(Iteration 9721 / 39200) loss: 1.692501
(Iteration 9731 / 39200) loss: 2.077093
(Iteration 9741 / 39200) loss: 1.525089
(Iteration 9751 / 39200) loss: 1.733442
(Iteration 9761 / 39200) loss: 1.422401
(Iteration 9771 / 39200) loss: 1.395157
(Iteration 9781 / 39200) loss: 1.327509
(Iteration 9791 / 39200) loss: 1.212991
(Epoch 5 / 20) train acc: 0.509000; val_acc: 0.489000
(Iteration 9801 / 39200) loss: 1.600999

(Iteration 9811 / 39200) loss: 1.788123
(Iteration 9821 / 39200) loss: 1.738724
(Iteration 9831 / 39200) loss: 1.315715
(Iteration 9841 / 39200) loss: 1.548587
(Iteration 9851 / 39200) loss: 1.788999
(Iteration 9861 / 39200) loss: 1.515471
(Iteration 9871 / 39200) loss: 1.246388
(Iteration 9881 / 39200) loss: 1.476631
(Iteration 9891 / 39200) loss: 2.193028
(Iteration 9901 / 39200) loss: 1.457079
(Iteration 9911 / 39200) loss: 1.696867
(Iteration 9921 / 39200) loss: 1.573996
(Iteration 9931 / 39200) loss: 1.617384
(Iteration 9941 / 39200) loss: 1.376476
(Iteration 9951 / 39200) loss: 1.878198
(Iteration 9961 / 39200) loss: 1.473506
(Iteration 9971 / 39200) loss: 1.338391
(Iteration 9981 / 39200) loss: 1.877372
(Iteration 9991 / 39200) loss: 1.633374
(Iteration 10001 / 39200) loss: 1.518474
(Iteration 10011 / 39200) loss: 1.686315
(Iteration 10021 / 39200) loss: 1.541757
(Iteration 10031 / 39200) loss: 1.626779
(Iteration 10041 / 39200) loss: 1.895073
(Iteration 10051 / 39200) loss: 1.813971
(Iteration 10061 / 39200) loss: 2.138323
(Iteration 10071 / 39200) loss: 1.743915
(Iteration 10081 / 39200) loss: 1.680230
(Iteration 10091 / 39200) loss: 1.415992
(Iteration 10101 / 39200) loss: 1.674438
(Iteration 10111 / 39200) loss: 1.753648
(Iteration 10121 / 39200) loss: 1.482688
(Iteration 10131 / 39200) loss: 1.358934
(Iteration 10141 / 39200) loss: 1.627083
(Iteration 10151 / 39200) loss: 1.723659
(Iteration 10161 / 39200) loss: 1.841201
(Iteration 10171 / 39200) loss: 1.451541
(Iteration 10181 / 39200) loss: 1.468158
(Iteration 10191 / 39200) loss: 1.518254
(Iteration 10201 / 39200) loss: 1.543351
(Iteration 10211 / 39200) loss: 1.896665
(Iteration 10221 / 39200) loss: 1.791126
(Iteration 10231 / 39200) loss: 1.586457
(Iteration 10241 / 39200) loss: 1.352234
(Iteration 10251 / 39200) loss: 1.839286
(Iteration 10261 / 39200) loss: 1.885385
(Iteration 10271 / 39200) loss: 1.172972
(Iteration 10281 / 39200) loss: 1.830700

(Iteration 10291 / 39200) loss: 1.519391
(Iteration 10301 / 39200) loss: 1.501085
(Iteration 10311 / 39200) loss: 1.604337
(Iteration 10321 / 39200) loss: 1.377992
(Iteration 10331 / 39200) loss: 1.608108
(Iteration 10341 / 39200) loss: 1.923414
(Iteration 10351 / 39200) loss: 1.547467
(Iteration 10361 / 39200) loss: 1.953178
(Iteration 10371 / 39200) loss: 1.605155
(Iteration 10381 / 39200) loss: 1.520413
(Iteration 10391 / 39200) loss: 1.244723
(Iteration 10401 / 39200) loss: 1.584973
(Iteration 10411 / 39200) loss: 1.643347
(Iteration 10421 / 39200) loss: 1.731136
(Iteration 10431 / 39200) loss: 1.517752
(Iteration 10441 / 39200) loss: 1.224969
(Iteration 10451 / 39200) loss: 1.632341
(Iteration 10461 / 39200) loss: 1.599823
(Iteration 10471 / 39200) loss: 1.496511
(Iteration 10481 / 39200) loss: 2.025079
(Iteration 10491 / 39200) loss: 1.636454
(Iteration 10501 / 39200) loss: 1.414405
(Iteration 10511 / 39200) loss: 1.703855
(Iteration 10521 / 39200) loss: 1.834823
(Iteration 10531 / 39200) loss: 1.828133
(Iteration 10541 / 39200) loss: 1.634898
(Iteration 10551 / 39200) loss: 1.600695
(Iteration 10561 / 39200) loss: 1.639903
(Iteration 10571 / 39200) loss: 1.678928
(Iteration 10581 / 39200) loss: 1.605142
(Iteration 10591 / 39200) loss: 1.280872
(Iteration 10601 / 39200) loss: 1.554378
(Iteration 10611 / 39200) loss: 1.502844
(Iteration 10621 / 39200) loss: 1.467198
(Iteration 10631 / 39200) loss: 1.616484
(Iteration 10641 / 39200) loss: 1.794787
(Iteration 10651 / 39200) loss: 1.811121
(Iteration 10661 / 39200) loss: 1.441462
(Iteration 10671 / 39200) loss: 1.717097
(Iteration 10681 / 39200) loss: 2.121883
(Iteration 10691 / 39200) loss: 1.408459
(Iteration 10701 / 39200) loss: 1.697014
(Iteration 10711 / 39200) loss: 1.638478
(Iteration 10721 / 39200) loss: 1.721959
(Iteration 10731 / 39200) loss: 1.835902
(Iteration 10741 / 39200) loss: 2.010557
(Iteration 10751 / 39200) loss: 1.802177
(Iteration 10761 / 39200) loss: 1.521842

(Iteration 10771 / 39200) loss: 1.924563
(Iteration 10781 / 39200) loss: 1.717719
(Iteration 10791 / 39200) loss: 1.908119
(Iteration 10801 / 39200) loss: 1.478323
(Iteration 10811 / 39200) loss: 1.321253
(Iteration 10821 / 39200) loss: 1.830058
(Iteration 10831 / 39200) loss: 1.749587
(Iteration 10841 / 39200) loss: 1.674926
(Iteration 10851 / 39200) loss: 1.670143
(Iteration 10861 / 39200) loss: 1.606267
(Iteration 10871 / 39200) loss: 1.267185
(Iteration 10881 / 39200) loss: 1.917053
(Iteration 10891 / 39200) loss: 1.344125
(Iteration 10901 / 39200) loss: 1.580354
(Iteration 10911 / 39200) loss: 1.613581
(Iteration 10921 / 39200) loss: 1.765900
(Iteration 10931 / 39200) loss: 1.848303
(Iteration 10941 / 39200) loss: 1.572541
(Iteration 10951 / 39200) loss: 1.535301
(Iteration 10961 / 39200) loss: 1.553884
(Iteration 10971 / 39200) loss: 1.432020
(Iteration 10981 / 39200) loss: 1.920352
(Iteration 10991 / 39200) loss: 1.887556
(Iteration 11001 / 39200) loss: 1.687717
(Iteration 11011 / 39200) loss: 1.177729
(Iteration 11021 / 39200) loss: 1.864361
(Iteration 11031 / 39200) loss: 2.065240
(Iteration 11041 / 39200) loss: 1.360166
(Iteration 11051 / 39200) loss: 1.643682
(Iteration 11061 / 39200) loss: 1.339128
(Iteration 11071 / 39200) loss: 1.388990
(Iteration 11081 / 39200) loss: 1.633058
(Iteration 11091 / 39200) loss: 1.512961
(Iteration 11101 / 39200) loss: 1.616855
(Iteration 11111 / 39200) loss: 1.333561
(Iteration 11121 / 39200) loss: 1.518750
(Iteration 11131 / 39200) loss: 1.743677
(Iteration 11141 / 39200) loss: 1.734183
(Iteration 11151 / 39200) loss: 1.325207
(Iteration 11161 / 39200) loss: 1.442173
(Iteration 11171 / 39200) loss: 2.143736
(Iteration 11181 / 39200) loss: 1.763267
(Iteration 11191 / 39200) loss: 1.426760
(Iteration 11201 / 39200) loss: 1.561750
(Iteration 11211 / 39200) loss: 1.627924
(Iteration 11221 / 39200) loss: 1.853641
(Iteration 11231 / 39200) loss: 1.873774
(Iteration 11241 / 39200) loss: 1.710628

(Iteration 11251 / 39200) loss: 1.655113
(Iteration 11261 / 39200) loss: 1.491514
(Iteration 11271 / 39200) loss: 1.449044
(Iteration 11281 / 39200) loss: 1.406933
(Iteration 11291 / 39200) loss: 1.597353
(Iteration 11301 / 39200) loss: 1.503595
(Iteration 11311 / 39200) loss: 1.535806
(Iteration 11321 / 39200) loss: 1.490185
(Iteration 11331 / 39200) loss: 1.533834
(Iteration 11341 / 39200) loss: 1.276255
(Iteration 11351 / 39200) loss: 1.306332
(Iteration 11361 / 39200) loss: 1.625329
(Iteration 11371 / 39200) loss: 1.472677
(Iteration 11381 / 39200) loss: 1.265109
(Iteration 11391 / 39200) loss: 1.418258
(Iteration 11401 / 39200) loss: 1.436404
(Iteration 11411 / 39200) loss: 1.419311
(Iteration 11421 / 39200) loss: 1.670927
(Iteration 11431 / 39200) loss: 1.635977
(Iteration 11441 / 39200) loss: 1.282371
(Iteration 11451 / 39200) loss: 1.656822
(Iteration 11461 / 39200) loss: 2.041522
(Iteration 11471 / 39200) loss: 1.701603
(Iteration 11481 / 39200) loss: 1.669931
(Iteration 11491 / 39200) loss: 1.469710
(Iteration 11501 / 39200) loss: 1.428346
(Iteration 11511 / 39200) loss: 1.472562
(Iteration 11521 / 39200) loss: 1.698786
(Iteration 11531 / 39200) loss: 1.708869
(Iteration 11541 / 39200) loss: 1.809119
(Iteration 11551 / 39200) loss: 1.639943
(Iteration 11561 / 39200) loss: 1.718110
(Iteration 11571 / 39200) loss: 1.859382
(Iteration 11581 / 39200) loss: 2.096197
(Iteration 11591 / 39200) loss: 1.151726
(Iteration 11601 / 39200) loss: 1.384317
(Iteration 11611 / 39200) loss: 1.690194
(Iteration 11621 / 39200) loss: 1.785738
(Iteration 11631 / 39200) loss: 1.384152
(Iteration 11641 / 39200) loss: 1.612067
(Iteration 11651 / 39200) loss: 1.791270
(Iteration 11661 / 39200) loss: 2.148379
(Iteration 11671 / 39200) loss: 1.555696
(Iteration 11681 / 39200) loss: 2.042834
(Iteration 11691 / 39200) loss: 1.290470
(Iteration 11701 / 39200) loss: 1.526727
(Iteration 11711 / 39200) loss: 1.794025
(Iteration 11721 / 39200) loss: 1.395615

(Iteration 11731 / 39200) loss: 1.391747
(Iteration 11741 / 39200) loss: 1.648471
(Iteration 11751 / 39200) loss: 1.216619
(Epoch 6 / 20) train acc: 0.524000; val_acc: 0.502000
(Iteration 11761 / 39200) loss: 1.392666
(Iteration 11771 / 39200) loss: 1.816419
(Iteration 11781 / 39200) loss: 1.536564
(Iteration 11791 / 39200) loss: 2.183370
(Iteration 11801 / 39200) loss: 1.415098
(Iteration 11811 / 39200) loss: 1.413838
(Iteration 11821 / 39200) loss: 1.649093
(Iteration 11831 / 39200) loss: 1.353139
(Iteration 11841 / 39200) loss: 1.700615
(Iteration 11851 / 39200) loss: 2.151483
(Iteration 11861 / 39200) loss: 1.282267
(Iteration 11871 / 39200) loss: 1.807659
(Iteration 11881 / 39200) loss: 1.605493
(Iteration 11891 / 39200) loss: 2.037884
(Iteration 11901 / 39200) loss: 1.194372
(Iteration 11911 / 39200) loss: 1.508116
(Iteration 11921 / 39200) loss: 1.519900
(Iteration 11931 / 39200) loss: 1.378644
(Iteration 11941 / 39200) loss: 1.468439
(Iteration 11951 / 39200) loss: 1.826008
(Iteration 11961 / 39200) loss: 1.696635
(Iteration 11971 / 39200) loss: 1.744527
(Iteration 11981 / 39200) loss: 1.334887
(Iteration 11991 / 39200) loss: 1.481165
(Iteration 12001 / 39200) loss: 1.556049
(Iteration 12011 / 39200) loss: 1.424535
(Iteration 12021 / 39200) loss: 1.569749
(Iteration 12031 / 39200) loss: 1.437585
(Iteration 12041 / 39200) loss: 1.632310
(Iteration 12051 / 39200) loss: 1.568266
(Iteration 12061 / 39200) loss: 1.294896
(Iteration 12071 / 39200) loss: 1.516015
(Iteration 12081 / 39200) loss: 1.683111
(Iteration 12091 / 39200) loss: 1.386060
(Iteration 12101 / 39200) loss: 1.501078
(Iteration 12111 / 39200) loss: 1.354871
(Iteration 12121 / 39200) loss: 1.595069
(Iteration 12131 / 39200) loss: 1.789215
(Iteration 12141 / 39200) loss: 1.734896
(Iteration 12151 / 39200) loss: 1.675962
(Iteration 12161 / 39200) loss: 1.520853
(Iteration 12171 / 39200) loss: 1.515308
(Iteration 12181 / 39200) loss: 1.350591
(Iteration 12191 / 39200) loss: 1.665093

(Iteration 12201 / 39200) loss: 1.955912
(Iteration 12211 / 39200) loss: 1.442031
(Iteration 12221 / 39200) loss: 1.528131
(Iteration 12231 / 39200) loss: 1.902664
(Iteration 12241 / 39200) loss: 1.664930
(Iteration 12251 / 39200) loss: 1.777727
(Iteration 12261 / 39200) loss: 2.023363
(Iteration 12271 / 39200) loss: 1.625155
(Iteration 12281 / 39200) loss: 1.682971
(Iteration 12291 / 39200) loss: 1.574448
(Iteration 12301 / 39200) loss: 1.503664
(Iteration 12311 / 39200) loss: 1.326351
(Iteration 12321 / 39200) loss: 1.695834
(Iteration 12331 / 39200) loss: 1.417729
(Iteration 12341 / 39200) loss: 1.881868
(Iteration 12351 / 39200) loss: 1.434910
(Iteration 12361 / 39200) loss: 1.805485
(Iteration 12371 / 39200) loss: 1.321141
(Iteration 12381 / 39200) loss: 1.666500
(Iteration 12391 / 39200) loss: 2.102212
(Iteration 12401 / 39200) loss: 1.754100
(Iteration 12411 / 39200) loss: 1.430483
(Iteration 12421 / 39200) loss: 1.630067
(Iteration 12431 / 39200) loss: 1.394644
(Iteration 12441 / 39200) loss: 1.891652
(Iteration 12451 / 39200) loss: 1.596454
(Iteration 12461 / 39200) loss: 1.677036
(Iteration 12471 / 39200) loss: 1.888885
(Iteration 12481 / 39200) loss: 1.653610
(Iteration 12491 / 39200) loss: 1.659944
(Iteration 12501 / 39200) loss: 1.744917
(Iteration 12511 / 39200) loss: 1.567518
(Iteration 12521 / 39200) loss: 1.849441
(Iteration 12531 / 39200) loss: 1.380531
(Iteration 12541 / 39200) loss: 1.334627
(Iteration 12551 / 39200) loss: 1.722516
(Iteration 12561 / 39200) loss: 1.205091
(Iteration 12571 / 39200) loss: 1.312924
(Iteration 12581 / 39200) loss: 1.522827
(Iteration 12591 / 39200) loss: 1.563241
(Iteration 12601 / 39200) loss: 1.523385
(Iteration 12611 / 39200) loss: 1.159743
(Iteration 12621 / 39200) loss: 1.520311
(Iteration 12631 / 39200) loss: 1.761085
(Iteration 12641 / 39200) loss: 1.481794
(Iteration 12651 / 39200) loss: 1.458126
(Iteration 12661 / 39200) loss: 1.712816
(Iteration 12671 / 39200) loss: 1.492347

(Iteration 12681 / 39200) loss: 1.348715
(Iteration 12691 / 39200) loss: 1.649827
(Iteration 12701 / 39200) loss: 1.486577
(Iteration 12711 / 39200) loss: 1.315290
(Iteration 12721 / 39200) loss: 1.374487
(Iteration 12731 / 39200) loss: 1.211572
(Iteration 12741 / 39200) loss: 1.593677
(Iteration 12751 / 39200) loss: 1.811467
(Iteration 12761 / 39200) loss: 1.528675
(Iteration 12771 / 39200) loss: 1.484479
(Iteration 12781 / 39200) loss: 1.675903
(Iteration 12791 / 39200) loss: 1.473879
(Iteration 12801 / 39200) loss: 1.767225
(Iteration 12811 / 39200) loss: 1.413038
(Iteration 12821 / 39200) loss: 1.524928
(Iteration 12831 / 39200) loss: 1.727562
(Iteration 12841 / 39200) loss: 1.726293
(Iteration 12851 / 39200) loss: 1.532035
(Iteration 12861 / 39200) loss: 1.646206
(Iteration 12871 / 39200) loss: 1.594379
(Iteration 12881 / 39200) loss: 1.556220
(Iteration 12891 / 39200) loss: 1.758463
(Iteration 12901 / 39200) loss: 1.670148
(Iteration 12911 / 39200) loss: 1.773388
(Iteration 12921 / 39200) loss: 1.545586
(Iteration 12931 / 39200) loss: 1.632222
(Iteration 12941 / 39200) loss: 1.821353
(Iteration 12951 / 39200) loss: 1.350781
(Iteration 12961 / 39200) loss: 1.446705
(Iteration 12971 / 39200) loss: 1.775821
(Iteration 12981 / 39200) loss: 1.785149
(Iteration 12991 / 39200) loss: 1.773164
(Iteration 13001 / 39200) loss: 1.618037
(Iteration 13011 / 39200) loss: 1.481926
(Iteration 13021 / 39200) loss: 1.702272
(Iteration 13031 / 39200) loss: 1.917050
(Iteration 13041 / 39200) loss: 1.419937
(Iteration 13051 / 39200) loss: 1.588065
(Iteration 13061 / 39200) loss: 1.574782
(Iteration 13071 / 39200) loss: 1.592400
(Iteration 13081 / 39200) loss: 1.405808
(Iteration 13091 / 39200) loss: 2.074990
(Iteration 13101 / 39200) loss: 1.787706
(Iteration 13111 / 39200) loss: 1.639775
(Iteration 13121 / 39200) loss: 1.421720
(Iteration 13131 / 39200) loss: 1.710797
(Iteration 13141 / 39200) loss: 1.883171
(Iteration 13151 / 39200) loss: 1.681260

(Iteration 13161 / 39200) loss: 1.459157
(Iteration 13171 / 39200) loss: 1.798761
(Iteration 13181 / 39200) loss: 1.797911
(Iteration 13191 / 39200) loss: 1.619062
(Iteration 13201 / 39200) loss: 1.388722
(Iteration 13211 / 39200) loss: 2.017420
(Iteration 13221 / 39200) loss: 1.443329
(Iteration 13231 / 39200) loss: 1.370849
(Iteration 13241 / 39200) loss: 1.507488
(Iteration 13251 / 39200) loss: 1.816762
(Iteration 13261 / 39200) loss: 1.624984
(Iteration 13271 / 39200) loss: 1.736254
(Iteration 13281 / 39200) loss: 1.756957
(Iteration 13291 / 39200) loss: 1.212543
(Iteration 13301 / 39200) loss: 1.406398
(Iteration 13311 / 39200) loss: 1.878498
(Iteration 13321 / 39200) loss: 1.863850
(Iteration 13331 / 39200) loss: 1.205016
(Iteration 13341 / 39200) loss: 1.512832
(Iteration 13351 / 39200) loss: 1.380709
(Iteration 13361 / 39200) loss: 1.526368
(Iteration 13371 / 39200) loss: 1.305690
(Iteration 13381 / 39200) loss: 1.465005
(Iteration 13391 / 39200) loss: 1.205435
(Iteration 13401 / 39200) loss: 1.433877
(Iteration 13411 / 39200) loss: 1.361898
(Iteration 13421 / 39200) loss: 1.220580
(Iteration 13431 / 39200) loss: 1.523776
(Iteration 13441 / 39200) loss: 1.486099
(Iteration 13451 / 39200) loss: 1.453228
(Iteration 13461 / 39200) loss: 1.595462
(Iteration 13471 / 39200) loss: 1.420161
(Iteration 13481 / 39200) loss: 1.435812
(Iteration 13491 / 39200) loss: 1.529671
(Iteration 13501 / 39200) loss: 1.986825
(Iteration 13511 / 39200) loss: 1.217749
(Iteration 13521 / 39200) loss: 1.685425
(Iteration 13531 / 39200) loss: 1.379632
(Iteration 13541 / 39200) loss: 1.594546
(Iteration 13551 / 39200) loss: 1.490662
(Iteration 13561 / 39200) loss: 1.319623
(Iteration 13571 / 39200) loss: 1.450070
(Iteration 13581 / 39200) loss: 1.519302
(Iteration 13591 / 39200) loss: 1.300259
(Iteration 13601 / 39200) loss: 1.786911
(Iteration 13611 / 39200) loss: 1.449296
(Iteration 13621 / 39200) loss: 1.524744
(Iteration 13631 / 39200) loss: 1.535564

(Iteration 13641 / 39200) loss: 1.451363
(Iteration 13651 / 39200) loss: 1.466089
(Iteration 13661 / 39200) loss: 1.517328
(Iteration 13671 / 39200) loss: 1.507762
(Iteration 13681 / 39200) loss: 1.771205
(Iteration 13691 / 39200) loss: 1.418418
(Iteration 13701 / 39200) loss: 1.346347
(Iteration 13711 / 39200) loss: 1.525211
(Epoch 7 / 20) train acc: 0.546000; val_acc: 0.508000
(Iteration 13721 / 39200) loss: 1.700588
(Iteration 13731 / 39200) loss: 1.531061
(Iteration 13741 / 39200) loss: 1.191579
(Iteration 13751 / 39200) loss: 1.528591
(Iteration 13761 / 39200) loss: 1.706115
(Iteration 13771 / 39200) loss: 1.767526
(Iteration 13781 / 39200) loss: 1.237619
(Iteration 13791 / 39200) loss: 1.982165
(Iteration 13801 / 39200) loss: 1.599379
(Iteration 13811 / 39200) loss: 1.513912
(Iteration 13821 / 39200) loss: 1.767994
(Iteration 13831 / 39200) loss: 1.716622
(Iteration 13841 / 39200) loss: 1.824852
(Iteration 13851 / 39200) loss: 1.902272
(Iteration 13861 / 39200) loss: 1.528377
(Iteration 13871 / 39200) loss: 1.588931
(Iteration 13881 / 39200) loss: 1.229436
(Iteration 13891 / 39200) loss: 1.867734
(Iteration 13901 / 39200) loss: 1.690229
(Iteration 13911 / 39200) loss: 1.706863
(Iteration 13921 / 39200) loss: 1.638397
(Iteration 13931 / 39200) loss: 1.675646
(Iteration 13941 / 39200) loss: 1.269644
(Iteration 13951 / 39200) loss: 1.564058
(Iteration 13961 / 39200) loss: 1.609284
(Iteration 13971 / 39200) loss: 1.560425
(Iteration 13981 / 39200) loss: 1.597592
(Iteration 13991 / 39200) loss: 1.657183
(Iteration 14001 / 39200) loss: 2.108646
(Iteration 14011 / 39200) loss: 1.743309
(Iteration 14021 / 39200) loss: 1.399187
(Iteration 14031 / 39200) loss: 1.694763
(Iteration 14041 / 39200) loss: 1.433125
(Iteration 14051 / 39200) loss: 1.603193
(Iteration 14061 / 39200) loss: 1.420985
(Iteration 14071 / 39200) loss: 1.399519
(Iteration 14081 / 39200) loss: 1.772841
(Iteration 14091 / 39200) loss: 1.854271
(Iteration 14101 / 39200) loss: 1.525200

(Iteration 14111 / 39200) loss: 1.684265
(Iteration 14121 / 39200) loss: 1.222326
(Iteration 14131 / 39200) loss: 1.307105
(Iteration 14141 / 39200) loss: 1.515921
(Iteration 14151 / 39200) loss: 1.702112
(Iteration 14161 / 39200) loss: 1.357204
(Iteration 14171 / 39200) loss: 1.757572
(Iteration 14181 / 39200) loss: 1.723072
(Iteration 14191 / 39200) loss: 1.548342
(Iteration 14201 / 39200) loss: 1.433721
(Iteration 14211 / 39200) loss: 1.360461
(Iteration 14221 / 39200) loss: 1.456996
(Iteration 14231 / 39200) loss: 1.816235
(Iteration 14241 / 39200) loss: 2.050710
(Iteration 14251 / 39200) loss: 1.485383
(Iteration 14261 / 39200) loss: 1.589950
(Iteration 14271 / 39200) loss: 2.033577
(Iteration 14281 / 39200) loss: 1.696839
(Iteration 14291 / 39200) loss: 1.569111
(Iteration 14301 / 39200) loss: 1.311074
(Iteration 14311 / 39200) loss: 1.406878
(Iteration 14321 / 39200) loss: 1.338942
(Iteration 14331 / 39200) loss: 1.511262
(Iteration 14341 / 39200) loss: 1.432660
(Iteration 14351 / 39200) loss: 1.393165
(Iteration 14361 / 39200) loss: 2.112152
(Iteration 14371 / 39200) loss: 1.729584
(Iteration 14381 / 39200) loss: 1.477978
(Iteration 14391 / 39200) loss: 1.421063
(Iteration 14401 / 39200) loss: 1.217135
(Iteration 14411 / 39200) loss: 1.475228
(Iteration 14421 / 39200) loss: 1.430208
(Iteration 14431 / 39200) loss: 1.490527
(Iteration 14441 / 39200) loss: 1.416424
(Iteration 14451 / 39200) loss: 1.869443
(Iteration 14461 / 39200) loss: 1.368015
(Iteration 14471 / 39200) loss: 1.527534
(Iteration 14481 / 39200) loss: 1.371114
(Iteration 14491 / 39200) loss: 1.402764
(Iteration 14501 / 39200) loss: 1.291929
(Iteration 14511 / 39200) loss: 1.352982
(Iteration 14521 / 39200) loss: 1.504501
(Iteration 14531 / 39200) loss: 1.472237
(Iteration 14541 / 39200) loss: 1.443696
(Iteration 14551 / 39200) loss: 1.511030
(Iteration 14561 / 39200) loss: 1.627691
(Iteration 14571 / 39200) loss: 1.721861
(Iteration 14581 / 39200) loss: 1.621194

(Iteration 14591 / 39200) loss: 1.565089
(Iteration 14601 / 39200) loss: 1.459433
(Iteration 14611 / 39200) loss: 1.555281
(Iteration 14621 / 39200) loss: 1.837779
(Iteration 14631 / 39200) loss: 1.935853
(Iteration 14641 / 39200) loss: 1.338605
(Iteration 14651 / 39200) loss: 1.827601
(Iteration 14661 / 39200) loss: 1.858401
(Iteration 14671 / 39200) loss: 1.709396
(Iteration 14681 / 39200) loss: 1.733823
(Iteration 14691 / 39200) loss: 1.380893
(Iteration 14701 / 39200) loss: 1.798899
(Iteration 14711 / 39200) loss: 1.606850
(Iteration 14721 / 39200) loss: 1.300040
(Iteration 14731 / 39200) loss: 1.462396
(Iteration 14741 / 39200) loss: 1.637687
(Iteration 14751 / 39200) loss: 1.356281
(Iteration 14761 / 39200) loss: 1.264915
(Iteration 14771 / 39200) loss: 1.507848
(Iteration 14781 / 39200) loss: 1.734811
(Iteration 14791 / 39200) loss: 2.087633
(Iteration 14801 / 39200) loss: 1.610297
(Iteration 14811 / 39200) loss: 1.434873
(Iteration 14821 / 39200) loss: 1.391842
(Iteration 14831 / 39200) loss: 1.941342
(Iteration 14841 / 39200) loss: 1.471463
(Iteration 14851 / 39200) loss: 1.412969
(Iteration 14861 / 39200) loss: 1.606755
(Iteration 14871 / 39200) loss: 1.766030
(Iteration 14881 / 39200) loss: 1.437245
(Iteration 14891 / 39200) loss: 1.242937
(Iteration 14901 / 39200) loss: 1.737401
(Iteration 14911 / 39200) loss: 1.979550
(Iteration 14921 / 39200) loss: 1.565604
(Iteration 14931 / 39200) loss: 1.537236
(Iteration 14941 / 39200) loss: 1.680533
(Iteration 14951 / 39200) loss: 1.370795
(Iteration 14961 / 39200) loss: 1.477840
(Iteration 14971 / 39200) loss: 1.353418
(Iteration 14981 / 39200) loss: 1.432053
(Iteration 14991 / 39200) loss: 1.510456
(Iteration 15001 / 39200) loss: 1.565010
(Iteration 15011 / 39200) loss: 1.526648
(Iteration 15021 / 39200) loss: 1.865565
(Iteration 15031 / 39200) loss: 1.519632
(Iteration 15041 / 39200) loss: 1.641802
(Iteration 15051 / 39200) loss: 1.717407
(Iteration 15061 / 39200) loss: 1.102395

(Iteration 15071 / 39200) loss: 1.555755
(Iteration 15081 / 39200) loss: 1.376321
(Iteration 15091 / 39200) loss: 1.863844
(Iteration 15101 / 39200) loss: 1.653713
(Iteration 15111 / 39200) loss: 1.562820
(Iteration 15121 / 39200) loss: 1.572661
(Iteration 15131 / 39200) loss: 1.530488
(Iteration 15141 / 39200) loss: 1.507597
(Iteration 15151 / 39200) loss: 1.512742
(Iteration 15161 / 39200) loss: 2.083230
(Iteration 15171 / 39200) loss: 1.608576
(Iteration 15181 / 39200) loss: 1.381574
(Iteration 15191 / 39200) loss: 1.481466
(Iteration 15201 / 39200) loss: 1.767587
(Iteration 15211 / 39200) loss: 1.836417
(Iteration 15221 / 39200) loss: 1.320763
(Iteration 15231 / 39200) loss: 1.860945
(Iteration 15241 / 39200) loss: 2.028786
(Iteration 15251 / 39200) loss: 1.774844
(Iteration 15261 / 39200) loss: 1.486299
(Iteration 15271 / 39200) loss: 1.641923
(Iteration 15281 / 39200) loss: 1.477096
(Iteration 15291 / 39200) loss: 1.809364
(Iteration 15301 / 39200) loss: 1.140640
(Iteration 15311 / 39200) loss: 1.370661
(Iteration 15321 / 39200) loss: 1.727090
(Iteration 15331 / 39200) loss: 1.480701
(Iteration 15341 / 39200) loss: 1.474167
(Iteration 15351 / 39200) loss: 1.488589
(Iteration 15361 / 39200) loss: 1.575389
(Iteration 15371 / 39200) loss: 1.567741
(Iteration 15381 / 39200) loss: 1.431521
(Iteration 15391 / 39200) loss: 1.518121
(Iteration 15401 / 39200) loss: 1.452415
(Iteration 15411 / 39200) loss: 1.809436
(Iteration 15421 / 39200) loss: 1.432432
(Iteration 15431 / 39200) loss: 1.279708
(Iteration 15441 / 39200) loss: 1.371862
(Iteration 15451 / 39200) loss: 1.549997
(Iteration 15461 / 39200) loss: 1.534437
(Iteration 15471 / 39200) loss: 1.460644
(Iteration 15481 / 39200) loss: 1.637727
(Iteration 15491 / 39200) loss: 1.504844
(Iteration 15501 / 39200) loss: 1.493954
(Iteration 15511 / 39200) loss: 1.649379
(Iteration 15521 / 39200) loss: 1.190072
(Iteration 15531 / 39200) loss: 1.524966
(Iteration 15541 / 39200) loss: 1.249448

(Iteration 15551 / 39200) loss: 1.604259
(Iteration 15561 / 39200) loss: 1.545427
(Iteration 15571 / 39200) loss: 1.952673
(Iteration 15581 / 39200) loss: 1.515184
(Iteration 15591 / 39200) loss: 1.725428
(Iteration 15601 / 39200) loss: 1.511805
(Iteration 15611 / 39200) loss: 1.471241
(Iteration 15621 / 39200) loss: 1.828081
(Iteration 15631 / 39200) loss: 1.408973
(Iteration 15641 / 39200) loss: 1.581774
(Iteration 15651 / 39200) loss: 1.799127
(Iteration 15661 / 39200) loss: 1.235101
(Iteration 15671 / 39200) loss: 1.436312
(Epoch 8 / 20) train acc: 0.506000; val_acc: 0.502000
(Iteration 15681 / 39200) loss: 1.608108
(Iteration 15691 / 39200) loss: 1.473902
(Iteration 15701 / 39200) loss: 1.439191
(Iteration 15711 / 39200) loss: 1.411505
(Iteration 15721 / 39200) loss: 1.998878
(Iteration 15731 / 39200) loss: 1.477904
(Iteration 15741 / 39200) loss: 1.867019
(Iteration 15751 / 39200) loss: 1.364838
(Iteration 15761 / 39200) loss: 1.515023
(Iteration 15771 / 39200) loss: 1.339235
(Iteration 15781 / 39200) loss: 1.369993
(Iteration 15791 / 39200) loss: 1.533790
(Iteration 15801 / 39200) loss: 1.558706
(Iteration 15811 / 39200) loss: 1.847058
(Iteration 15821 / 39200) loss: 1.582357
(Iteration 15831 / 39200) loss: 1.420349
(Iteration 15841 / 39200) loss: 1.378392
(Iteration 15851 / 39200) loss: 1.517349
(Iteration 15861 / 39200) loss: 1.657601
(Iteration 15871 / 39200) loss: 1.467990
(Iteration 15881 / 39200) loss: 1.990254
(Iteration 15891 / 39200) loss: 1.871500
(Iteration 15901 / 39200) loss: 1.584338
(Iteration 15911 / 39200) loss: 1.749538
(Iteration 15921 / 39200) loss: 1.316145
(Iteration 15931 / 39200) loss: 1.595045
(Iteration 15941 / 39200) loss: 1.491288
(Iteration 15951 / 39200) loss: 1.830980
(Iteration 15961 / 39200) loss: 1.737104
(Iteration 15971 / 39200) loss: 1.429235
(Iteration 15981 / 39200) loss: 1.867706
(Iteration 15991 / 39200) loss: 1.421002
(Iteration 16001 / 39200) loss: 1.644703
(Iteration 16011 / 39200) loss: 1.745142

(Iteration 16021 / 39200) loss: 1.804240
(Iteration 16031 / 39200) loss: 1.457362
(Iteration 16041 / 39200) loss: 1.661276
(Iteration 16051 / 39200) loss: 1.492587
(Iteration 16061 / 39200) loss: 1.806048
(Iteration 16071 / 39200) loss: 1.896299
(Iteration 16081 / 39200) loss: 1.768726
(Iteration 16091 / 39200) loss: 1.513224
(Iteration 16101 / 39200) loss: 1.612608
(Iteration 16111 / 39200) loss: 1.407005
(Iteration 16121 / 39200) loss: 2.079855
(Iteration 16131 / 39200) loss: 1.698566
(Iteration 16141 / 39200) loss: 1.261658
(Iteration 16151 / 39200) loss: 1.832989
(Iteration 16161 / 39200) loss: 1.447431
(Iteration 16171 / 39200) loss: 1.498670
(Iteration 16181 / 39200) loss: 1.760077
(Iteration 16191 / 39200) loss: 1.400195
(Iteration 16201 / 39200) loss: 1.680129
(Iteration 16211 / 39200) loss: 1.928362
(Iteration 16221 / 39200) loss: 1.657879
(Iteration 16231 / 39200) loss: 1.732319
(Iteration 16241 / 39200) loss: 1.560435
(Iteration 16251 / 39200) loss: 1.486551
(Iteration 16261 / 39200) loss: 1.836583
(Iteration 16271 / 39200) loss: 1.791760
(Iteration 16281 / 39200) loss: 1.455616
(Iteration 16291 / 39200) loss: 1.318613
(Iteration 16301 / 39200) loss: 1.255908
(Iteration 16311 / 39200) loss: 1.722605
(Iteration 16321 / 39200) loss: 1.729015
(Iteration 16331 / 39200) loss: 1.592784
(Iteration 16341 / 39200) loss: 1.670577
(Iteration 16351 / 39200) loss: 1.470004
(Iteration 16361 / 39200) loss: 1.444498
(Iteration 16371 / 39200) loss: 1.584320
(Iteration 16381 / 39200) loss: 1.690874
(Iteration 16391 / 39200) loss: 1.469808
(Iteration 16401 / 39200) loss: 1.441522
(Iteration 16411 / 39200) loss: 1.960765
(Iteration 16421 / 39200) loss: 1.403385
(Iteration 16431 / 39200) loss: 1.544866
(Iteration 16441 / 39200) loss: 1.787970
(Iteration 16451 / 39200) loss: 1.437997
(Iteration 16461 / 39200) loss: 1.722451
(Iteration 16471 / 39200) loss: 1.772573
(Iteration 16481 / 39200) loss: 1.974584
(Iteration 16491 / 39200) loss: 1.543202

(Iteration 16501 / 39200) loss: 1.471716
(Iteration 16511 / 39200) loss: 1.833745
(Iteration 16521 / 39200) loss: 1.296815
(Iteration 16531 / 39200) loss: 1.279908
(Iteration 16541 / 39200) loss: 1.799629
(Iteration 16551 / 39200) loss: 1.740199
(Iteration 16561 / 39200) loss: 1.380732
(Iteration 16571 / 39200) loss: 1.759471
(Iteration 16581 / 39200) loss: 1.481676
(Iteration 16591 / 39200) loss: 1.502158
(Iteration 16601 / 39200) loss: 1.729157
(Iteration 16611 / 39200) loss: 1.468250
(Iteration 16621 / 39200) loss: 1.332445
(Iteration 16631 / 39200) loss: 1.463373
(Iteration 16641 / 39200) loss: 1.539792
(Iteration 16651 / 39200) loss: 1.526927
(Iteration 16661 / 39200) loss: 1.445262
(Iteration 16671 / 39200) loss: 1.660656
(Iteration 16681 / 39200) loss: 1.410037
(Iteration 16691 / 39200) loss: 1.865080
(Iteration 16701 / 39200) loss: 1.879891
(Iteration 16711 / 39200) loss: 1.822657
(Iteration 16721 / 39200) loss: 1.887238
(Iteration 16731 / 39200) loss: 1.621944
(Iteration 16741 / 39200) loss: 1.229594
(Iteration 16751 / 39200) loss: 1.788733
(Iteration 16761 / 39200) loss: 1.327403
(Iteration 16771 / 39200) loss: 1.606865
(Iteration 16781 / 39200) loss: 1.276268
(Iteration 16791 / 39200) loss: 1.757397
(Iteration 16801 / 39200) loss: 1.253052
(Iteration 16811 / 39200) loss: 1.312248
(Iteration 16821 / 39200) loss: 1.208396
(Iteration 16831 / 39200) loss: 1.414077
(Iteration 16841 / 39200) loss: 1.760474
(Iteration 16851 / 39200) loss: 1.869151
(Iteration 16861 / 39200) loss: 1.688082
(Iteration 16871 / 39200) loss: 1.876815
(Iteration 16881 / 39200) loss: 1.460795
(Iteration 16891 / 39200) loss: 1.266702
(Iteration 16901 / 39200) loss: 1.859629
(Iteration 16911 / 39200) loss: 1.742680
(Iteration 16921 / 39200) loss: 1.353756
(Iteration 16931 / 39200) loss: 1.545303
(Iteration 16941 / 39200) loss: 1.662387
(Iteration 16951 / 39200) loss: 1.786188
(Iteration 16961 / 39200) loss: 1.378358
(Iteration 16971 / 39200) loss: 1.557582

(Iteration 16981 / 39200) loss: 1.517250
(Iteration 16991 / 39200) loss: 1.424817
(Iteration 17001 / 39200) loss: 1.492198
(Iteration 17011 / 39200) loss: 1.678658
(Iteration 17021 / 39200) loss: 1.823868
(Iteration 17031 / 39200) loss: 1.556643
(Iteration 17041 / 39200) loss: 1.605796
(Iteration 17051 / 39200) loss: 1.796482
(Iteration 17061 / 39200) loss: 1.442617
(Iteration 17071 / 39200) loss: 1.343311
(Iteration 17081 / 39200) loss: 1.443891
(Iteration 17091 / 39200) loss: 1.931644
(Iteration 17101 / 39200) loss: 1.361581
(Iteration 17111 / 39200) loss: 1.333799
(Iteration 17121 / 39200) loss: 1.579568
(Iteration 17131 / 39200) loss: 1.663275
(Iteration 17141 / 39200) loss: 1.653203
(Iteration 17151 / 39200) loss: 1.294242
(Iteration 17161 / 39200) loss: 1.768633
(Iteration 17171 / 39200) loss: 1.717886
(Iteration 17181 / 39200) loss: 1.335052
(Iteration 17191 / 39200) loss: 1.395461
(Iteration 17201 / 39200) loss: 1.559508
(Iteration 17211 / 39200) loss: 1.374906
(Iteration 17221 / 39200) loss: 1.713657
(Iteration 17231 / 39200) loss: 1.684872
(Iteration 17241 / 39200) loss: 1.602163
(Iteration 17251 / 39200) loss: 1.596827
(Iteration 17261 / 39200) loss: 1.516822
(Iteration 17271 / 39200) loss: 1.993471
(Iteration 17281 / 39200) loss: 1.585773
(Iteration 17291 / 39200) loss: 1.310608
(Iteration 17301 / 39200) loss: 1.588524
(Iteration 17311 / 39200) loss: 1.974020
(Iteration 17321 / 39200) loss: 1.272896
(Iteration 17331 / 39200) loss: 1.280974
(Iteration 17341 / 39200) loss: 1.542654
(Iteration 17351 / 39200) loss: 1.527865
(Iteration 17361 / 39200) loss: 1.655022
(Iteration 17371 / 39200) loss: 1.657093
(Iteration 17381 / 39200) loss: 1.338953
(Iteration 17391 / 39200) loss: 1.634243
(Iteration 17401 / 39200) loss: 1.416381
(Iteration 17411 / 39200) loss: 1.508514
(Iteration 17421 / 39200) loss: 1.238718
(Iteration 17431 / 39200) loss: 1.693599
(Iteration 17441 / 39200) loss: 1.827815
(Iteration 17451 / 39200) loss: 1.519816

(Iteration 17461 / 39200) loss: 1.523877
(Iteration 17471 / 39200) loss: 1.249649
(Iteration 17481 / 39200) loss: 1.792274
(Iteration 17491 / 39200) loss: 1.602160
(Iteration 17501 / 39200) loss: 1.594281
(Iteration 17511 / 39200) loss: 1.613309
(Iteration 17521 / 39200) loss: 1.881015
(Iteration 17531 / 39200) loss: 1.548631
(Iteration 17541 / 39200) loss: 1.237111
(Iteration 17551 / 39200) loss: 1.669450
(Iteration 17561 / 39200) loss: 1.372589
(Iteration 17571 / 39200) loss: 1.210272
(Iteration 17581 / 39200) loss: 1.416546
(Iteration 17591 / 39200) loss: 1.450066
(Iteration 17601 / 39200) loss: 1.592749
(Iteration 17611 / 39200) loss: 1.567500
(Iteration 17621 / 39200) loss: 1.408635
(Iteration 17631 / 39200) loss: 1.723082
(Epoch 9 / 20) train acc: 0.532000; val_acc: 0.509000
(Iteration 17641 / 39200) loss: 1.703475
(Iteration 17651 / 39200) loss: 1.231895
(Iteration 17661 / 39200) loss: 1.739053
(Iteration 17671 / 39200) loss: 1.724047
(Iteration 17681 / 39200) loss: 1.910654
(Iteration 17691 / 39200) loss: 1.636152
(Iteration 17701 / 39200) loss: 1.512506
(Iteration 17711 / 39200) loss: 1.831487
(Iteration 17721 / 39200) loss: 1.698194
(Iteration 17731 / 39200) loss: 1.388685
(Iteration 17741 / 39200) loss: 1.748956
(Iteration 17751 / 39200) loss: 1.656668
(Iteration 17761 / 39200) loss: 1.531054
(Iteration 17771 / 39200) loss: 1.311657
(Iteration 17781 / 39200) loss: 1.646832
(Iteration 17791 / 39200) loss: 1.325480
(Iteration 17801 / 39200) loss: 1.427501
(Iteration 17811 / 39200) loss: 1.809035
(Iteration 17821 / 39200) loss: 1.249457
(Iteration 17831 / 39200) loss: 1.732028
(Iteration 17841 / 39200) loss: 1.378731
(Iteration 17851 / 39200) loss: 1.415548
(Iteration 17861 / 39200) loss: 1.364210
(Iteration 17871 / 39200) loss: 1.814052
(Iteration 17881 / 39200) loss: 1.433012
(Iteration 17891 / 39200) loss: 1.086657
(Iteration 17901 / 39200) loss: 1.353531
(Iteration 17911 / 39200) loss: 1.599149
(Iteration 17921 / 39200) loss: 1.495785

(Iteration 17931 / 39200) loss: 1.683150
(Iteration 17941 / 39200) loss: 1.621028
(Iteration 17951 / 39200) loss: 1.573083
(Iteration 17961 / 39200) loss: 1.167121
(Iteration 17971 / 39200) loss: 1.424365
(Iteration 17981 / 39200) loss: 1.889306
(Iteration 17991 / 39200) loss: 1.914884
(Iteration 18001 / 39200) loss: 1.294499
(Iteration 18011 / 39200) loss: 1.683816
(Iteration 18021 / 39200) loss: 1.584875
(Iteration 18031 / 39200) loss: 1.435161
(Iteration 18041 / 39200) loss: 1.530145
(Iteration 18051 / 39200) loss: 1.611728
(Iteration 18061 / 39200) loss: 1.789856
(Iteration 18071 / 39200) loss: 1.484231
(Iteration 18081 / 39200) loss: 1.523161
(Iteration 18091 / 39200) loss: 1.802553
(Iteration 18101 / 39200) loss: 1.807630
(Iteration 18111 / 39200) loss: 1.803201
(Iteration 18121 / 39200) loss: 1.263774
(Iteration 18131 / 39200) loss: 1.747315
(Iteration 18141 / 39200) loss: 1.743821
(Iteration 18151 / 39200) loss: 1.773489
(Iteration 18161 / 39200) loss: 2.051609
(Iteration 18171 / 39200) loss: 2.089517
(Iteration 18181 / 39200) loss: 1.387161
(Iteration 18191 / 39200) loss: 1.430146
(Iteration 18201 / 39200) loss: 1.682222
(Iteration 18211 / 39200) loss: 1.688748
(Iteration 18221 / 39200) loss: 1.533372
(Iteration 18231 / 39200) loss: 1.559833
(Iteration 18241 / 39200) loss: 1.557085
(Iteration 18251 / 39200) loss: 1.278538
(Iteration 18261 / 39200) loss: 1.236233
(Iteration 18271 / 39200) loss: 1.472310
(Iteration 18281 / 39200) loss: 2.221531
(Iteration 18291 / 39200) loss: 1.665695
(Iteration 18301 / 39200) loss: 1.306743
(Iteration 18311 / 39200) loss: 1.707616
(Iteration 18321 / 39200) loss: 1.603170
(Iteration 18331 / 39200) loss: 1.539905
(Iteration 18341 / 39200) loss: 1.424596
(Iteration 18351 / 39200) loss: 1.473355
(Iteration 18361 / 39200) loss: 1.968670
(Iteration 18371 / 39200) loss: 1.514644
(Iteration 18381 / 39200) loss: 1.562063
(Iteration 18391 / 39200) loss: 1.505927
(Iteration 18401 / 39200) loss: 1.507047

(Iteration 18411 / 39200) loss: 1.653233
(Iteration 18421 / 39200) loss: 1.426373
(Iteration 18431 / 39200) loss: 1.606694
(Iteration 18441 / 39200) loss: 1.717518
(Iteration 18451 / 39200) loss: 1.537141
(Iteration 18461 / 39200) loss: 1.624928
(Iteration 18471 / 39200) loss: 1.220996
(Iteration 18481 / 39200) loss: 1.597710
(Iteration 18491 / 39200) loss: 1.404197
(Iteration 18501 / 39200) loss: 1.536845
(Iteration 18511 / 39200) loss: 1.431839
(Iteration 18521 / 39200) loss: 1.655821
(Iteration 18531 / 39200) loss: 1.561472
(Iteration 18541 / 39200) loss: 1.746721
(Iteration 18551 / 39200) loss: 1.602225
(Iteration 18561 / 39200) loss: 1.544813
(Iteration 18571 / 39200) loss: 1.290690
(Iteration 18581 / 39200) loss: 1.712130
(Iteration 18591 / 39200) loss: 1.568346
(Iteration 18601 / 39200) loss: 1.264280
(Iteration 18611 / 39200) loss: 1.759644
(Iteration 18621 / 39200) loss: 1.296597
(Iteration 18631 / 39200) loss: 1.643646
(Iteration 18641 / 39200) loss: 1.388424
(Iteration 18651 / 39200) loss: 2.062738
(Iteration 18661 / 39200) loss: 1.875880
(Iteration 18671 / 39200) loss: 1.651000
(Iteration 18681 / 39200) loss: 1.370430
(Iteration 18691 / 39200) loss: 1.533293
(Iteration 18701 / 39200) loss: 1.570687
(Iteration 18711 / 39200) loss: 1.439172
(Iteration 18721 / 39200) loss: 1.686039
(Iteration 18731 / 39200) loss: 1.852434
(Iteration 18741 / 39200) loss: 1.587288
(Iteration 18751 / 39200) loss: 1.416163
(Iteration 18761 / 39200) loss: 1.380290
(Iteration 18771 / 39200) loss: 1.409318
(Iteration 18781 / 39200) loss: 1.395653
(Iteration 18791 / 39200) loss: 1.429576
(Iteration 18801 / 39200) loss: 1.778653
(Iteration 18811 / 39200) loss: 1.539178
(Iteration 18821 / 39200) loss: 1.340117
(Iteration 18831 / 39200) loss: 1.147756
(Iteration 18841 / 39200) loss: 1.540297
(Iteration 18851 / 39200) loss: 1.731077
(Iteration 18861 / 39200) loss: 1.658088
(Iteration 18871 / 39200) loss: 1.457723
(Iteration 18881 / 39200) loss: 1.448876

(Iteration 18891 / 39200) loss: 1.366481
(Iteration 18901 / 39200) loss: 1.639848
(Iteration 18911 / 39200) loss: 1.771703
(Iteration 18921 / 39200) loss: 1.514471
(Iteration 18931 / 39200) loss: 1.706243
(Iteration 18941 / 39200) loss: 1.511879
(Iteration 18951 / 39200) loss: 1.544635
(Iteration 18961 / 39200) loss: 1.282747
(Iteration 18971 / 39200) loss: 1.660146
(Iteration 18981 / 39200) loss: 1.510971
(Iteration 18991 / 39200) loss: 1.827239
(Iteration 19001 / 39200) loss: 1.597560
(Iteration 19011 / 39200) loss: 1.813908
(Iteration 19021 / 39200) loss: 1.808180
(Iteration 19031 / 39200) loss: 1.414155
(Iteration 19041 / 39200) loss: 1.710545
(Iteration 19051 / 39200) loss: 1.727949
(Iteration 19061 / 39200) loss: 1.649623
(Iteration 19071 / 39200) loss: 1.658746
(Iteration 19081 / 39200) loss: 1.490302
(Iteration 19091 / 39200) loss: 1.454977
(Iteration 19101 / 39200) loss: 1.231802
(Iteration 19111 / 39200) loss: 1.619274
(Iteration 19121 / 39200) loss: 1.914069
(Iteration 19131 / 39200) loss: 1.159261
(Iteration 19141 / 39200) loss: 1.358402
(Iteration 19151 / 39200) loss: 1.428623
(Iteration 19161 / 39200) loss: 2.024965
(Iteration 19171 / 39200) loss: 1.728116
(Iteration 19181 / 39200) loss: 1.192509
(Iteration 19191 / 39200) loss: 1.583524
(Iteration 19201 / 39200) loss: 1.462192
(Iteration 19211 / 39200) loss: 1.529047
(Iteration 19221 / 39200) loss: 1.268439
(Iteration 19231 / 39200) loss: 1.625467
(Iteration 19241 / 39200) loss: 1.604947
(Iteration 19251 / 39200) loss: 1.471931
(Iteration 19261 / 39200) loss: 1.344008
(Iteration 19271 / 39200) loss: 1.286757
(Iteration 19281 / 39200) loss: 1.570609
(Iteration 19291 / 39200) loss: 1.354937
(Iteration 19301 / 39200) loss: 1.400785
(Iteration 19311 / 39200) loss: 1.460461
(Iteration 19321 / 39200) loss: 1.463257
(Iteration 19331 / 39200) loss: 1.902419
(Iteration 19341 / 39200) loss: 1.848293
(Iteration 19351 / 39200) loss: 1.350749
(Iteration 19361 / 39200) loss: 1.461289

(Iteration 19371 / 39200) loss: 1.388342
(Iteration 19381 / 39200) loss: 1.525327
(Iteration 19391 / 39200) loss: 1.868208
(Iteration 19401 / 39200) loss: 1.853494
(Iteration 19411 / 39200) loss: 1.501023
(Iteration 19421 / 39200) loss: 1.106651
(Iteration 19431 / 39200) loss: 1.500272
(Iteration 19441 / 39200) loss: 1.691398
(Iteration 19451 / 39200) loss: 1.797111
(Iteration 19461 / 39200) loss: 1.532640
(Iteration 19471 / 39200) loss: 1.501898
(Iteration 19481 / 39200) loss: 1.704835
(Iteration 19491 / 39200) loss: 1.223309
(Iteration 19501 / 39200) loss: 1.682776
(Iteration 19511 / 39200) loss: 1.536686
(Iteration 19521 / 39200) loss: 1.379468
(Iteration 19531 / 39200) loss: 1.601168
(Iteration 19541 / 39200) loss: 1.282269
(Iteration 19551 / 39200) loss: 1.687668
(Iteration 19561 / 39200) loss: 1.707186
(Iteration 19571 / 39200) loss: 1.553544
(Iteration 19581 / 39200) loss: 1.424703
(Iteration 19591 / 39200) loss: 1.567278
(Epoch 10 / 20) train acc: 0.550000; val_acc: 0.495000
(Iteration 19601 / 39200) loss: 1.206832
(Iteration 19611 / 39200) loss: 1.593073
(Iteration 19621 / 39200) loss: 1.445860
(Iteration 19631 / 39200) loss: 1.407474
(Iteration 19641 / 39200) loss: 1.562365
(Iteration 19651 / 39200) loss: 1.620773
(Iteration 19661 / 39200) loss: 1.511729
(Iteration 19671 / 39200) loss: 1.461100
(Iteration 19681 / 39200) loss: 1.922788
(Iteration 19691 / 39200) loss: 1.672181
(Iteration 19701 / 39200) loss: 1.797014
(Iteration 19711 / 39200) loss: 1.565106
(Iteration 19721 / 39200) loss: 1.766713
(Iteration 19731 / 39200) loss: 1.695096
(Iteration 19741 / 39200) loss: 1.296836
(Iteration 19751 / 39200) loss: 1.761929
(Iteration 19761 / 39200) loss: 1.871508
(Iteration 19771 / 39200) loss: 1.470637
(Iteration 19781 / 39200) loss: 1.845807
(Iteration 19791 / 39200) loss: 1.615254
(Iteration 19801 / 39200) loss: 1.824172
(Iteration 19811 / 39200) loss: 1.278676
(Iteration 19821 / 39200) loss: 1.178942
(Iteration 19831 / 39200) loss: 1.804460

(Iteration 19841 / 39200) loss: 1.382224
(Iteration 19851 / 39200) loss: 1.310559
(Iteration 19861 / 39200) loss: 1.664446
(Iteration 19871 / 39200) loss: 1.773069
(Iteration 19881 / 39200) loss: 1.737379
(Iteration 19891 / 39200) loss: 1.516684
(Iteration 19901 / 39200) loss: 1.955824
(Iteration 19911 / 39200) loss: 1.438565
(Iteration 19921 / 39200) loss: 1.372789
(Iteration 19931 / 39200) loss: 1.193568
(Iteration 19941 / 39200) loss: 1.547576
(Iteration 19951 / 39200) loss: 1.375224
(Iteration 19961 / 39200) loss: 1.740504
(Iteration 19971 / 39200) loss: 1.489688
(Iteration 19981 / 39200) loss: 1.569379
(Iteration 19991 / 39200) loss: 1.584218
(Iteration 20001 / 39200) loss: 1.494722
(Iteration 20011 / 39200) loss: 1.608307
(Iteration 20021 / 39200) loss: 1.474500
(Iteration 20031 / 39200) loss: 1.614767
(Iteration 20041 / 39200) loss: 1.574932
(Iteration 20051 / 39200) loss: 1.673047
(Iteration 20061 / 39200) loss: 1.509786
(Iteration 20071 / 39200) loss: 1.571964
(Iteration 20081 / 39200) loss: 1.663833
(Iteration 20091 / 39200) loss: 1.452229
(Iteration 20101 / 39200) loss: 1.632658
(Iteration 20111 / 39200) loss: 1.557574
(Iteration 20121 / 39200) loss: 1.409945
(Iteration 20131 / 39200) loss: 1.605669
(Iteration 20141 / 39200) loss: 1.588301
(Iteration 20151 / 39200) loss: 1.406832
(Iteration 20161 / 39200) loss: 1.340611
(Iteration 20171 / 39200) loss: 1.551807
(Iteration 20181 / 39200) loss: 1.634663
(Iteration 20191 / 39200) loss: 1.389456
(Iteration 20201 / 39200) loss: 1.471127
(Iteration 20211 / 39200) loss: 1.859801
(Iteration 20221 / 39200) loss: 1.522738
(Iteration 20231 / 39200) loss: 1.356110
(Iteration 20241 / 39200) loss: 1.721280
(Iteration 20251 / 39200) loss: 1.536079
(Iteration 20261 / 39200) loss: 1.652822
(Iteration 20271 / 39200) loss: 1.343917
(Iteration 20281 / 39200) loss: 1.738658
(Iteration 20291 / 39200) loss: 1.862844
(Iteration 20301 / 39200) loss: 1.615434
(Iteration 20311 / 39200) loss: 1.588557

(Iteration 20321 / 39200) loss: 1.676098
(Iteration 20331 / 39200) loss: 1.520590
(Iteration 20341 / 39200) loss: 1.311061
(Iteration 20351 / 39200) loss: 1.181924
(Iteration 20361 / 39200) loss: 1.621488
(Iteration 20371 / 39200) loss: 1.693987
(Iteration 20381 / 39200) loss: 1.495481
(Iteration 20391 / 39200) loss: 1.600471
(Iteration 20401 / 39200) loss: 1.111175
(Iteration 20411 / 39200) loss: 2.090662
(Iteration 20421 / 39200) loss: 1.652854
(Iteration 20431 / 39200) loss: 1.676455
(Iteration 20441 / 39200) loss: 1.515741
(Iteration 20451 / 39200) loss: 1.567300
(Iteration 20461 / 39200) loss: 1.619683
(Iteration 20471 / 39200) loss: 1.108299
(Iteration 20481 / 39200) loss: 1.449359
(Iteration 20491 / 39200) loss: 1.690100
(Iteration 20501 / 39200) loss: 1.854559
(Iteration 20511 / 39200) loss: 1.690850
(Iteration 20521 / 39200) loss: 1.671024
(Iteration 20531 / 39200) loss: 1.284696
(Iteration 20541 / 39200) loss: 1.600300
(Iteration 20551 / 39200) loss: 1.467362
(Iteration 20561 / 39200) loss: 1.563900
(Iteration 20571 / 39200) loss: 1.435213
(Iteration 20581 / 39200) loss: 1.549889
(Iteration 20591 / 39200) loss: 1.872143
(Iteration 20601 / 39200) loss: 1.604330
(Iteration 20611 / 39200) loss: 1.865114
(Iteration 20621 / 39200) loss: 1.673353
(Iteration 20631 / 39200) loss: 1.422128
(Iteration 20641 / 39200) loss: 1.534347
(Iteration 20651 / 39200) loss: 1.410080
(Iteration 20661 / 39200) loss: 1.688797
(Iteration 20671 / 39200) loss: 1.402869
(Iteration 20681 / 39200) loss: 2.146608
(Iteration 20691 / 39200) loss: 1.848194
(Iteration 20701 / 39200) loss: 1.500602
(Iteration 20711 / 39200) loss: 1.130700
(Iteration 20721 / 39200) loss: 1.714925
(Iteration 20731 / 39200) loss: 1.286614
(Iteration 20741 / 39200) loss: 1.521424
(Iteration 20751 / 39200) loss: 1.014274
(Iteration 20761 / 39200) loss: 1.299426
(Iteration 20771 / 39200) loss: 1.680872
(Iteration 20781 / 39200) loss: 1.603119
(Iteration 20791 / 39200) loss: 1.788298

(Iteration 20801 / 39200) loss: 1.844659
(Iteration 20811 / 39200) loss: 1.677598
(Iteration 20821 / 39200) loss: 1.899715
(Iteration 20831 / 39200) loss: 1.769411
(Iteration 20841 / 39200) loss: 1.447278
(Iteration 20851 / 39200) loss: 1.607300
(Iteration 20861 / 39200) loss: 1.774518
(Iteration 20871 / 39200) loss: 1.710772
(Iteration 20881 / 39200) loss: 1.571055
(Iteration 20891 / 39200) loss: 1.776613
(Iteration 20901 / 39200) loss: 1.360991
(Iteration 20911 / 39200) loss: 1.382028
(Iteration 20921 / 39200) loss: 1.311177
(Iteration 20931 / 39200) loss: 1.553846
(Iteration 20941 / 39200) loss: 1.347801
(Iteration 20951 / 39200) loss: 1.397642
(Iteration 20961 / 39200) loss: 1.805547
(Iteration 20971 / 39200) loss: 1.300343
(Iteration 20981 / 39200) loss: 1.293945
(Iteration 20991 / 39200) loss: 1.853996
(Iteration 21001 / 39200) loss: 1.862814
(Iteration 21011 / 39200) loss: 1.288365
(Iteration 21021 / 39200) loss: 1.138797
(Iteration 21031 / 39200) loss: 1.404592
(Iteration 21041 / 39200) loss: 1.406475
(Iteration 21051 / 39200) loss: 1.681938
(Iteration 21061 / 39200) loss: 1.634276
(Iteration 21071 / 39200) loss: 1.315723
(Iteration 21081 / 39200) loss: 1.387830
(Iteration 21091 / 39200) loss: 1.434241
(Iteration 21101 / 39200) loss: 1.256573
(Iteration 21111 / 39200) loss: 1.543238
(Iteration 21121 / 39200) loss: 1.557938
(Iteration 21131 / 39200) loss: 1.727192
(Iteration 21141 / 39200) loss: 1.691276
(Iteration 21151 / 39200) loss: 1.620194
(Iteration 21161 / 39200) loss: 1.439522
(Iteration 21171 / 39200) loss: 1.683045
(Iteration 21181 / 39200) loss: 1.445737
(Iteration 21191 / 39200) loss: 1.485608
(Iteration 21201 / 39200) loss: 1.542679
(Iteration 21211 / 39200) loss: 1.157451
(Iteration 21221 / 39200) loss: 1.713006
(Iteration 21231 / 39200) loss: 1.568307
(Iteration 21241 / 39200) loss: 1.219291
(Iteration 21251 / 39200) loss: 1.561386
(Iteration 21261 / 39200) loss: 1.450461
(Iteration 21271 / 39200) loss: 1.334558

(Iteration 21281 / 39200) loss: 1.393613
(Iteration 21291 / 39200) loss: 1.434699
(Iteration 21301 / 39200) loss: 1.879764
(Iteration 21311 / 39200) loss: 1.365351
(Iteration 21321 / 39200) loss: 1.624305
(Iteration 21331 / 39200) loss: 1.398739
(Iteration 21341 / 39200) loss: 1.242328
(Iteration 21351 / 39200) loss: 1.625553
(Iteration 21361 / 39200) loss: 1.305255
(Iteration 21371 / 39200) loss: 1.573573
(Iteration 21381 / 39200) loss: 1.927704
(Iteration 21391 / 39200) loss: 1.471754
(Iteration 21401 / 39200) loss: 2.111659
(Iteration 21411 / 39200) loss: 1.329026
(Iteration 21421 / 39200) loss: 1.642917
(Iteration 21431 / 39200) loss: 1.426115
(Iteration 21441 / 39200) loss: 1.350945
(Iteration 21451 / 39200) loss: 2.226262
(Iteration 21461 / 39200) loss: 1.503932
(Iteration 21471 / 39200) loss: 1.600930
(Iteration 21481 / 39200) loss: 1.456168
(Iteration 21491 / 39200) loss: 1.501484
(Iteration 21501 / 39200) loss: 1.482807
(Iteration 21511 / 39200) loss: 1.478032
(Iteration 21521 / 39200) loss: 1.387645
(Iteration 21531 / 39200) loss: 1.841560
(Iteration 21541 / 39200) loss: 1.653346
(Iteration 21551 / 39200) loss: 1.436353
(Epoch 11 / 20) train acc: 0.545000; val_acc: 0.516000
(Iteration 21561 / 39200) loss: 1.535334
(Iteration 21571 / 39200) loss: 1.940137
(Iteration 21581 / 39200) loss: 1.348034
(Iteration 21591 / 39200) loss: 1.755858
(Iteration 21601 / 39200) loss: 2.142599
(Iteration 21611 / 39200) loss: 1.507096
(Iteration 21621 / 39200) loss: 1.791821
(Iteration 21631 / 39200) loss: 1.894716
(Iteration 21641 / 39200) loss: 1.688589
(Iteration 21651 / 39200) loss: 1.668809
(Iteration 21661 / 39200) loss: 1.338923
(Iteration 21671 / 39200) loss: 1.508538
(Iteration 21681 / 39200) loss: 1.644859
(Iteration 21691 / 39200) loss: 1.328554
(Iteration 21701 / 39200) loss: 1.516712
(Iteration 21711 / 39200) loss: 1.435357
(Iteration 21721 / 39200) loss: 1.549944
(Iteration 21731 / 39200) loss: 1.743108
(Iteration 21741 / 39200) loss: 1.456388

(Iteration 21751 / 39200) loss: 1.518190
(Iteration 21761 / 39200) loss: 2.102370
(Iteration 21771 / 39200) loss: 1.306573
(Iteration 21781 / 39200) loss: 1.494648
(Iteration 21791 / 39200) loss: 1.608199
(Iteration 21801 / 39200) loss: 1.378177
(Iteration 21811 / 39200) loss: 1.180913
(Iteration 21821 / 39200) loss: 1.626862
(Iteration 21831 / 39200) loss: 1.753811
(Iteration 21841 / 39200) loss: 1.528297
(Iteration 21851 / 39200) loss: 1.427169
(Iteration 21861 / 39200) loss: 1.488185
(Iteration 21871 / 39200) loss: 1.457577
(Iteration 21881 / 39200) loss: 1.892434
(Iteration 21891 / 39200) loss: 1.538396
(Iteration 21901 / 39200) loss: 1.887723
(Iteration 21911 / 39200) loss: 1.645507
(Iteration 21921 / 39200) loss: 1.348295
(Iteration 21931 / 39200) loss: 1.444718
(Iteration 21941 / 39200) loss: 1.516955
(Iteration 21951 / 39200) loss: 1.348502
(Iteration 21961 / 39200) loss: 1.176078
(Iteration 21971 / 39200) loss: 1.634612
(Iteration 21981 / 39200) loss: 1.661177
(Iteration 21991 / 39200) loss: 1.541343
(Iteration 22001 / 39200) loss: 1.299803
(Iteration 22011 / 39200) loss: 1.560512
(Iteration 22021 / 39200) loss: 1.346625
(Iteration 22031 / 39200) loss: 1.554755
(Iteration 22041 / 39200) loss: 1.485702
(Iteration 22051 / 39200) loss: 1.446910
(Iteration 22061 / 39200) loss: 1.272501
(Iteration 22071 / 39200) loss: 1.386016
(Iteration 22081 / 39200) loss: 1.408809
(Iteration 22091 / 39200) loss: 1.389765
(Iteration 22101 / 39200) loss: 1.411746
(Iteration 22111 / 39200) loss: 1.305351
(Iteration 22121 / 39200) loss: 1.614717
(Iteration 22131 / 39200) loss: 1.703661
(Iteration 22141 / 39200) loss: 1.309501
(Iteration 22151 / 39200) loss: 1.558972
(Iteration 22161 / 39200) loss: 1.643925
(Iteration 22171 / 39200) loss: 1.516092
(Iteration 22181 / 39200) loss: 1.757945
(Iteration 22191 / 39200) loss: 1.553022
(Iteration 22201 / 39200) loss: 1.358941
(Iteration 22211 / 39200) loss: 1.491203
(Iteration 22221 / 39200) loss: 1.206650

(Iteration 22231 / 39200) loss: 1.487478
(Iteration 22241 / 39200) loss: 1.392970
(Iteration 22251 / 39200) loss: 1.395620
(Iteration 22261 / 39200) loss: 1.460707
(Iteration 22271 / 39200) loss: 1.538462
(Iteration 22281 / 39200) loss: 1.985459
(Iteration 22291 / 39200) loss: 1.643655
(Iteration 22301 / 39200) loss: 1.781364
(Iteration 22311 / 39200) loss: 1.169366
(Iteration 22321 / 39200) loss: 1.504047
(Iteration 22331 / 39200) loss: 1.306282
(Iteration 22341 / 39200) loss: 1.265600
(Iteration 22351 / 39200) loss: 1.380711
(Iteration 22361 / 39200) loss: 1.378323
(Iteration 22371 / 39200) loss: 1.491145
(Iteration 22381 / 39200) loss: 1.572820
(Iteration 22391 / 39200) loss: 2.113147
(Iteration 22401 / 39200) loss: 1.527309
(Iteration 22411 / 39200) loss: 1.556745
(Iteration 22421 / 39200) loss: 1.651961
(Iteration 22431 / 39200) loss: 1.500293
(Iteration 22441 / 39200) loss: 1.608704
(Iteration 22451 / 39200) loss: 1.323790
(Iteration 22461 / 39200) loss: 1.257436
(Iteration 22471 / 39200) loss: 1.819583
(Iteration 22481 / 39200) loss: 1.708677
(Iteration 22491 / 39200) loss: 1.409468
(Iteration 22501 / 39200) loss: 1.489128
(Iteration 22511 / 39200) loss: 1.801360
(Iteration 22521 / 39200) loss: 1.170823
(Iteration 22531 / 39200) loss: 1.759028
(Iteration 22541 / 39200) loss: 1.759901
(Iteration 22551 / 39200) loss: 1.727712
(Iteration 22561 / 39200) loss: 1.310291
(Iteration 22571 / 39200) loss: 1.610494
(Iteration 22581 / 39200) loss: 1.565192
(Iteration 22591 / 39200) loss: 1.702501
(Iteration 22601 / 39200) loss: 1.742852
(Iteration 22611 / 39200) loss: 1.458354
(Iteration 22621 / 39200) loss: 1.507306
(Iteration 22631 / 39200) loss: 1.474576
(Iteration 22641 / 39200) loss: 1.688248
(Iteration 22651 / 39200) loss: 1.456258
(Iteration 22661 / 39200) loss: 1.353905
(Iteration 22671 / 39200) loss: 1.305327
(Iteration 22681 / 39200) loss: 1.762269
(Iteration 22691 / 39200) loss: 1.495832
(Iteration 22701 / 39200) loss: 1.636472

(Iteration 22711 / 39200) loss: 1.510183
(Iteration 22721 / 39200) loss: 1.660748
(Iteration 22731 / 39200) loss: 1.184725
(Iteration 22741 / 39200) loss: 1.316874
(Iteration 22751 / 39200) loss: 1.397319
(Iteration 22761 / 39200) loss: 1.547996
(Iteration 22771 / 39200) loss: 1.488539
(Iteration 22781 / 39200) loss: 1.710688
(Iteration 22791 / 39200) loss: 1.361395
(Iteration 22801 / 39200) loss: 1.932370
(Iteration 22811 / 39200) loss: 1.794960
(Iteration 22821 / 39200) loss: 1.719068
(Iteration 22831 / 39200) loss: 1.443709
(Iteration 22841 / 39200) loss: 1.889540
(Iteration 22851 / 39200) loss: 1.569339
(Iteration 22861 / 39200) loss: 1.562966
(Iteration 22871 / 39200) loss: 1.542555
(Iteration 22881 / 39200) loss: 1.705687
(Iteration 22891 / 39200) loss: 1.500563
(Iteration 22901 / 39200) loss: 1.173164
(Iteration 22911 / 39200) loss: 1.548040
(Iteration 22921 / 39200) loss: 1.672701
(Iteration 22931 / 39200) loss: 1.318665
(Iteration 22941 / 39200) loss: 1.691955
(Iteration 22951 / 39200) loss: 1.549283
(Iteration 22961 / 39200) loss: 1.147038
(Iteration 22971 / 39200) loss: 1.634105
(Iteration 22981 / 39200) loss: 1.144126
(Iteration 22991 / 39200) loss: 1.476763
(Iteration 23001 / 39200) loss: 1.991784
(Iteration 23011 / 39200) loss: 1.472469
(Iteration 23021 / 39200) loss: 1.682796
(Iteration 23031 / 39200) loss: 1.347092
(Iteration 23041 / 39200) loss: 1.696513
(Iteration 23051 / 39200) loss: 1.315539
(Iteration 23061 / 39200) loss: 1.442367
(Iteration 23071 / 39200) loss: 1.391079
(Iteration 23081 / 39200) loss: 1.611755
(Iteration 23091 / 39200) loss: 1.703958
(Iteration 23101 / 39200) loss: 1.189845
(Iteration 23111 / 39200) loss: 1.678427
(Iteration 23121 / 39200) loss: 1.662401
(Iteration 23131 / 39200) loss: 1.525957
(Iteration 23141 / 39200) loss: 1.229024
(Iteration 23151 / 39200) loss: 1.606420
(Iteration 23161 / 39200) loss: 1.227966
(Iteration 23171 / 39200) loss: 1.492114
(Iteration 23181 / 39200) loss: 1.398159

(Iteration 23191 / 39200) loss: 1.296478
(Iteration 23201 / 39200) loss: 1.214818
(Iteration 23211 / 39200) loss: 1.573833
(Iteration 23221 / 39200) loss: 1.254664
(Iteration 23231 / 39200) loss: 1.462236
(Iteration 23241 / 39200) loss: 2.239626
(Iteration 23251 / 39200) loss: 1.555134
(Iteration 23261 / 39200) loss: 1.828072
(Iteration 23271 / 39200) loss: 1.682724
(Iteration 23281 / 39200) loss: 1.668014
(Iteration 23291 / 39200) loss: 1.828501
(Iteration 23301 / 39200) loss: 1.473742
(Iteration 23311 / 39200) loss: 1.491084
(Iteration 23321 / 39200) loss: 1.220634
(Iteration 23331 / 39200) loss: 1.092280
(Iteration 23341 / 39200) loss: 1.243004
(Iteration 23351 / 39200) loss: 1.446696
(Iteration 23361 / 39200) loss: 1.627038
(Iteration 23371 / 39200) loss: 1.220714
(Iteration 23381 / 39200) loss: 1.450115
(Iteration 23391 / 39200) loss: 1.296242
(Iteration 23401 / 39200) loss: 1.195985
(Iteration 23411 / 39200) loss: 1.857864
(Iteration 23421 / 39200) loss: 1.488979
(Iteration 23431 / 39200) loss: 1.376664
(Iteration 23441 / 39200) loss: 2.051287
(Iteration 23451 / 39200) loss: 1.430545
(Iteration 23461 / 39200) loss: 1.599640
(Iteration 23471 / 39200) loss: 1.777580
(Iteration 23481 / 39200) loss: 1.249350
(Iteration 23491 / 39200) loss: 1.535381
(Iteration 23501 / 39200) loss: 1.627685
(Iteration 23511 / 39200) loss: 1.532474
(Epoch 12 / 20) train acc: 0.547000; val_acc: 0.509000
(Iteration 23521 / 39200) loss: 1.410659
(Iteration 23531 / 39200) loss: 1.347927
(Iteration 23541 / 39200) loss: 1.652605
(Iteration 23551 / 39200) loss: 2.512988
(Iteration 23561 / 39200) loss: 1.727561
(Iteration 23571 / 39200) loss: 1.534085
(Iteration 23581 / 39200) loss: 1.505006
(Iteration 23591 / 39200) loss: 1.298794
(Iteration 23601 / 39200) loss: 1.442556
(Iteration 23611 / 39200) loss: 2.001456
(Iteration 23621 / 39200) loss: 1.362596
(Iteration 23631 / 39200) loss: 1.774504
(Iteration 23641 / 39200) loss: 1.454345
(Iteration 23651 / 39200) loss: 1.900099

(Iteration 23661 / 39200) loss: 1.533490
(Iteration 23671 / 39200) loss: 1.392258
(Iteration 23681 / 39200) loss: 1.483466
(Iteration 23691 / 39200) loss: 1.441427
(Iteration 23701 / 39200) loss: 1.833859
(Iteration 23711 / 39200) loss: 1.405947
(Iteration 23721 / 39200) loss: 1.383985
(Iteration 23731 / 39200) loss: 1.438380
(Iteration 23741 / 39200) loss: 1.207604
(Iteration 23751 / 39200) loss: 1.942365
(Iteration 23761 / 39200) loss: 1.436211
(Iteration 23771 / 39200) loss: 1.497542
(Iteration 23781 / 39200) loss: 1.454919
(Iteration 23791 / 39200) loss: 1.371139
(Iteration 23801 / 39200) loss: 1.860833
(Iteration 23811 / 39200) loss: 1.682926
(Iteration 23821 / 39200) loss: 1.577874
(Iteration 23831 / 39200) loss: 1.660106
(Iteration 23841 / 39200) loss: 1.568764
(Iteration 23851 / 39200) loss: 1.519080
(Iteration 23861 / 39200) loss: 1.606112
(Iteration 23871 / 39200) loss: 1.277404
(Iteration 23881 / 39200) loss: 1.509552
(Iteration 23891 / 39200) loss: 1.392291
(Iteration 23901 / 39200) loss: 1.509098
(Iteration 23911 / 39200) loss: 1.282110
(Iteration 23921 / 39200) loss: 1.614049
(Iteration 23931 / 39200) loss: 1.422225
(Iteration 23941 / 39200) loss: 1.528636
(Iteration 23951 / 39200) loss: 1.253396
(Iteration 23961 / 39200) loss: 1.528139
(Iteration 23971 / 39200) loss: 1.223252
(Iteration 23981 / 39200) loss: 1.920921
(Iteration 23991 / 39200) loss: 1.455401
(Iteration 24001 / 39200) loss: 1.047605
(Iteration 24011 / 39200) loss: 1.197015
(Iteration 24021 / 39200) loss: 1.453537
(Iteration 24031 / 39200) loss: 1.212388
(Iteration 24041 / 39200) loss: 1.552691
(Iteration 24051 / 39200) loss: 1.392837
(Iteration 24061 / 39200) loss: 1.569212
(Iteration 24071 / 39200) loss: 2.043093
(Iteration 24081 / 39200) loss: 1.498966
(Iteration 24091 / 39200) loss: 1.250482
(Iteration 24101 / 39200) loss: 1.101261
(Iteration 24111 / 39200) loss: 1.455692
(Iteration 24121 / 39200) loss: 1.849820
(Iteration 24131 / 39200) loss: 1.500138

(Iteration 24141 / 39200) loss: 1.687079
(Iteration 24151 / 39200) loss: 1.522398
(Iteration 24161 / 39200) loss: 1.433526
(Iteration 24171 / 39200) loss: 1.321351
(Iteration 24181 / 39200) loss: 1.186205
(Iteration 24191 / 39200) loss: 1.499583
(Iteration 24201 / 39200) loss: 1.381436
(Iteration 24211 / 39200) loss: 1.593688
(Iteration 24221 / 39200) loss: 1.362628
(Iteration 24231 / 39200) loss: 1.474728
(Iteration 24241 / 39200) loss: 1.474026
(Iteration 24251 / 39200) loss: 1.416931
(Iteration 24261 / 39200) loss: 1.463212
(Iteration 24271 / 39200) loss: 1.668960
(Iteration 24281 / 39200) loss: 1.739446
(Iteration 24291 / 39200) loss: 1.761783
(Iteration 24301 / 39200) loss: 1.491245
(Iteration 24311 / 39200) loss: 1.139563
(Iteration 24321 / 39200) loss: 1.443000
(Iteration 24331 / 39200) loss: 1.329481
(Iteration 24341 / 39200) loss: 1.379990
(Iteration 24351 / 39200) loss: 1.399485
(Iteration 24361 / 39200) loss: 1.516162
(Iteration 24371 / 39200) loss: 1.547904
(Iteration 24381 / 39200) loss: 1.794669
(Iteration 24391 / 39200) loss: 1.650053
(Iteration 24401 / 39200) loss: 1.369636
(Iteration 24411 / 39200) loss: 1.383176
(Iteration 24421 / 39200) loss: 1.342219
(Iteration 24431 / 39200) loss: 1.333941
(Iteration 24441 / 39200) loss: 1.404062
(Iteration 24451 / 39200) loss: 1.693343
(Iteration 24461 / 39200) loss: 1.595245
(Iteration 24471 / 39200) loss: 1.349163
(Iteration 24481 / 39200) loss: 1.411603
(Iteration 24491 / 39200) loss: 1.353747
(Iteration 24501 / 39200) loss: 2.120713
(Iteration 24511 / 39200) loss: 1.844473
(Iteration 24521 / 39200) loss: 1.621286
(Iteration 24531 / 39200) loss: 1.451409
(Iteration 24541 / 39200) loss: 1.206524
(Iteration 24551 / 39200) loss: 1.228852
(Iteration 24561 / 39200) loss: 1.804163
(Iteration 24571 / 39200) loss: 1.442378
(Iteration 24581 / 39200) loss: 1.496996
(Iteration 24591 / 39200) loss: 1.799333
(Iteration 24601 / 39200) loss: 1.916420
(Iteration 24611 / 39200) loss: 1.649450

(Iteration 24621 / 39200) loss: 1.198747
(Iteration 24631 / 39200) loss: 1.381858
(Iteration 24641 / 39200) loss: 1.614836
(Iteration 24651 / 39200) loss: 1.899414
(Iteration 24661 / 39200) loss: 1.190219
(Iteration 24671 / 39200) loss: 1.482296
(Iteration 24681 / 39200) loss: 1.455110
(Iteration 24691 / 39200) loss: 1.472897
(Iteration 24701 / 39200) loss: 1.716042
(Iteration 24711 / 39200) loss: 1.661357
(Iteration 24721 / 39200) loss: 1.576950
(Iteration 24731 / 39200) loss: 1.808815
(Iteration 24741 / 39200) loss: 1.465967
(Iteration 24751 / 39200) loss: 1.487041
(Iteration 24761 / 39200) loss: 1.235357
(Iteration 24771 / 39200) loss: 1.223084
(Iteration 24781 / 39200) loss: 1.415495
(Iteration 24791 / 39200) loss: 1.679011
(Iteration 24801 / 39200) loss: 1.927051
(Iteration 24811 / 39200) loss: 1.441018
(Iteration 24821 / 39200) loss: 1.584297
(Iteration 24831 / 39200) loss: 1.767073
(Iteration 24841 / 39200) loss: 1.836498
(Iteration 24851 / 39200) loss: 1.718742
(Iteration 24861 / 39200) loss: 1.443756
(Iteration 24871 / 39200) loss: 1.850465
(Iteration 24881 / 39200) loss: 1.713848
(Iteration 24891 / 39200) loss: 1.517548
(Iteration 24901 / 39200) loss: 1.434105
(Iteration 24911 / 39200) loss: 1.412806
(Iteration 24921 / 39200) loss: 1.532855
(Iteration 24931 / 39200) loss: 1.548372
(Iteration 24941 / 39200) loss: 1.590559
(Iteration 24951 / 39200) loss: 1.450992
(Iteration 24961 / 39200) loss: 1.496564
(Iteration 24971 / 39200) loss: 1.456577
(Iteration 24981 / 39200) loss: 1.264500
(Iteration 24991 / 39200) loss: 1.419014
(Iteration 25001 / 39200) loss: 1.652201
(Iteration 25011 / 39200) loss: 1.578090
(Iteration 25021 / 39200) loss: 1.309337
(Iteration 25031 / 39200) loss: 1.535367
(Iteration 25041 / 39200) loss: 1.387907
(Iteration 25051 / 39200) loss: 1.947412
(Iteration 25061 / 39200) loss: 2.033970
(Iteration 25071 / 39200) loss: 1.237943
(Iteration 25081 / 39200) loss: 1.987530
(Iteration 25091 / 39200) loss: 1.289671

(Iteration 25101 / 39200) loss: 1.319507
(Iteration 25111 / 39200) loss: 1.513481
(Iteration 25121 / 39200) loss: 1.367210
(Iteration 25131 / 39200) loss: 1.563137
(Iteration 25141 / 39200) loss: 1.039364
(Iteration 25151 / 39200) loss: 1.612993
(Iteration 25161 / 39200) loss: 1.559587
(Iteration 25171 / 39200) loss: 1.651659
(Iteration 25181 / 39200) loss: 1.592001
(Iteration 25191 / 39200) loss: 1.544336
(Iteration 25201 / 39200) loss: 1.162560
(Iteration 25211 / 39200) loss: 1.664846
(Iteration 25221 / 39200) loss: 1.277265
(Iteration 25231 / 39200) loss: 1.587645
(Iteration 25241 / 39200) loss: 1.023380
(Iteration 25251 / 39200) loss: 1.590599
(Iteration 25261 / 39200) loss: 1.444141
(Iteration 25271 / 39200) loss: 1.531596
(Iteration 25281 / 39200) loss: 1.174760
(Iteration 25291 / 39200) loss: 1.657694
(Iteration 25301 / 39200) loss: 1.469004
(Iteration 25311 / 39200) loss: 1.589766
(Iteration 25321 / 39200) loss: 1.561212
(Iteration 25331 / 39200) loss: 1.885596
(Iteration 25341 / 39200) loss: 1.452573
(Iteration 25351 / 39200) loss: 1.781005
(Iteration 25361 / 39200) loss: 1.296379
(Iteration 25371 / 39200) loss: 1.051599
(Iteration 25381 / 39200) loss: 1.726961
(Iteration 25391 / 39200) loss: 1.583554
(Iteration 25401 / 39200) loss: 1.263438
(Iteration 25411 / 39200) loss: 1.453004
(Iteration 25421 / 39200) loss: 1.567915
(Iteration 25431 / 39200) loss: 1.220285
(Iteration 25441 / 39200) loss: 1.046290
(Iteration 25451 / 39200) loss: 1.688638
(Iteration 25461 / 39200) loss: 1.649132
(Iteration 25471 / 39200) loss: 1.829452
(Epoch 13 / 20) train acc: 0.572000; val_acc: 0.530000
(Iteration 25481 / 39200) loss: 1.197091
(Iteration 25491 / 39200) loss: 1.424953
(Iteration 25501 / 39200) loss: 1.236606
(Iteration 25511 / 39200) loss: 1.416913
(Iteration 25521 / 39200) loss: 1.293524
(Iteration 25531 / 39200) loss: 1.279714
(Iteration 25541 / 39200) loss: 1.863343
(Iteration 25551 / 39200) loss: 1.829845
(Iteration 25561 / 39200) loss: 1.520255

(Iteration 25571 / 39200) loss: 1.859505
(Iteration 25581 / 39200) loss: 1.646465
(Iteration 25591 / 39200) loss: 1.670932
(Iteration 25601 / 39200) loss: 1.425007
(Iteration 25611 / 39200) loss: 1.485816
(Iteration 25621 / 39200) loss: 1.150346
(Iteration 25631 / 39200) loss: 1.638980
(Iteration 25641 / 39200) loss: 1.421952
(Iteration 25651 / 39200) loss: 1.672463
(Iteration 25661 / 39200) loss: 1.525811
(Iteration 25671 / 39200) loss: 1.503744
(Iteration 25681 / 39200) loss: 1.435230
(Iteration 25691 / 39200) loss: 1.654629
(Iteration 25701 / 39200) loss: 1.805253
(Iteration 25711 / 39200) loss: 1.441451
(Iteration 25721 / 39200) loss: 1.397569
(Iteration 25731 / 39200) loss: 1.522929
(Iteration 25741 / 39200) loss: 1.370733
(Iteration 25751 / 39200) loss: 1.764411
(Iteration 25761 / 39200) loss: 1.414773
(Iteration 25771 / 39200) loss: 1.436296
(Iteration 25781 / 39200) loss: 1.529899
(Iteration 25791 / 39200) loss: 1.476682
(Iteration 25801 / 39200) loss: 1.475600
(Iteration 25811 / 39200) loss: 1.624988
(Iteration 25821 / 39200) loss: 1.519931
(Iteration 25831 / 39200) loss: 1.729722
(Iteration 25841 / 39200) loss: 1.569018
(Iteration 25851 / 39200) loss: 1.733125
(Iteration 25861 / 39200) loss: 1.446401
(Iteration 25871 / 39200) loss: 1.523287
(Iteration 25881 / 39200) loss: 1.865549
(Iteration 25891 / 39200) loss: 1.692653
(Iteration 25901 / 39200) loss: 1.557162
(Iteration 25911 / 39200) loss: 1.897986
(Iteration 25921 / 39200) loss: 1.551813
(Iteration 25931 / 39200) loss: 1.361599
(Iteration 25941 / 39200) loss: 1.814668
(Iteration 25951 / 39200) loss: 1.764485
(Iteration 25961 / 39200) loss: 1.557898
(Iteration 25971 / 39200) loss: 1.523607
(Iteration 25981 / 39200) loss: 1.699879
(Iteration 25991 / 39200) loss: 1.567217
(Iteration 26001 / 39200) loss: 1.364156
(Iteration 26011 / 39200) loss: 1.408063
(Iteration 26021 / 39200) loss: 1.470141
(Iteration 26031 / 39200) loss: 1.505170
(Iteration 26041 / 39200) loss: 1.178544

(Iteration 26051 / 39200) loss: 2.358776
(Iteration 26061 / 39200) loss: 1.593551
(Iteration 26071 / 39200) loss: 1.699731
(Iteration 26081 / 39200) loss: 1.521698
(Iteration 26091 / 39200) loss: 1.632884
(Iteration 26101 / 39200) loss: 1.584966
(Iteration 26111 / 39200) loss: 1.504634
(Iteration 26121 / 39200) loss: 1.582234
(Iteration 26131 / 39200) loss: 1.743774
(Iteration 26141 / 39200) loss: 1.827503
(Iteration 26151 / 39200) loss: 1.302761
(Iteration 26161 / 39200) loss: 1.418841
(Iteration 26171 / 39200) loss: 1.504464
(Iteration 26181 / 39200) loss: 1.691675
(Iteration 26191 / 39200) loss: 1.506276
(Iteration 26201 / 39200) loss: 1.278993
(Iteration 26211 / 39200) loss: 1.351278
(Iteration 26221 / 39200) loss: 1.522329
(Iteration 26231 / 39200) loss: 1.122578
(Iteration 26241 / 39200) loss: 1.678286
(Iteration 26251 / 39200) loss: 1.634451
(Iteration 26261 / 39200) loss: 1.517596
(Iteration 26271 / 39200) loss: 1.533939
(Iteration 26281 / 39200) loss: 1.512183
(Iteration 26291 / 39200) loss: 1.262146
(Iteration 26301 / 39200) loss: 1.374498
(Iteration 26311 / 39200) loss: 1.348824
(Iteration 26321 / 39200) loss: 1.239459
(Iteration 26331 / 39200) loss: 1.276365
(Iteration 26341 / 39200) loss: 1.134486
(Iteration 26351 / 39200) loss: 1.642016
(Iteration 26361 / 39200) loss: 1.220596
(Iteration 26371 / 39200) loss: 1.528475
(Iteration 26381 / 39200) loss: 1.466348
(Iteration 26391 / 39200) loss: 1.539636
(Iteration 26401 / 39200) loss: 1.560686
(Iteration 26411 / 39200) loss: 1.502373
(Iteration 26421 / 39200) loss: 1.360227
(Iteration 26431 / 39200) loss: 1.678814
(Iteration 26441 / 39200) loss: 1.490409
(Iteration 26451 / 39200) loss: 1.595988
(Iteration 26461 / 39200) loss: 1.291834
(Iteration 26471 / 39200) loss: 1.430564
(Iteration 26481 / 39200) loss: 1.745519
(Iteration 26491 / 39200) loss: 1.620221
(Iteration 26501 / 39200) loss: 1.468131
(Iteration 26511 / 39200) loss: 1.867405
(Iteration 26521 / 39200) loss: 1.480509

(Iteration 26531 / 39200) loss: 2.027567
(Iteration 26541 / 39200) loss: 1.490966
(Iteration 26551 / 39200) loss: 1.591350
(Iteration 26561 / 39200) loss: 1.267930
(Iteration 26571 / 39200) loss: 1.755249
(Iteration 26581 / 39200) loss: 1.440710
(Iteration 26591 / 39200) loss: 1.380332
(Iteration 26601 / 39200) loss: 1.686442
(Iteration 26611 / 39200) loss: 1.734091
(Iteration 26621 / 39200) loss: 1.286439
(Iteration 26631 / 39200) loss: 1.088915
(Iteration 26641 / 39200) loss: 1.660003
(Iteration 26651 / 39200) loss: 1.356691
(Iteration 26661 / 39200) loss: 1.341156
(Iteration 26671 / 39200) loss: 1.404832
(Iteration 26681 / 39200) loss: 1.537461
(Iteration 26691 / 39200) loss: 1.938144
(Iteration 26701 / 39200) loss: 1.750690
(Iteration 26711 / 39200) loss: 1.499811
(Iteration 26721 / 39200) loss: 1.311774
(Iteration 26731 / 39200) loss: 1.166028
(Iteration 26741 / 39200) loss: 1.453705
(Iteration 26751 / 39200) loss: 1.562524
(Iteration 26761 / 39200) loss: 1.541215
(Iteration 26771 / 39200) loss: 1.858789
(Iteration 26781 / 39200) loss: 1.599620
(Iteration 26791 / 39200) loss: 1.139443
(Iteration 26801 / 39200) loss: 1.822305
(Iteration 26811 / 39200) loss: 1.405444
(Iteration 26821 / 39200) loss: 1.796878
(Iteration 26831 / 39200) loss: 1.281021
(Iteration 26841 / 39200) loss: 1.837889
(Iteration 26851 / 39200) loss: 1.394791
(Iteration 26861 / 39200) loss: 1.794752
(Iteration 26871 / 39200) loss: 1.433225
(Iteration 26881 / 39200) loss: 1.463300
(Iteration 26891 / 39200) loss: 1.557533
(Iteration 26901 / 39200) loss: 1.579098
(Iteration 26911 / 39200) loss: 1.744444
(Iteration 26921 / 39200) loss: 1.138836
(Iteration 26931 / 39200) loss: 1.829136
(Iteration 26941 / 39200) loss: 1.505723
(Iteration 26951 / 39200) loss: 1.466787
(Iteration 26961 / 39200) loss: 1.821336
(Iteration 26971 / 39200) loss: 1.553333
(Iteration 26981 / 39200) loss: 1.383078
(Iteration 26991 / 39200) loss: 1.342803
(Iteration 27001 / 39200) loss: 1.287091

(Iteration 27011 / 39200) loss: 1.274431
(Iteration 27021 / 39200) loss: 1.772569
(Iteration 27031 / 39200) loss: 1.381278
(Iteration 27041 / 39200) loss: 1.135832
(Iteration 27051 / 39200) loss: 1.444821
(Iteration 27061 / 39200) loss: 1.534972
(Iteration 27071 / 39200) loss: 1.560125
(Iteration 27081 / 39200) loss: 1.943726
(Iteration 27091 / 39200) loss: 0.983558
(Iteration 27101 / 39200) loss: 1.476742
(Iteration 27111 / 39200) loss: 1.299507
(Iteration 27121 / 39200) loss: 1.286513
(Iteration 27131 / 39200) loss: 1.422908
(Iteration 27141 / 39200) loss: 1.201503
(Iteration 27151 / 39200) loss: 1.730353
(Iteration 27161 / 39200) loss: 1.224223
(Iteration 27171 / 39200) loss: 1.299116
(Iteration 27181 / 39200) loss: 1.889553
(Iteration 27191 / 39200) loss: 2.028244
(Iteration 27201 / 39200) loss: 1.621056
(Iteration 27211 / 39200) loss: 1.571035
(Iteration 27221 / 39200) loss: 1.505834
(Iteration 27231 / 39200) loss: 1.439234
(Iteration 27241 / 39200) loss: 1.910609
(Iteration 27251 / 39200) loss: 1.339595
(Iteration 27261 / 39200) loss: 1.414765
(Iteration 27271 / 39200) loss: 1.305136
(Iteration 27281 / 39200) loss: 1.849813
(Iteration 27291 / 39200) loss: 1.512419
(Iteration 27301 / 39200) loss: 1.799417
(Iteration 27311 / 39200) loss: 1.219044
(Iteration 27321 / 39200) loss: 1.264347
(Iteration 27331 / 39200) loss: 1.729646
(Iteration 27341 / 39200) loss: 1.509329
(Iteration 27351 / 39200) loss: 1.614286
(Iteration 27361 / 39200) loss: 1.595964
(Iteration 27371 / 39200) loss: 1.702592
(Iteration 27381 / 39200) loss: 1.575864
(Iteration 27391 / 39200) loss: 1.812082
(Iteration 27401 / 39200) loss: 1.434449
(Iteration 27411 / 39200) loss: 1.485981
(Iteration 27421 / 39200) loss: 1.150440
(Iteration 27431 / 39200) loss: 1.426126
(Epoch 14 / 20) train acc: 0.556000; val_acc: 0.522000
(Iteration 27441 / 39200) loss: 1.266749
(Iteration 27451 / 39200) loss: 1.559063
(Iteration 27461 / 39200) loss: 1.574636
(Iteration 27471 / 39200) loss: 1.212213

(Iteration 27481 / 39200) loss: 1.436369
(Iteration 27491 / 39200) loss: 1.297501
(Iteration 27501 / 39200) loss: 1.308874
(Iteration 27511 / 39200) loss: 1.801772
(Iteration 27521 / 39200) loss: 1.380879
(Iteration 27531 / 39200) loss: 1.420804
(Iteration 27541 / 39200) loss: 1.459713
(Iteration 27551 / 39200) loss: 1.381060
(Iteration 27561 / 39200) loss: 1.290442
(Iteration 27571 / 39200) loss: 1.023018
(Iteration 27581 / 39200) loss: 1.614320
(Iteration 27591 / 39200) loss: 1.476796
(Iteration 27601 / 39200) loss: 1.177691
(Iteration 27611 / 39200) loss: 1.571152
(Iteration 27621 / 39200) loss: 1.681668
(Iteration 27631 / 39200) loss: 1.644836
(Iteration 27641 / 39200) loss: 1.335048
(Iteration 27651 / 39200) loss: 2.131965
(Iteration 27661 / 39200) loss: 1.604633
(Iteration 27671 / 39200) loss: 1.784306
(Iteration 27681 / 39200) loss: 1.464870
(Iteration 27691 / 39200) loss: 1.532734
(Iteration 27701 / 39200) loss: 1.785885
(Iteration 27711 / 39200) loss: 1.495928
(Iteration 27721 / 39200) loss: 1.349085
(Iteration 27731 / 39200) loss: 1.372334
(Iteration 27741 / 39200) loss: 1.890304
(Iteration 27751 / 39200) loss: 1.468675
(Iteration 27761 / 39200) loss: 1.364239
(Iteration 27771 / 39200) loss: 1.057299
(Iteration 27781 / 39200) loss: 1.866240
(Iteration 27791 / 39200) loss: 1.623765
(Iteration 27801 / 39200) loss: 1.360073
(Iteration 27811 / 39200) loss: 1.707205
(Iteration 27821 / 39200) loss: 1.528733
(Iteration 27831 / 39200) loss: 1.564664
(Iteration 27841 / 39200) loss: 1.457600
(Iteration 27851 / 39200) loss: 1.349802
(Iteration 27861 / 39200) loss: 1.783356
(Iteration 27871 / 39200) loss: 1.372331
(Iteration 27881 / 39200) loss: 1.751635
(Iteration 27891 / 39200) loss: 1.811583
(Iteration 27901 / 39200) loss: 1.824280
(Iteration 27911 / 39200) loss: 1.460984
(Iteration 27921 / 39200) loss: 1.240310
(Iteration 27931 / 39200) loss: 1.328316
(Iteration 27941 / 39200) loss: 1.536730
(Iteration 27951 / 39200) loss: 1.525337

(Iteration 27961 / 39200) loss: 1.458420
(Iteration 27971 / 39200) loss: 1.878571
(Iteration 27981 / 39200) loss: 1.947404
(Iteration 27991 / 39200) loss: 1.670412
(Iteration 28001 / 39200) loss: 1.350075
(Iteration 28011 / 39200) loss: 1.503732
(Iteration 28021 / 39200) loss: 1.370843
(Iteration 28031 / 39200) loss: 1.740914
(Iteration 28041 / 39200) loss: 1.479418
(Iteration 28051 / 39200) loss: 1.390713
(Iteration 28061 / 39200) loss: 1.547030
(Iteration 28071 / 39200) loss: 1.345246
(Iteration 28081 / 39200) loss: 1.649125
(Iteration 28091 / 39200) loss: 1.860760
(Iteration 28101 / 39200) loss: 1.714797
(Iteration 28111 / 39200) loss: 1.490193
(Iteration 28121 / 39200) loss: 1.443641
(Iteration 28131 / 39200) loss: 1.460293
(Iteration 28141 / 39200) loss: 1.322320
(Iteration 28151 / 39200) loss: 1.231991
(Iteration 28161 / 39200) loss: 1.236980
(Iteration 28171 / 39200) loss: 1.376669
(Iteration 28181 / 39200) loss: 1.646669
(Iteration 28191 / 39200) loss: 1.447613
(Iteration 28201 / 39200) loss: 1.134663
(Iteration 28211 / 39200) loss: 1.489494
(Iteration 28221 / 39200) loss: 1.234550
(Iteration 28231 / 39200) loss: 1.430256
(Iteration 28241 / 39200) loss: 1.442331
(Iteration 28251 / 39200) loss: 1.129656
(Iteration 28261 / 39200) loss: 1.391187
(Iteration 28271 / 39200) loss: 1.284829
(Iteration 28281 / 39200) loss: 1.132095
(Iteration 28291 / 39200) loss: 1.508589
(Iteration 28301 / 39200) loss: 1.510716
(Iteration 28311 / 39200) loss: 1.318473
(Iteration 28321 / 39200) loss: 1.481956
(Iteration 28331 / 39200) loss: 1.659371
(Iteration 28341 / 39200) loss: 1.462943
(Iteration 28351 / 39200) loss: 1.525139
(Iteration 28361 / 39200) loss: 1.262049
(Iteration 28371 / 39200) loss: 1.322709
(Iteration 28381 / 39200) loss: 1.517879
(Iteration 28391 / 39200) loss: 1.860025
(Iteration 28401 / 39200) loss: 1.643813
(Iteration 28411 / 39200) loss: 1.405814
(Iteration 28421 / 39200) loss: 1.173931
(Iteration 28431 / 39200) loss: 1.244121

(Iteration 28441 / 39200) loss: 1.700390
(Iteration 28451 / 39200) loss: 1.298672
(Iteration 28461 / 39200) loss: 1.677836
(Iteration 28471 / 39200) loss: 1.891529
(Iteration 28481 / 39200) loss: 1.756305
(Iteration 28491 / 39200) loss: 1.178963
(Iteration 28501 / 39200) loss: 1.474677
(Iteration 28511 / 39200) loss: 1.242661
(Iteration 28521 / 39200) loss: 1.495195
(Iteration 28531 / 39200) loss: 1.897070
(Iteration 28541 / 39200) loss: 1.599634
(Iteration 28551 / 39200) loss: 1.564692
(Iteration 28561 / 39200) loss: 1.523637
(Iteration 28571 / 39200) loss: 1.530383
(Iteration 28581 / 39200) loss: 1.493668
(Iteration 28591 / 39200) loss: 1.325323
(Iteration 28601 / 39200) loss: 1.758782
(Iteration 28611 / 39200) loss: 1.104072
(Iteration 28621 / 39200) loss: 1.504769
(Iteration 28631 / 39200) loss: 1.495663
(Iteration 28641 / 39200) loss: 1.282349
(Iteration 28651 / 39200) loss: 1.647636
(Iteration 28661 / 39200) loss: 1.170777
(Iteration 28671 / 39200) loss: 1.800459
(Iteration 28681 / 39200) loss: 1.315249
(Iteration 28691 / 39200) loss: 1.476379
(Iteration 28701 / 39200) loss: 1.512776
(Iteration 28711 / 39200) loss: 1.518600
(Iteration 28721 / 39200) loss: 1.634982
(Iteration 28731 / 39200) loss: 1.470188
(Iteration 28741 / 39200) loss: 1.563360
(Iteration 28751 / 39200) loss: 1.292828
(Iteration 28761 / 39200) loss: 1.430406
(Iteration 28771 / 39200) loss: 1.555523
(Iteration 28781 / 39200) loss: 1.356651
(Iteration 28791 / 39200) loss: 1.246717
(Iteration 28801 / 39200) loss: 1.542214
(Iteration 28811 / 39200) loss: 1.560467
(Iteration 28821 / 39200) loss: 1.622491
(Iteration 28831 / 39200) loss: 1.523617
(Iteration 28841 / 39200) loss: 1.627826
(Iteration 28851 / 39200) loss: 1.566201
(Iteration 28861 / 39200) loss: 1.407430
(Iteration 28871 / 39200) loss: 1.503969
(Iteration 28881 / 39200) loss: 1.273594
(Iteration 28891 / 39200) loss: 1.377193
(Iteration 28901 / 39200) loss: 1.312261
(Iteration 28911 / 39200) loss: 1.405655

(Iteration 28921 / 39200) loss: 1.495232
(Iteration 28931 / 39200) loss: 1.137929
(Iteration 28941 / 39200) loss: 1.205513
(Iteration 28951 / 39200) loss: 1.577806
(Iteration 28961 / 39200) loss: 1.670812
(Iteration 28971 / 39200) loss: 1.466129
(Iteration 28981 / 39200) loss: 1.526108
(Iteration 28991 / 39200) loss: 1.510601
(Iteration 29001 / 39200) loss: 1.381496
(Iteration 29011 / 39200) loss: 1.601095
(Iteration 29021 / 39200) loss: 1.514770
(Iteration 29031 / 39200) loss: 1.318089
(Iteration 29041 / 39200) loss: 1.245094
(Iteration 29051 / 39200) loss: 1.474439
(Iteration 29061 / 39200) loss: 1.358206
(Iteration 29071 / 39200) loss: 1.467399
(Iteration 29081 / 39200) loss: 1.246175
(Iteration 29091 / 39200) loss: 1.730263
(Iteration 29101 / 39200) loss: 1.593464
(Iteration 29111 / 39200) loss: 1.341201
(Iteration 29121 / 39200) loss: 1.653173
(Iteration 29131 / 39200) loss: 1.449470
(Iteration 29141 / 39200) loss: 1.472926
(Iteration 29151 / 39200) loss: 1.508584
(Iteration 29161 / 39200) loss: 1.495418
(Iteration 29171 / 39200) loss: 1.494699
(Iteration 29181 / 39200) loss: 1.270747
(Iteration 29191 / 39200) loss: 1.255458
(Iteration 29201 / 39200) loss: 2.103721
(Iteration 29211 / 39200) loss: 1.067213
(Iteration 29221 / 39200) loss: 1.490816
(Iteration 29231 / 39200) loss: 1.489814
(Iteration 29241 / 39200) loss: 1.276688
(Iteration 29251 / 39200) loss: 1.347488
(Iteration 29261 / 39200) loss: 1.436126
(Iteration 29271 / 39200) loss: 1.593916
(Iteration 29281 / 39200) loss: 1.316581
(Iteration 29291 / 39200) loss: 1.184277
(Iteration 29301 / 39200) loss: 1.263218
(Iteration 29311 / 39200) loss: 1.636438
(Iteration 29321 / 39200) loss: 1.999342
(Iteration 29331 / 39200) loss: 1.449312
(Iteration 29341 / 39200) loss: 1.297031
(Iteration 29351 / 39200) loss: 1.523293
(Iteration 29361 / 39200) loss: 1.295432
(Iteration 29371 / 39200) loss: 1.631727
(Iteration 29381 / 39200) loss: 1.677457
(Iteration 29391 / 39200) loss: 1.633173

(Epoch 15 / 20) train acc: 0.574000; val_acc: 0.524000
(Iteration 29401 / 39200) loss: 1.546757
(Iteration 29411 / 39200) loss: 1.786405
(Iteration 29421 / 39200) loss: 1.594567
(Iteration 29431 / 39200) loss: 1.846268
(Iteration 29441 / 39200) loss: 1.763890
(Iteration 29451 / 39200) loss: 1.583822
(Iteration 29461 / 39200) loss: 1.162943
(Iteration 29471 / 39200) loss: 1.400505
(Iteration 29481 / 39200) loss: 1.570715
(Iteration 29491 / 39200) loss: 1.800532
(Iteration 29501 / 39200) loss: 1.222805
(Iteration 29511 / 39200) loss: 1.420367
(Iteration 29521 / 39200) loss: 1.478265
(Iteration 29531 / 39200) loss: 1.262284
(Iteration 29541 / 39200) loss: 1.704734
(Iteration 29551 / 39200) loss: 1.327892
(Iteration 29561 / 39200) loss: 1.459427
(Iteration 29571 / 39200) loss: 1.334364
(Iteration 29581 / 39200) loss: 1.410748
(Iteration 29591 / 39200) loss: 1.147006
(Iteration 29601 / 39200) loss: 1.274317
(Iteration 29611 / 39200) loss: 1.308932
(Iteration 29621 / 39200) loss: 1.428099
(Iteration 29631 / 39200) loss: 1.204022
(Iteration 29641 / 39200) loss: 1.388783
(Iteration 29651 / 39200) loss: 1.338456
(Iteration 29661 / 39200) loss: 1.279479
(Iteration 29671 / 39200) loss: 1.517315
(Iteration 29681 / 39200) loss: 1.551385
(Iteration 29691 / 39200) loss: 1.363356
(Iteration 29701 / 39200) loss: 1.715607
(Iteration 29711 / 39200) loss: 1.623736
(Iteration 29721 / 39200) loss: 1.773619
(Iteration 29731 / 39200) loss: 1.348211
(Iteration 29741 / 39200) loss: 1.502691
(Iteration 29751 / 39200) loss: 1.578289
(Iteration 29761 / 39200) loss: 1.411314
(Iteration 29771 / 39200) loss: 1.612413
(Iteration 29781 / 39200) loss: 1.323179
(Iteration 29791 / 39200) loss: 1.263041
(Iteration 29801 / 39200) loss: 1.677340
(Iteration 29811 / 39200) loss: 1.701133
(Iteration 29821 / 39200) loss: 1.821447
(Iteration 29831 / 39200) loss: 1.278342
(Iteration 29841 / 39200) loss: 1.200946
(Iteration 29851 / 39200) loss: 1.529018
(Iteration 29861 / 39200) loss: 1.383433

(Iteration 29871 / 39200) loss: 1.022781
(Iteration 29881 / 39200) loss: 1.183205
(Iteration 29891 / 39200) loss: 1.706437
(Iteration 29901 / 39200) loss: 1.552554
(Iteration 29911 / 39200) loss: 1.659274
(Iteration 29921 / 39200) loss: 1.601705
(Iteration 29931 / 39200) loss: 1.658635
(Iteration 29941 / 39200) loss: 1.449188
(Iteration 29951 / 39200) loss: 1.372344
(Iteration 29961 / 39200) loss: 1.434609
(Iteration 29971 / 39200) loss: 1.566099
(Iteration 29981 / 39200) loss: 1.703047
(Iteration 29991 / 39200) loss: 1.436056
(Iteration 30001 / 39200) loss: 1.486905
(Iteration 30011 / 39200) loss: 1.595557
(Iteration 30021 / 39200) loss: 1.214547
(Iteration 30031 / 39200) loss: 1.909187
(Iteration 30041 / 39200) loss: 2.049502
(Iteration 30051 / 39200) loss: 1.632918
(Iteration 30061 / 39200) loss: 1.401594
(Iteration 30071 / 39200) loss: 1.745516
(Iteration 30081 / 39200) loss: 1.281705
(Iteration 30091 / 39200) loss: 1.842893
(Iteration 30101 / 39200) loss: 1.575758
(Iteration 30111 / 39200) loss: 1.328798
(Iteration 30121 / 39200) loss: 1.488243
(Iteration 30131 / 39200) loss: 1.183917
(Iteration 30141 / 39200) loss: 1.373103
(Iteration 30151 / 39200) loss: 1.738865
(Iteration 30161 / 39200) loss: 1.381301
(Iteration 30171 / 39200) loss: 1.460406
(Iteration 30181 / 39200) loss: 1.854486
(Iteration 30191 / 39200) loss: 1.686948
(Iteration 30201 / 39200) loss: 2.014771
(Iteration 30211 / 39200) loss: 1.763459
(Iteration 30221 / 39200) loss: 1.715172
(Iteration 30231 / 39200) loss: 1.239215
(Iteration 30241 / 39200) loss: 2.076905
(Iteration 30251 / 39200) loss: 1.868200
(Iteration 30261 / 39200) loss: 1.510641
(Iteration 30271 / 39200) loss: 1.561361
(Iteration 30281 / 39200) loss: 1.324891
(Iteration 30291 / 39200) loss: 1.399377
(Iteration 30301 / 39200) loss: 0.872587
(Iteration 30311 / 39200) loss: 1.780890
(Iteration 30321 / 39200) loss: 1.413105
(Iteration 30331 / 39200) loss: 1.734826
(Iteration 30341 / 39200) loss: 1.751663

(Iteration 30351 / 39200) loss: 1.843853
(Iteration 30361 / 39200) loss: 1.706768
(Iteration 30371 / 39200) loss: 1.414964
(Iteration 30381 / 39200) loss: 1.593874
(Iteration 30391 / 39200) loss: 1.418910
(Iteration 30401 / 39200) loss: 1.491812
(Iteration 30411 / 39200) loss: 1.668570
(Iteration 30421 / 39200) loss: 1.205539
(Iteration 30431 / 39200) loss: 1.368573
(Iteration 30441 / 39200) loss: 1.266344
(Iteration 30451 / 39200) loss: 1.491944
(Iteration 30461 / 39200) loss: 1.310217
(Iteration 30471 / 39200) loss: 1.547444
(Iteration 30481 / 39200) loss: 1.621990
(Iteration 30491 / 39200) loss: 1.701111
(Iteration 30501 / 39200) loss: 1.743812
(Iteration 30511 / 39200) loss: 1.662787
(Iteration 30521 / 39200) loss: 1.377962
(Iteration 30531 / 39200) loss: 1.919943
(Iteration 30541 / 39200) loss: 1.657771
(Iteration 30551 / 39200) loss: 1.451214
(Iteration 30561 / 39200) loss: 1.362706
(Iteration 30571 / 39200) loss: 1.207801
(Iteration 30581 / 39200) loss: 1.267702
(Iteration 30591 / 39200) loss: 1.616556
(Iteration 30601 / 39200) loss: 1.489886
(Iteration 30611 / 39200) loss: 1.440471
(Iteration 30621 / 39200) loss: 1.570045
(Iteration 30631 / 39200) loss: 1.246636
(Iteration 30641 / 39200) loss: 1.385814
(Iteration 30651 / 39200) loss: 1.373237
(Iteration 30661 / 39200) loss: 1.452420
(Iteration 30671 / 39200) loss: 1.393526
(Iteration 30681 / 39200) loss: 1.296476
(Iteration 30691 / 39200) loss: 1.651847
(Iteration 30701 / 39200) loss: 1.950228
(Iteration 30711 / 39200) loss: 1.374631
(Iteration 30721 / 39200) loss: 1.476256
(Iteration 30731 / 39200) loss: 1.227404
(Iteration 30741 / 39200) loss: 1.283096
(Iteration 30751 / 39200) loss: 1.210637
(Iteration 30761 / 39200) loss: 1.150668
(Iteration 30771 / 39200) loss: 1.640143
(Iteration 30781 / 39200) loss: 1.618322
(Iteration 30791 / 39200) loss: 1.818884
(Iteration 30801 / 39200) loss: 1.227796
(Iteration 30811 / 39200) loss: 1.481007
(Iteration 30821 / 39200) loss: 1.597330

(Iteration 30831 / 39200) loss: 1.611646
(Iteration 30841 / 39200) loss: 1.397218
(Iteration 30851 / 39200) loss: 1.246744
(Iteration 30861 / 39200) loss: 1.632841
(Iteration 30871 / 39200) loss: 1.333905
(Iteration 30881 / 39200) loss: 1.292280
(Iteration 30891 / 39200) loss: 1.601792
(Iteration 30901 / 39200) loss: 1.347088
(Iteration 30911 / 39200) loss: 1.680870
(Iteration 30921 / 39200) loss: 2.041366
(Iteration 30931 / 39200) loss: 1.273813
(Iteration 30941 / 39200) loss: 1.666906
(Iteration 30951 / 39200) loss: 2.042970
(Iteration 30961 / 39200) loss: 1.226509
(Iteration 30971 / 39200) loss: 1.588686
(Iteration 30981 / 39200) loss: 1.623629
(Iteration 30991 / 39200) loss: 1.340801
(Iteration 31001 / 39200) loss: 1.406686
(Iteration 31011 / 39200) loss: 1.695256
(Iteration 31021 / 39200) loss: 1.370786
(Iteration 31031 / 39200) loss: 1.594507
(Iteration 31041 / 39200) loss: 1.466764
(Iteration 31051 / 39200) loss: 1.828490
(Iteration 31061 / 39200) loss: 1.388359
(Iteration 31071 / 39200) loss: 1.927503
(Iteration 31081 / 39200) loss: 1.454793
(Iteration 31091 / 39200) loss: 1.521660
(Iteration 31101 / 39200) loss: 1.233242
(Iteration 31111 / 39200) loss: 1.541669
(Iteration 31121 / 39200) loss: 1.470097
(Iteration 31131 / 39200) loss: 1.143273
(Iteration 31141 / 39200) loss: 1.459180
(Iteration 31151 / 39200) loss: 1.381283
(Iteration 31161 / 39200) loss: 1.269943
(Iteration 31171 / 39200) loss: 1.471362
(Iteration 31181 / 39200) loss: 1.836711
(Iteration 31191 / 39200) loss: 1.558912
(Iteration 31201 / 39200) loss: 1.255562
(Iteration 31211 / 39200) loss: 1.524600
(Iteration 31221 / 39200) loss: 1.535687
(Iteration 31231 / 39200) loss: 1.441069
(Iteration 31241 / 39200) loss: 1.306739
(Iteration 31251 / 39200) loss: 1.666018
(Iteration 31261 / 39200) loss: 1.777827
(Iteration 31271 / 39200) loss: 1.455880
(Iteration 31281 / 39200) loss: 1.531709
(Iteration 31291 / 39200) loss: 1.335243
(Iteration 31301 / 39200) loss: 1.587205

(Iteration 31311 / 39200) loss: 1.786425
(Iteration 31321 / 39200) loss: 1.146778
(Iteration 31331 / 39200) loss: 1.313754
(Iteration 31341 / 39200) loss: 1.688229
(Iteration 31351 / 39200) loss: 1.445318
(Epoch 16 / 20) train acc: 0.563000; val_acc: 0.520000
(Iteration 31361 / 39200) loss: 2.005012
(Iteration 31371 / 39200) loss: 1.205207
(Iteration 31381 / 39200) loss: 1.219278
(Iteration 31391 / 39200) loss: 1.322122
(Iteration 31401 / 39200) loss: 1.517939
(Iteration 31411 / 39200) loss: 1.169771
(Iteration 31421 / 39200) loss: 1.333162
(Iteration 31431 / 39200) loss: 1.686565
(Iteration 31441 / 39200) loss: 1.498961
(Iteration 31451 / 39200) loss: 1.639275
(Iteration 31461 / 39200) loss: 1.539650
(Iteration 31471 / 39200) loss: 1.634760
(Iteration 31481 / 39200) loss: 1.488139
(Iteration 31491 / 39200) loss: 1.735644
(Iteration 31501 / 39200) loss: 1.809351
(Iteration 31511 / 39200) loss: 1.465511
(Iteration 31521 / 39200) loss: 1.126153
(Iteration 31531 / 39200) loss: 1.197782
(Iteration 31541 / 39200) loss: 1.339954
(Iteration 31551 / 39200) loss: 1.769131
(Iteration 31561 / 39200) loss: 1.637339
(Iteration 31571 / 39200) loss: 1.561393
(Iteration 31581 / 39200) loss: 1.803344
(Iteration 31591 / 39200) loss: 1.424665
(Iteration 31601 / 39200) loss: 1.669951
(Iteration 31611 / 39200) loss: 1.588639
(Iteration 31621 / 39200) loss: 1.855098
(Iteration 31631 / 39200) loss: 1.385488
(Iteration 31641 / 39200) loss: 1.658619
(Iteration 31651 / 39200) loss: 1.634325
(Iteration 31661 / 39200) loss: 1.377648
(Iteration 31671 / 39200) loss: 1.438829
(Iteration 31681 / 39200) loss: 1.546155
(Iteration 31691 / 39200) loss: 1.426322
(Iteration 31701 / 39200) loss: 1.422573
(Iteration 31711 / 39200) loss: 1.573399
(Iteration 31721 / 39200) loss: 1.358589
(Iteration 31731 / 39200) loss: 1.648638
(Iteration 31741 / 39200) loss: 1.593847
(Iteration 31751 / 39200) loss: 1.488414
(Iteration 31761 / 39200) loss: 1.304064
(Iteration 31771 / 39200) loss: 1.004704

(Iteration 31781 / 39200) loss: 1.713648
(Iteration 31791 / 39200) loss: 1.590851
(Iteration 31801 / 39200) loss: 1.393469
(Iteration 31811 / 39200) loss: 1.960972
(Iteration 31821 / 39200) loss: 1.388839
(Iteration 31831 / 39200) loss: 1.200391
(Iteration 31841 / 39200) loss: 1.661726
(Iteration 31851 / 39200) loss: 1.420715
(Iteration 31861 / 39200) loss: 1.222495
(Iteration 31871 / 39200) loss: 1.479410
(Iteration 31881 / 39200) loss: 1.692054
(Iteration 31891 / 39200) loss: 1.050747
(Iteration 31901 / 39200) loss: 1.426650
(Iteration 31911 / 39200) loss: 1.893247
(Iteration 31921 / 39200) loss: 1.544588
(Iteration 31931 / 39200) loss: 1.472498
(Iteration 31941 / 39200) loss: 1.112475
(Iteration 31951 / 39200) loss: 1.597311
(Iteration 31961 / 39200) loss: 1.375156
(Iteration 31971 / 39200) loss: 1.501977
(Iteration 31981 / 39200) loss: 1.328630
(Iteration 31991 / 39200) loss: 1.893232
(Iteration 32001 / 39200) loss: 1.170897
(Iteration 32011 / 39200) loss: 1.720246
(Iteration 32021 / 39200) loss: 1.676467
(Iteration 32031 / 39200) loss: 1.446413
(Iteration 32041 / 39200) loss: 1.687426
(Iteration 32051 / 39200) loss: 1.317492
(Iteration 32061 / 39200) loss: 1.697281
(Iteration 32071 / 39200) loss: 1.139649
(Iteration 32081 / 39200) loss: 1.084533
(Iteration 32091 / 39200) loss: 1.410885
(Iteration 32101 / 39200) loss: 1.283498
(Iteration 32111 / 39200) loss: 1.365154
(Iteration 32121 / 39200) loss: 1.761155
(Iteration 32131 / 39200) loss: 1.571044
(Iteration 32141 / 39200) loss: 1.237535
(Iteration 32151 / 39200) loss: 1.420898
(Iteration 32161 / 39200) loss: 1.270921
(Iteration 32171 / 39200) loss: 1.356701
(Iteration 32181 / 39200) loss: 1.358310
(Iteration 32191 / 39200) loss: 1.496936
(Iteration 32201 / 39200) loss: 1.371506
(Iteration 32211 / 39200) loss: 1.379263
(Iteration 32221 / 39200) loss: 2.240460
(Iteration 32231 / 39200) loss: 1.433318
(Iteration 32241 / 39200) loss: 1.575406
(Iteration 32251 / 39200) loss: 1.363612

(Iteration 32261 / 39200) loss: 1.373107
(Iteration 32271 / 39200) loss: 1.647309
(Iteration 32281 / 39200) loss: 1.363658
(Iteration 32291 / 39200) loss: 1.164537
(Iteration 32301 / 39200) loss: 1.670432
(Iteration 32311 / 39200) loss: 1.269471
(Iteration 32321 / 39200) loss: 1.348840
(Iteration 32331 / 39200) loss: 1.517729
(Iteration 32341 / 39200) loss: 1.697061
(Iteration 32351 / 39200) loss: 1.449645
(Iteration 32361 / 39200) loss: 1.734627
(Iteration 32371 / 39200) loss: 1.282495
(Iteration 32381 / 39200) loss: 1.359763
(Iteration 32391 / 39200) loss: 1.395594
(Iteration 32401 / 39200) loss: 1.530395
(Iteration 32411 / 39200) loss: 1.459172
(Iteration 32421 / 39200) loss: 1.700784
(Iteration 32431 / 39200) loss: 1.585866
(Iteration 32441 / 39200) loss: 1.682492
(Iteration 32451 / 39200) loss: 1.686224
(Iteration 32461 / 39200) loss: 1.153193
(Iteration 32471 / 39200) loss: 1.683198
(Iteration 32481 / 39200) loss: 1.513664
(Iteration 32491 / 39200) loss: 1.345496
(Iteration 32501 / 39200) loss: 1.714447
(Iteration 32511 / 39200) loss: 1.402969
(Iteration 32521 / 39200) loss: 1.222679
(Iteration 32531 / 39200) loss: 1.426725
(Iteration 32541 / 39200) loss: 1.740786
(Iteration 32551 / 39200) loss: 1.254909
(Iteration 32561 / 39200) loss: 1.348305
(Iteration 32571 / 39200) loss: 0.986327
(Iteration 32581 / 39200) loss: 1.565101
(Iteration 32591 / 39200) loss: 1.529695
(Iteration 32601 / 39200) loss: 1.567616
(Iteration 32611 / 39200) loss: 1.340716
(Iteration 32621 / 39200) loss: 1.448102
(Iteration 32631 / 39200) loss: 1.002445
(Iteration 32641 / 39200) loss: 1.848446
(Iteration 32651 / 39200) loss: 1.402972
(Iteration 32661 / 39200) loss: 1.350827
(Iteration 32671 / 39200) loss: 1.464492
(Iteration 32681 / 39200) loss: 1.215618
(Iteration 32691 / 39200) loss: 1.400986
(Iteration 32701 / 39200) loss: 1.422533
(Iteration 32711 / 39200) loss: 1.171442
(Iteration 32721 / 39200) loss: 1.089664
(Iteration 32731 / 39200) loss: 1.277258

(Iteration 32741 / 39200) loss: 1.512572
(Iteration 32751 / 39200) loss: 1.462176
(Iteration 32761 / 39200) loss: 1.530651
(Iteration 32771 / 39200) loss: 1.316290
(Iteration 32781 / 39200) loss: 1.733894
(Iteration 32791 / 39200) loss: 1.427454
(Iteration 32801 / 39200) loss: 1.496487
(Iteration 32811 / 39200) loss: 1.313144
(Iteration 32821 / 39200) loss: 1.437808
(Iteration 32831 / 39200) loss: 1.455924
(Iteration 32841 / 39200) loss: 1.503745
(Iteration 32851 / 39200) loss: 1.709529
(Iteration 32861 / 39200) loss: 1.249499
(Iteration 32871 / 39200) loss: 1.420899
(Iteration 32881 / 39200) loss: 1.654695
(Iteration 32891 / 39200) loss: 1.148781
(Iteration 32901 / 39200) loss: 1.507378
(Iteration 32911 / 39200) loss: 1.087512
(Iteration 32921 / 39200) loss: 1.468836
(Iteration 32931 / 39200) loss: 1.642185
(Iteration 32941 / 39200) loss: 1.210940
(Iteration 32951 / 39200) loss: 1.222661
(Iteration 32961 / 39200) loss: 1.771289
(Iteration 32971 / 39200) loss: 1.642706
(Iteration 32981 / 39200) loss: 1.932216
(Iteration 32991 / 39200) loss: 1.929835
(Iteration 33001 / 39200) loss: 1.552069
(Iteration 33011 / 39200) loss: 1.447487
(Iteration 33021 / 39200) loss: 1.298372
(Iteration 33031 / 39200) loss: 1.541735
(Iteration 33041 / 39200) loss: 1.530521
(Iteration 33051 / 39200) loss: 1.481803
(Iteration 33061 / 39200) loss: 1.309861
(Iteration 33071 / 39200) loss: 1.489023
(Iteration 33081 / 39200) loss: 1.437209
(Iteration 33091 / 39200) loss: 1.412539
(Iteration 33101 / 39200) loss: 1.064908
(Iteration 33111 / 39200) loss: 1.673339
(Iteration 33121 / 39200) loss: 1.302983
(Iteration 33131 / 39200) loss: 1.628457
(Iteration 33141 / 39200) loss: 1.540304
(Iteration 33151 / 39200) loss: 1.418159
(Iteration 33161 / 39200) loss: 1.401337
(Iteration 33171 / 39200) loss: 1.252875
(Iteration 33181 / 39200) loss: 1.558645
(Iteration 33191 / 39200) loss: 1.359014
(Iteration 33201 / 39200) loss: 1.275626
(Iteration 33211 / 39200) loss: 1.914641

(Iteration 33221 / 39200) loss: 1.937083
(Iteration 33231 / 39200) loss: 1.719467
(Iteration 33241 / 39200) loss: 1.397081
(Iteration 33251 / 39200) loss: 1.341785
(Iteration 33261 / 39200) loss: 1.255612
(Iteration 33271 / 39200) loss: 1.489341
(Iteration 33281 / 39200) loss: 1.199251
(Iteration 33291 / 39200) loss: 1.791460
(Iteration 33301 / 39200) loss: 1.376073
(Iteration 33311 / 39200) loss: 1.355475
(Epoch 17 / 20) train acc: 0.560000; val_acc: 0.516000
(Iteration 33321 / 39200) loss: 1.532901
(Iteration 33331 / 39200) loss: 1.742337
(Iteration 33341 / 39200) loss: 1.660613
(Iteration 33351 / 39200) loss: 1.348418
(Iteration 33361 / 39200) loss: 1.062019
(Iteration 33371 / 39200) loss: 1.280452
(Iteration 33381 / 39200) loss: 1.195841
(Iteration 33391 / 39200) loss: 1.828738
(Iteration 33401 / 39200) loss: 1.440287
(Iteration 33411 / 39200) loss: 1.436490
(Iteration 33421 / 39200) loss: 1.416552
(Iteration 33431 / 39200) loss: 1.326330
(Iteration 33441 / 39200) loss: 1.410820
(Iteration 33451 / 39200) loss: 1.516230
(Iteration 33461 / 39200) loss: 1.394603
(Iteration 33471 / 39200) loss: 1.559063
(Iteration 33481 / 39200) loss: 1.318445
(Iteration 33491 / 39200) loss: 1.618508
(Iteration 33501 / 39200) loss: 1.362775
(Iteration 33511 / 39200) loss: 1.573659
(Iteration 33521 / 39200) loss: 1.244672
(Iteration 33531 / 39200) loss: 1.712149
(Iteration 33541 / 39200) loss: 1.494436
(Iteration 33551 / 39200) loss: 1.625109
(Iteration 33561 / 39200) loss: 1.248434
(Iteration 33571 / 39200) loss: 1.592869
(Iteration 33581 / 39200) loss: 1.464968
(Iteration 33591 / 39200) loss: 1.339050
(Iteration 33601 / 39200) loss: 1.550670
(Iteration 33611 / 39200) loss: 1.436316
(Iteration 33621 / 39200) loss: 1.448658
(Iteration 33631 / 39200) loss: 1.484629
(Iteration 33641 / 39200) loss: 1.276002
(Iteration 33651 / 39200) loss: 1.410045
(Iteration 33661 / 39200) loss: 1.297940
(Iteration 33671 / 39200) loss: 1.793793
(Iteration 33681 / 39200) loss: 1.371789

(Iteration 33691 / 39200) loss: 1.771089
(Iteration 33701 / 39200) loss: 1.471145
(Iteration 33711 / 39200) loss: 1.635285
(Iteration 33721 / 39200) loss: 1.304484
(Iteration 33731 / 39200) loss: 1.476199
(Iteration 33741 / 39200) loss: 1.374057
(Iteration 33751 / 39200) loss: 2.019584
(Iteration 33761 / 39200) loss: 1.326300
(Iteration 33771 / 39200) loss: 1.729180
(Iteration 33781 / 39200) loss: 1.381393
(Iteration 33791 / 39200) loss: 1.187397
(Iteration 33801 / 39200) loss: 1.615805
(Iteration 33811 / 39200) loss: 1.464697
(Iteration 33821 / 39200) loss: 1.589917
(Iteration 33831 / 39200) loss: 1.243238
(Iteration 33841 / 39200) loss: 1.443253
(Iteration 33851 / 39200) loss: 1.597615
(Iteration 33861 / 39200) loss: 1.865872
(Iteration 33871 / 39200) loss: 1.054708
(Iteration 33881 / 39200) loss: 1.407614
(Iteration 33891 / 39200) loss: 1.485970
(Iteration 33901 / 39200) loss: 1.814723
(Iteration 33911 / 39200) loss: 1.466931
(Iteration 33921 / 39200) loss: 1.117546
(Iteration 33931 / 39200) loss: 1.259843
(Iteration 33941 / 39200) loss: 1.101741
(Iteration 33951 / 39200) loss: 1.091883
(Iteration 33961 / 39200) loss: 1.171391
(Iteration 33971 / 39200) loss: 1.437130
(Iteration 33981 / 39200) loss: 1.306619
(Iteration 33991 / 39200) loss: 1.437748
(Iteration 34001 / 39200) loss: 1.332691
(Iteration 34011 / 39200) loss: 1.030798
(Iteration 34021 / 39200) loss: 1.792936
(Iteration 34031 / 39200) loss: 1.295550
(Iteration 34041 / 39200) loss: 1.279049
(Iteration 34051 / 39200) loss: 1.091624
(Iteration 34061 / 39200) loss: 1.487577
(Iteration 34071 / 39200) loss: 1.663397
(Iteration 34081 / 39200) loss: 1.385209
(Iteration 34091 / 39200) loss: 1.409843
(Iteration 34101 / 39200) loss: 1.533377
(Iteration 34111 / 39200) loss: 1.917766
(Iteration 34121 / 39200) loss: 1.347266
(Iteration 34131 / 39200) loss: 1.633355
(Iteration 34141 / 39200) loss: 1.731542
(Iteration 34151 / 39200) loss: 1.511440
(Iteration 34161 / 39200) loss: 1.739477

(Iteration 34171 / 39200) loss: 1.511375
(Iteration 34181 / 39200) loss: 1.691601
(Iteration 34191 / 39200) loss: 1.373844
(Iteration 34201 / 39200) loss: 1.339852
(Iteration 34211 / 39200) loss: 1.493445
(Iteration 34221 / 39200) loss: 1.400819
(Iteration 34231 / 39200) loss: 1.553492
(Iteration 34241 / 39200) loss: 1.061606
(Iteration 34251 / 39200) loss: 1.519244
(Iteration 34261 / 39200) loss: 1.630587
(Iteration 34271 / 39200) loss: 1.618102
(Iteration 34281 / 39200) loss: 1.504275
(Iteration 34291 / 39200) loss: 1.450160
(Iteration 34301 / 39200) loss: 1.646132
(Iteration 34311 / 39200) loss: 1.471975
(Iteration 34321 / 39200) loss: 1.316219
(Iteration 34331 / 39200) loss: 1.749615
(Iteration 34341 / 39200) loss: 1.397352
(Iteration 34351 / 39200) loss: 1.487242
(Iteration 34361 / 39200) loss: 1.864720
(Iteration 34371 / 39200) loss: 1.412102
(Iteration 34381 / 39200) loss: 1.278797
(Iteration 34391 / 39200) loss: 1.548705
(Iteration 34401 / 39200) loss: 1.549803
(Iteration 34411 / 39200) loss: 1.433387
(Iteration 34421 / 39200) loss: 1.480755
(Iteration 34431 / 39200) loss: 1.281181
(Iteration 34441 / 39200) loss: 1.609380
(Iteration 34451 / 39200) loss: 1.581556
(Iteration 34461 / 39200) loss: 1.506082
(Iteration 34471 / 39200) loss: 1.174725
(Iteration 34481 / 39200) loss: 1.413514
(Iteration 34491 / 39200) loss: 1.788948
(Iteration 34501 / 39200) loss: 1.721620
(Iteration 34511 / 39200) loss: 1.533967
(Iteration 34521 / 39200) loss: 1.696514
(Iteration 34531 / 39200) loss: 1.010021
(Iteration 34541 / 39200) loss: 1.335254
(Iteration 34551 / 39200) loss: 1.255321
(Iteration 34561 / 39200) loss: 1.674091
(Iteration 34571 / 39200) loss: 1.733518
(Iteration 34581 / 39200) loss: 1.619406
(Iteration 34591 / 39200) loss: 1.322908
(Iteration 34601 / 39200) loss: 1.102080
(Iteration 34611 / 39200) loss: 1.242084
(Iteration 34621 / 39200) loss: 1.395013
(Iteration 34631 / 39200) loss: 1.327227
(Iteration 34641 / 39200) loss: 1.529363

(Iteration 34651 / 39200) loss: 1.450194
(Iteration 34661 / 39200) loss: 1.292637
(Iteration 34671 / 39200) loss: 1.243970
(Iteration 34681 / 39200) loss: 1.616310
(Iteration 34691 / 39200) loss: 1.385903
(Iteration 34701 / 39200) loss: 1.920836
(Iteration 34711 / 39200) loss: 1.131154
(Iteration 34721 / 39200) loss: 1.543604
(Iteration 34731 / 39200) loss: 1.654459
(Iteration 34741 / 39200) loss: 1.135622
(Iteration 34751 / 39200) loss: 1.256197
(Iteration 34761 / 39200) loss: 1.223495
(Iteration 34771 / 39200) loss: 1.013420
(Iteration 34781 / 39200) loss: 1.270918
(Iteration 34791 / 39200) loss: 1.294176
(Iteration 34801 / 39200) loss: 1.746580
(Iteration 34811 / 39200) loss: 1.205575
(Iteration 34821 / 39200) loss: 1.173427
(Iteration 34831 / 39200) loss: 1.392999
(Iteration 34841 / 39200) loss: 1.422751
(Iteration 34851 / 39200) loss: 1.466080
(Iteration 34861 / 39200) loss: 1.033829
(Iteration 34871 / 39200) loss: 1.733401
(Iteration 34881 / 39200) loss: 1.459795
(Iteration 34891 / 39200) loss: 1.625162
(Iteration 34901 / 39200) loss: 1.538630
(Iteration 34911 / 39200) loss: 1.583637
(Iteration 34921 / 39200) loss: 1.862605
(Iteration 34931 / 39200) loss: 1.416252
(Iteration 34941 / 39200) loss: 1.674119
(Iteration 34951 / 39200) loss: 1.760979
(Iteration 34961 / 39200) loss: 1.405836
(Iteration 34971 / 39200) loss: 1.193275
(Iteration 34981 / 39200) loss: 1.232212
(Iteration 34991 / 39200) loss: 1.465843
(Iteration 35001 / 39200) loss: 1.680984
(Iteration 35011 / 39200) loss: 1.457094
(Iteration 35021 / 39200) loss: 1.177658
(Iteration 35031 / 39200) loss: 1.716900
(Iteration 35041 / 39200) loss: 1.599481
(Iteration 35051 / 39200) loss: 1.910360
(Iteration 35061 / 39200) loss: 1.508410
(Iteration 35071 / 39200) loss: 1.549076
(Iteration 35081 / 39200) loss: 1.835932
(Iteration 35091 / 39200) loss: 1.258469
(Iteration 35101 / 39200) loss: 1.628974
(Iteration 35111 / 39200) loss: 1.591085
(Iteration 35121 / 39200) loss: 1.528940

(Iteration 35131 / 39200) loss: 1.739042
(Iteration 35141 / 39200) loss: 1.362490
(Iteration 35151 / 39200) loss: 1.541112
(Iteration 35161 / 39200) loss: 1.401557
(Iteration 35171 / 39200) loss: 1.285325
(Iteration 35181 / 39200) loss: 1.386639
(Iteration 35191 / 39200) loss: 1.355439
(Iteration 35201 / 39200) loss: 1.534751
(Iteration 35211 / 39200) loss: 1.734000
(Iteration 35221 / 39200) loss: 1.563294
(Iteration 35231 / 39200) loss: 1.259407
(Iteration 35241 / 39200) loss: 1.682235
(Iteration 35251 / 39200) loss: 1.746615
(Iteration 35261 / 39200) loss: 1.636262
(Iteration 35271 / 39200) loss: 1.138963
(Epoch 18 / 20) train acc: 0.582000; val_acc: 0.520000
(Iteration 35281 / 39200) loss: 1.430728
(Iteration 35291 / 39200) loss: 1.211727
(Iteration 35301 / 39200) loss: 1.236739
(Iteration 35311 / 39200) loss: 1.886860
(Iteration 35321 / 39200) loss: 1.600525
(Iteration 35331 / 39200) loss: 1.359850
(Iteration 35341 / 39200) loss: 1.174059
(Iteration 35351 / 39200) loss: 1.460825
(Iteration 35361 / 39200) loss: 1.505318
(Iteration 35371 / 39200) loss: 1.576222
(Iteration 35381 / 39200) loss: 1.202569
(Iteration 35391 / 39200) loss: 1.409932
(Iteration 35401 / 39200) loss: 1.407142
(Iteration 35411 / 39200) loss: 1.503996
(Iteration 35421 / 39200) loss: 1.096735
(Iteration 35431 / 39200) loss: 1.624044
(Iteration 35441 / 39200) loss: 1.290955
(Iteration 35451 / 39200) loss: 1.649307
(Iteration 35461 / 39200) loss: 1.523102
(Iteration 35471 / 39200) loss: 1.399138
(Iteration 35481 / 39200) loss: 1.491358
(Iteration 35491 / 39200) loss: 1.535238
(Iteration 35501 / 39200) loss: 1.178846
(Iteration 35511 / 39200) loss: 1.632840
(Iteration 35521 / 39200) loss: 1.271799
(Iteration 35531 / 39200) loss: 1.414843
(Iteration 35541 / 39200) loss: 1.617824
(Iteration 35551 / 39200) loss: 1.258963
(Iteration 35561 / 39200) loss: 1.247621
(Iteration 35571 / 39200) loss: 1.683405
(Iteration 35581 / 39200) loss: 1.714939
(Iteration 35591 / 39200) loss: 1.219840

(Iteration 35601 / 39200) loss: 1.760734
(Iteration 35611 / 39200) loss: 1.164312
(Iteration 35621 / 39200) loss: 1.422647
(Iteration 35631 / 39200) loss: 1.442514
(Iteration 35641 / 39200) loss: 1.351175
(Iteration 35651 / 39200) loss: 1.326996
(Iteration 35661 / 39200) loss: 1.405891
(Iteration 35671 / 39200) loss: 1.487802
(Iteration 35681 / 39200) loss: 1.746616
(Iteration 35691 / 39200) loss: 1.529936
(Iteration 35701 / 39200) loss: 1.825820
(Iteration 35711 / 39200) loss: 1.185816
(Iteration 35721 / 39200) loss: 2.044804
(Iteration 35731 / 39200) loss: 1.256340
(Iteration 35741 / 39200) loss: 1.513226
(Iteration 35751 / 39200) loss: 1.487147
(Iteration 35761 / 39200) loss: 1.411830
(Iteration 35771 / 39200) loss: 1.620199
(Iteration 35781 / 39200) loss: 1.568176
(Iteration 35791 / 39200) loss: 1.510014
(Iteration 35801 / 39200) loss: 1.342679
(Iteration 35811 / 39200) loss: 1.276689
(Iteration 35821 / 39200) loss: 1.757302
(Iteration 35831 / 39200) loss: 1.294922
(Iteration 35841 / 39200) loss: 1.727245
(Iteration 35851 / 39200) loss: 1.336514
(Iteration 35861 / 39200) loss: 1.222221
(Iteration 35871 / 39200) loss: 1.472336
(Iteration 35881 / 39200) loss: 1.332866
(Iteration 35891 / 39200) loss: 1.207540
(Iteration 35901 / 39200) loss: 1.654273
(Iteration 35911 / 39200) loss: 1.099243
(Iteration 35921 / 39200) loss: 1.216313
(Iteration 35931 / 39200) loss: 1.769135
(Iteration 35941 / 39200) loss: 1.409540
(Iteration 35951 / 39200) loss: 1.659613
(Iteration 35961 / 39200) loss: 1.453053
(Iteration 35971 / 39200) loss: 1.475940
(Iteration 35981 / 39200) loss: 1.161403
(Iteration 35991 / 39200) loss: 1.262066
(Iteration 36001 / 39200) loss: 1.194512
(Iteration 36011 / 39200) loss: 1.704985
(Iteration 36021 / 39200) loss: 1.504326
(Iteration 36031 / 39200) loss: 1.589410
(Iteration 36041 / 39200) loss: 1.434862
(Iteration 36051 / 39200) loss: 1.429090
(Iteration 36061 / 39200) loss: 1.176772
(Iteration 36071 / 39200) loss: 1.177323

(Iteration 36081 / 39200) loss: 1.223158
(Iteration 36091 / 39200) loss: 1.896892
(Iteration 36101 / 39200) loss: 1.238763
(Iteration 36111 / 39200) loss: 1.296071
(Iteration 36121 / 39200) loss: 1.310035
(Iteration 36131 / 39200) loss: 1.669716
(Iteration 36141 / 39200) loss: 1.333074
(Iteration 36151 / 39200) loss: 1.364965
(Iteration 36161 / 39200) loss: 1.219023
(Iteration 36171 / 39200) loss: 1.101223
(Iteration 36181 / 39200) loss: 1.335447
(Iteration 36191 / 39200) loss: 1.204668
(Iteration 36201 / 39200) loss: 1.274803
(Iteration 36211 / 39200) loss: 1.472091
(Iteration 36221 / 39200) loss: 1.410792
(Iteration 36231 / 39200) loss: 1.476561
(Iteration 36241 / 39200) loss: 1.274095
(Iteration 36251 / 39200) loss: 1.564776
(Iteration 36261 / 39200) loss: 1.396791
(Iteration 36271 / 39200) loss: 1.648864
(Iteration 36281 / 39200) loss: 1.355381
(Iteration 36291 / 39200) loss: 1.551524
(Iteration 36301 / 39200) loss: 1.555685
(Iteration 36311 / 39200) loss: 1.205612
(Iteration 36321 / 39200) loss: 1.457668
(Iteration 36331 / 39200) loss: 1.679608
(Iteration 36341 / 39200) loss: 1.594965
(Iteration 36351 / 39200) loss: 1.588368
(Iteration 36361 / 39200) loss: 1.717314
(Iteration 36371 / 39200) loss: 1.416097
(Iteration 36381 / 39200) loss: 1.679104
(Iteration 36391 / 39200) loss: 1.585757
(Iteration 36401 / 39200) loss: 1.484659
(Iteration 36411 / 39200) loss: 1.154239
(Iteration 36421 / 39200) loss: 1.743115
(Iteration 36431 / 39200) loss: 1.394819
(Iteration 36441 / 39200) loss: 1.652773
(Iteration 36451 / 39200) loss: 1.777370
(Iteration 36461 / 39200) loss: 1.392793
(Iteration 36471 / 39200) loss: 1.589733
(Iteration 36481 / 39200) loss: 1.428589
(Iteration 36491 / 39200) loss: 1.238856
(Iteration 36501 / 39200) loss: 1.194922
(Iteration 36511 / 39200) loss: 1.411891
(Iteration 36521 / 39200) loss: 1.711923
(Iteration 36531 / 39200) loss: 1.382522
(Iteration 36541 / 39200) loss: 1.404011
(Iteration 36551 / 39200) loss: 1.820970

(Iteration 36561 / 39200) loss: 1.617607
(Iteration 36571 / 39200) loss: 1.702655
(Iteration 36581 / 39200) loss: 1.537810
(Iteration 36591 / 39200) loss: 1.570862
(Iteration 36601 / 39200) loss: 1.666117
(Iteration 36611 / 39200) loss: 1.527857
(Iteration 36621 / 39200) loss: 1.434740
(Iteration 36631 / 39200) loss: 1.226074
(Iteration 36641 / 39200) loss: 1.382173
(Iteration 36651 / 39200) loss: 1.265308
(Iteration 36661 / 39200) loss: 1.567755
(Iteration 36671 / 39200) loss: 1.322431
(Iteration 36681 / 39200) loss: 1.178941
(Iteration 36691 / 39200) loss: 1.205421
(Iteration 36701 / 39200) loss: 1.383342
(Iteration 36711 / 39200) loss: 1.458422
(Iteration 36721 / 39200) loss: 1.410297
(Iteration 36731 / 39200) loss: 1.407209
(Iteration 36741 / 39200) loss: 1.782167
(Iteration 36751 / 39200) loss: 1.640653
(Iteration 36761 / 39200) loss: 1.466702
(Iteration 36771 / 39200) loss: 1.382128
(Iteration 36781 / 39200) loss: 1.632887
(Iteration 36791 / 39200) loss: 1.285792
(Iteration 36801 / 39200) loss: 1.670855
(Iteration 36811 / 39200) loss: 1.468052
(Iteration 36821 / 39200) loss: 2.176928
(Iteration 36831 / 39200) loss: 1.704663
(Iteration 36841 / 39200) loss: 1.315971
(Iteration 36851 / 39200) loss: 1.633057
(Iteration 36861 / 39200) loss: 1.464164
(Iteration 36871 / 39200) loss: 1.353796
(Iteration 36881 / 39200) loss: 1.562880
(Iteration 36891 / 39200) loss: 1.517136
(Iteration 36901 / 39200) loss: 1.242193
(Iteration 36911 / 39200) loss: 1.156932
(Iteration 36921 / 39200) loss: 1.285318
(Iteration 36931 / 39200) loss: 1.163005
(Iteration 36941 / 39200) loss: 1.833913
(Iteration 36951 / 39200) loss: 1.090742
(Iteration 36961 / 39200) loss: 1.516514
(Iteration 36971 / 39200) loss: 1.397382
(Iteration 36981 / 39200) loss: 1.448083
(Iteration 36991 / 39200) loss: 1.812439
(Iteration 37001 / 39200) loss: 1.510154
(Iteration 37011 / 39200) loss: 1.740045
(Iteration 37021 / 39200) loss: 1.405643
(Iteration 37031 / 39200) loss: 1.335994

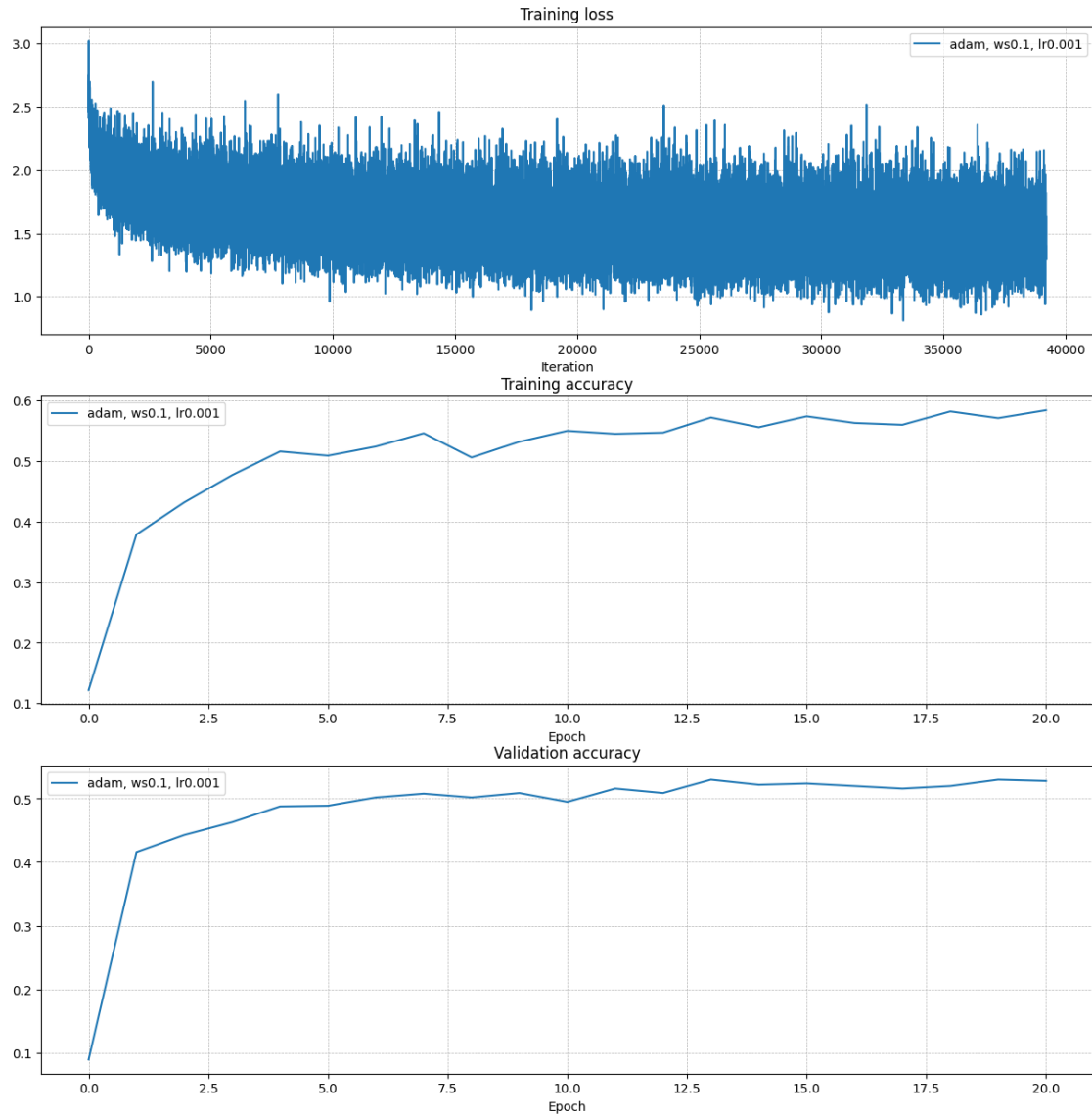
(Iteration 37041 / 39200) loss: 1.465143
(Iteration 37051 / 39200) loss: 1.680878
(Iteration 37061 / 39200) loss: 1.779943
(Iteration 37071 / 39200) loss: 1.893626
(Iteration 37081 / 39200) loss: 1.441388
(Iteration 37091 / 39200) loss: 1.246178
(Iteration 37101 / 39200) loss: 1.380257
(Iteration 37111 / 39200) loss: 1.984723
(Iteration 37121 / 39200) loss: 1.494187
(Iteration 37131 / 39200) loss: 1.350734
(Iteration 37141 / 39200) loss: 1.171030
(Iteration 37151 / 39200) loss: 1.476237
(Iteration 37161 / 39200) loss: 1.697937
(Iteration 37171 / 39200) loss: 1.578442
(Iteration 37181 / 39200) loss: 1.349287
(Iteration 37191 / 39200) loss: 1.398285
(Iteration 37201 / 39200) loss: 1.588068
(Iteration 37211 / 39200) loss: 1.512096
(Iteration 37221 / 39200) loss: 1.284497
(Iteration 37231 / 39200) loss: 1.554575
(Epoch 19 / 20) train acc: 0.571000; val_acc: 0.530000
(Iteration 37241 / 39200) loss: 1.154400
(Iteration 37251 / 39200) loss: 1.860699
(Iteration 37261 / 39200) loss: 1.450402
(Iteration 37271 / 39200) loss: 1.376937
(Iteration 37281 / 39200) loss: 1.185668
(Iteration 37291 / 39200) loss: 1.621389
(Iteration 37301 / 39200) loss: 1.677826
(Iteration 37311 / 39200) loss: 1.642646
(Iteration 37321 / 39200) loss: 1.311285
(Iteration 37331 / 39200) loss: 1.641363
(Iteration 37341 / 39200) loss: 1.613408
(Iteration 37351 / 39200) loss: 1.201288
(Iteration 37361 / 39200) loss: 1.533100
(Iteration 37371 / 39200) loss: 1.519838
(Iteration 37381 / 39200) loss: 1.453364
(Iteration 37391 / 39200) loss: 1.599590
(Iteration 37401 / 39200) loss: 1.692153
(Iteration 37411 / 39200) loss: 1.635598
(Iteration 37421 / 39200) loss: 1.550427
(Iteration 37431 / 39200) loss: 1.453106
(Iteration 37441 / 39200) loss: 1.970423
(Iteration 37451 / 39200) loss: 1.397716
(Iteration 37461 / 39200) loss: 1.313220
(Iteration 37471 / 39200) loss: 1.543272
(Iteration 37481 / 39200) loss: 1.217846
(Iteration 37491 / 39200) loss: 1.124582
(Iteration 37501 / 39200) loss: 1.156268

(Iteration 37511 / 39200) loss: 1.290527
(Iteration 37521 / 39200) loss: 1.567198
(Iteration 37531 / 39200) loss: 1.542421
(Iteration 37541 / 39200) loss: 1.353529
(Iteration 37551 / 39200) loss: 1.323141
(Iteration 37561 / 39200) loss: 1.481449
(Iteration 37571 / 39200) loss: 1.131654
(Iteration 37581 / 39200) loss: 2.009534
(Iteration 37591 / 39200) loss: 1.530775
(Iteration 37601 / 39200) loss: 1.394145
(Iteration 37611 / 39200) loss: 1.437444
(Iteration 37621 / 39200) loss: 1.639552
(Iteration 37631 / 39200) loss: 1.274116
(Iteration 37641 / 39200) loss: 1.414119
(Iteration 37651 / 39200) loss: 1.618537
(Iteration 37661 / 39200) loss: 0.919143
(Iteration 37671 / 39200) loss: 1.558317
(Iteration 37681 / 39200) loss: 1.628864
(Iteration 37691 / 39200) loss: 1.330780
(Iteration 37701 / 39200) loss: 1.479757
(Iteration 37711 / 39200) loss: 1.566704
(Iteration 37721 / 39200) loss: 1.405479
(Iteration 37731 / 39200) loss: 1.412263
(Iteration 37741 / 39200) loss: 1.262120
(Iteration 37751 / 39200) loss: 1.603950
(Iteration 37761 / 39200) loss: 1.457680
(Iteration 37771 / 39200) loss: 1.535984
(Iteration 37781 / 39200) loss: 1.665535
(Iteration 37791 / 39200) loss: 1.374847
(Iteration 37801 / 39200) loss: 1.316746
(Iteration 37811 / 39200) loss: 1.281369
(Iteration 37821 / 39200) loss: 1.293372
(Iteration 37831 / 39200) loss: 1.721140
(Iteration 37841 / 39200) loss: 1.467265
(Iteration 37851 / 39200) loss: 1.482763
(Iteration 37861 / 39200) loss: 1.535446
(Iteration 37871 / 39200) loss: 1.409225
(Iteration 37881 / 39200) loss: 1.519274
(Iteration 37891 / 39200) loss: 1.490690
(Iteration 37901 / 39200) loss: 2.144543
(Iteration 37911 / 39200) loss: 1.229122
(Iteration 37921 / 39200) loss: 1.233906
(Iteration 37931 / 39200) loss: 1.643403
(Iteration 37941 / 39200) loss: 1.914449
(Iteration 37951 / 39200) loss: 1.380295
(Iteration 37961 / 39200) loss: 1.152370
(Iteration 37971 / 39200) loss: 1.217494
(Iteration 37981 / 39200) loss: 1.500909

(Iteration 37991 / 39200) loss: 1.197077
(Iteration 38001 / 39200) loss: 1.131662
(Iteration 38011 / 39200) loss: 1.451550
(Iteration 38021 / 39200) loss: 1.575514
(Iteration 38031 / 39200) loss: 1.681148
(Iteration 38041 / 39200) loss: 1.464197
(Iteration 38051 / 39200) loss: 1.254635
(Iteration 38061 / 39200) loss: 1.380344
(Iteration 38071 / 39200) loss: 2.122780
(Iteration 38081 / 39200) loss: 1.557535
(Iteration 38091 / 39200) loss: 2.080544
(Iteration 38101 / 39200) loss: 1.359138
(Iteration 38111 / 39200) loss: 1.643854
(Iteration 38121 / 39200) loss: 1.567410
(Iteration 38131 / 39200) loss: 1.071182
(Iteration 38141 / 39200) loss: 1.187774
(Iteration 38151 / 39200) loss: 1.554213
(Iteration 38161 / 39200) loss: 1.736208
(Iteration 38171 / 39200) loss: 1.358247
(Iteration 38181 / 39200) loss: 1.216588
(Iteration 38191 / 39200) loss: 1.793886
(Iteration 38201 / 39200) loss: 1.918951
(Iteration 38211 / 39200) loss: 1.142483
(Iteration 38221 / 39200) loss: 1.381186
(Iteration 38231 / 39200) loss: 1.342196
(Iteration 38241 / 39200) loss: 1.488917
(Iteration 38251 / 39200) loss: 1.444014
(Iteration 38261 / 39200) loss: 1.613409
(Iteration 38271 / 39200) loss: 1.555504
(Iteration 38281 / 39200) loss: 1.614638
(Iteration 38291 / 39200) loss: 1.325881
(Iteration 38301 / 39200) loss: 1.665375
(Iteration 38311 / 39200) loss: 1.735804
(Iteration 38321 / 39200) loss: 1.425268
(Iteration 38331 / 39200) loss: 1.373054
(Iteration 38341 / 39200) loss: 1.381462
(Iteration 38351 / 39200) loss: 1.754082
(Iteration 38361 / 39200) loss: 1.697477
(Iteration 38371 / 39200) loss: 1.467534
(Iteration 38381 / 39200) loss: 1.261278
(Iteration 38391 / 39200) loss: 1.152937
(Iteration 38401 / 39200) loss: 1.287930
(Iteration 38411 / 39200) loss: 1.540791
(Iteration 38421 / 39200) loss: 1.337848
(Iteration 38431 / 39200) loss: 1.754951
(Iteration 38441 / 39200) loss: 1.478913
(Iteration 38451 / 39200) loss: 1.688721
(Iteration 38461 / 39200) loss: 1.236732

(Iteration 38471 / 39200) loss: 1.350210
(Iteration 38481 / 39200) loss: 1.509688
(Iteration 38491 / 39200) loss: 1.175201
(Iteration 38501 / 39200) loss: 1.431627
(Iteration 38511 / 39200) loss: 1.375466
(Iteration 38521 / 39200) loss: 1.413533
(Iteration 38531 / 39200) loss: 1.114352
(Iteration 38541 / 39200) loss: 1.277136
(Iteration 38551 / 39200) loss: 1.664767
(Iteration 38561 / 39200) loss: 1.408561
(Iteration 38571 / 39200) loss: 1.519237
(Iteration 38581 / 39200) loss: 1.364156
(Iteration 38591 / 39200) loss: 1.642908
(Iteration 38601 / 39200) loss: 1.646205
(Iteration 38611 / 39200) loss: 1.334798
(Iteration 38621 / 39200) loss: 1.541877
(Iteration 38631 / 39200) loss: 1.174213
(Iteration 38641 / 39200) loss: 1.878352
(Iteration 38651 / 39200) loss: 1.422012
(Iteration 38661 / 39200) loss: 1.499422
(Iteration 38671 / 39200) loss: 1.498002
(Iteration 38681 / 39200) loss: 1.706338
(Iteration 38691 / 39200) loss: 1.459487
(Iteration 38701 / 39200) loss: 1.570401
(Iteration 38711 / 39200) loss: 1.403327
(Iteration 38721 / 39200) loss: 1.363846
(Iteration 38731 / 39200) loss: 1.416786
(Iteration 38741 / 39200) loss: 1.309256
(Iteration 38751 / 39200) loss: 1.745585
(Iteration 38761 / 39200) loss: 1.208247
(Iteration 38771 / 39200) loss: 1.450172
(Iteration 38781 / 39200) loss: 1.224391
(Iteration 38791 / 39200) loss: 1.625977
(Iteration 38801 / 39200) loss: 1.272538
(Iteration 38811 / 39200) loss: 1.566337
(Iteration 38821 / 39200) loss: 2.146612
(Iteration 38831 / 39200) loss: 1.641641
(Iteration 38841 / 39200) loss: 1.750059
(Iteration 38851 / 39200) loss: 1.533887
(Iteration 38861 / 39200) loss: 1.356065
(Iteration 38871 / 39200) loss: 1.334848
(Iteration 38881 / 39200) loss: 1.627015
(Iteration 38891 / 39200) loss: 1.473266
(Iteration 38901 / 39200) loss: 1.652339
(Iteration 38911 / 39200) loss: 1.638979
(Iteration 38921 / 39200) loss: 1.585358
(Iteration 38931 / 39200) loss: 1.369694
(Iteration 38941 / 39200) loss: 1.413108

```
(Iteration 38951 / 39200) loss: 1.294431
(Iteration 38961 / 39200) loss: 1.349846
(Iteration 38971 / 39200) loss: 1.358090
(Iteration 38981 / 39200) loss: 1.669236
(Iteration 38991 / 39200) loss: 1.551336
(Iteration 39001 / 39200) loss: 1.244776
(Iteration 39011 / 39200) loss: 1.478716
(Iteration 39021 / 39200) loss: 1.350960
(Iteration 39031 / 39200) loss: 1.280180
(Iteration 39041 / 39200) loss: 1.847195
(Iteration 39051 / 39200) loss: 1.106569
(Iteration 39061 / 39200) loss: 1.410189
(Iteration 39071 / 39200) loss: 1.556252
(Iteration 39081 / 39200) loss: 1.412870
(Iteration 39091 / 39200) loss: 1.088023
(Iteration 39101 / 39200) loss: 1.479721
(Iteration 39111 / 39200) loss: 1.382590
(Iteration 39121 / 39200) loss: 1.311161
(Iteration 39131 / 39200) loss: 1.238633
(Iteration 39141 / 39200) loss: 1.307348
(Iteration 39151 / 39200) loss: 1.633950
(Iteration 39161 / 39200) loss: 1.710490
(Iteration 39171 / 39200) loss: 1.512052
(Iteration 39181 / 39200) loss: 1.861129
(Iteration 39191 / 39200) loss: 1.633866
(Epoch 20 / 20) train acc: 0.584000; val_acc: 0.528000
weight_scale 1.000000e-01 lr 1.000000e-03 rule adam train accuracy: 0.584000
val accuracy: 0.530000
best val accuracy achieved: 0.530000
```



6 Test Your Model!

Run your best model on the validation and test sets. Are you able to outperform the baseline model that has no Batchnorm or Dropout?

```
[21]: y_test_pred = np.argmax(best_model_improved.loss(data['X_test']), axis=1)
y_val_pred = np.argmax(best_model_improved.loss(data['X_val']), axis=1)
print('Validation set accuracy: ', (y_val_pred == data['y_val']).mean())
print('Test set accuracy: ', (y_test_pred == data['y_test']).mean())
```

Validation set accuracy: 0.497

Test set accuracy: 0.52

BatchNormalization

February 16, 2024

```
[ ]: # This mounts your Google Drive to the Colab VM.
from google.colab import drive
drive.mount('/content/drive')

# TODO: Enter the foldername in your Drive where you have saved the unzipped
# assignment folder, e.g. 'cse493g1/assignments/assignment3/'
FOLDERNAME = 'cse493g1/assignments/assignment3/'
assert FOLDERNAME is not None, "[!] Enter the foldername."

# Now that we've mounted your Drive, this ensures that
# the Python interpreter of the Colab VM can load
# python files from within it.
import sys
sys.path.append('/content/drive/My Drive/{}'.format(FOLDERNAME))

# This downloads the COCO dataset to your Drive
# if it doesn't already exist.
%cd /content/drive/My\ Drive/$FOLDERNAME/cse493g1/datasets/
!bash get_datasets.sh
%cd /content/drive/My\ Drive/$FOLDERNAME
```

```
Mounted at /content/drive
/content/drive/My Drive/cse493g1/assignments/assignment3/cse493g1/datasets
/content/drive/My Drive/cse493g1/assignments/assignment3
```

1 Batch Normalization

One way to make deep networks easier to train is to use more sophisticated optimization procedures such as SGD+momentum, RMSProp, or Adam. Another strategy is to change the architecture of the network to make it easier to train. One idea along these lines is batch normalization, proposed by [1] in 2015.

To understand the goal of batch normalization, it is important to first recognize that machine learning methods tend to perform better with input data consisting of uncorrelated features with zero mean and unit variance. When training a neural network, we can preprocess the data before feeding it to the network to explicitly decorrelate its features. This will ensure that the first layer of the network sees data that follows a nice distribution. However, even if we preprocess the input data, the activations at deeper layers of the network will likely no longer be decorrelated and will

no longer have zero mean or unit variance, since they are output from earlier layers in the network. Even worse, during the training process the distribution of features at each layer of the network will shift as the weights of each layer are updated.

The authors of [1] hypothesize that the shifting distribution of features inside deep neural networks may make training deep networks more difficult. To overcome this problem, they propose to insert into the network layers that normalize batches. At training time, such a layer uses a minibatch of data to estimate the mean and standard deviation of each feature. These estimated means and standard deviations are then used to center and normalize the features of the minibatch. A running average of these means and standard deviations is kept during training, and at test time these running averages are used to center and normalize features.

It is possible that this normalization strategy could reduce the representational power of the network, since it may sometimes be optimal for certain layers to have features that are not zero-mean or unit variance. To this end, the batch normalization layer includes learnable shift and scale parameters for each feature dimension.

[1] Sergey Ioffe and Christian Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”, ICML 2015.

```
[ ]: # Setup cell.
import time
import numpy as np
import matplotlib.pyplot as plt
from cse493g1.classifiers.fc_net import *
from cse493g1.data_utils import get_CIFAR10_data
from cse493g1.gradient_check import eval_numerical_gradient, \
    eval_numerical_gradient_array
from cse493g1.solver import Solver

%matplotlib inline
plt.rcParams["figure.figsize"] = (10.0, 8.0) # Set default size of plots.
plt.rcParams["image.interpolation"] = "nearest"
plt.rcParams["image.cmap"] = "gray"

%load_ext autoreload
%autoreload 2

def rel_error(x, y):
    """Returns relative error."""
    return np.max(np.abs(x - y) / (np.maximum(1e-8, np.abs(x) + np.abs(y))))

def print_mean_std(x,axis=0):
    print(f"  means: {x.mean(axis=axis)}")
    print(f"  stds:  {x.std(axis=axis)}\n")
```

===== You can safely ignore the message below if you are NOT working on ConvolutionalNetworks.ipynb =====

You will need to compile a Cython extension for a portion of this

assignment.

The instructions to do this will be given in a section of the notebook below.

```
[ ]: # Load the (preprocessed) CIFAR-10 data.
data = get_CIFAR10_data()
for k, v in list(data.items()):
    print(f"{k}: {v.shape}")
```

```
X_train: (49000, 3, 32, 32)
y_train: (49000,)
X_val: (1000, 3, 32, 32)
y_val: (1000,)
X_test: (1000, 3, 32, 32)
y_test: (1000,)
```

2 Batch Normalization: Forward Pass

In the file `cse493g1/layers.py`, implement the batch normalization forward pass in the function `batchnorm_forward`. Once you have done so, run the following to test your implementation.

Referencing the paper linked to above in [1] may be helpful!

```
[ ]: # Check the training-time forward pass by checking means and variances
# of features both before and after batch normalization

# Simulate the forward pass for a two-layer network.
np.random.seed(493)
N, D1, D2, D3 = 200, 50, 60, 3
X = np.random.randn(N, D1)
W1 = np.random.randn(D1, D2)
W2 = np.random.randn(D2, D3)
a = np.maximum(0, X.dot(W1)).dot(W2)

print('Before batch normalization:')
print_mean_std(a,axis=0)

gamma = np.ones((D3,))
beta = np.zeros((D3,))

# Means should be close to zero and stds close to one.
print('After batch normalization (gamma=1, beta=0)')
a_norm, _ = batchnorm_forward(a, gamma, beta, {'mode': 'train'})
print_mean_std(a_norm,axis=0)

gamma = np.asarray([1.0, 2.0, 3.0])
beta = np.asarray([11.0, 12.0, 13.0])
```



```
# Now means should be close to beta and stds close to gamma.
print('After batch normalization (gamma=', gamma, ', beta=', beta, ')')
a_norm, _ = batchnorm_forward(a, gamma, beta, {'mode': 'train'})
print_mean_std(a_norm,axis=0)
```

Before batch normalization:

```
means: [-10.78452322  16.26772621 -30.25276687]
stds:  [29.12943373  37.33627641  30.56256822]
```

After batch normalization (gamma=1, beta=0)

```
means: [ 9.32587341e-17  9.88098492e-17 -1.70419234e-16]
stds:  [0.99999999  1.          0.99999999]
```

After batch normalization (gamma= [1. 2. 3.] , beta= [11. 12. 13.])

```
means: [11. 12. 13.]
stds:  [0.99999999  1.99999999  2.99999998]
```

```
[ ]: # Check the test-time forward pass by running the training-time
# forward pass many times to warm up the running averages, and then
# checking the means and variances of activations after a test-time
# forward pass.
```

```
np.random.seed(493)
N, D1, D2, D3 = 200, 50, 60, 3
W1 = np.random.randn(D1, D2)
W2 = np.random.randn(D2, D3)

bn_param = {'mode': 'train'}
gamma = np.ones(D3)
beta = np.zeros(D3)

for t in range(50):
    X = np.random.randn(N, D1)
    a = np.maximum(0, X.dot(W1)).dot(W2)
    batchnorm_forward(a, gamma, beta, bn_param)

bn_param['mode'] = 'test'
X = np.random.randn(N, D1)
a = np.maximum(0, X.dot(W1)).dot(W2)
a_norm, _ = batchnorm_forward(a, gamma, beta, bn_param)

# Means should be close to zero and stds close to one, but will be
# noisier than training-time forward passes.
print('After batch normalization (test-time):')
print_mean_std(a_norm,axis=0)
```

After batch normalization (test-time):

```
means: [ 0.02116281 -0.00291808  0.05612367]
stds:  [1.10717728  1.01804534  0.99964292]
```

3 Batch Normalization: Backward Pass

Now implement the backward pass for batch normalization in the function `batchnorm_backward`.

To derive the backward pass you should write out the computation graph for batch normalization and backprop through each of the intermediate nodes. Some intermediates may have multiple outgoing branches; make sure to sum gradients across these branches in the backward pass.

Once you have finished, run the following to numerically check your backward pass.

```
[ ]: # Gradient check batchnorm backward pass.
np.random.seed(493)
N, D = 4, 5
x = 5 * np.random.randn(N, D) + 12
gamma = np.random.randn(D)
beta = np.random.randn(D)
dout = np.random.randn(N, D)

bn_param = {'mode': 'train'}
fx = lambda x: batchnorm_forward(x, gamma, beta, bn_param)[0]
fg = lambda a: batchnorm_forward(x, a, beta, bn_param)[0]
fb = lambda b: batchnorm_forward(x, gamma, b, bn_param)[0]

dx_num = eval_numerical_gradient_array(fx, x, dout)
da_num = eval_numerical_gradient_array(fg, gamma.copy(), dout)
db_num = eval_numerical_gradient_array(fb, beta.copy(), dout)

_, cache = batchnorm_forward(x, gamma, beta, bn_param)
dx, dgamma, dbeta = batchnorm_backward(dout, cache)

# You should expect to see relative errors between 1e-13 and 1e-8.
print('dx error: ', rel_error(dx_num, dx))
print('dgamma error: ', rel_error(da_num, dgamma))
print('dbeta error: ', rel_error(db_num, dbeta))
```

```
dx error:  1.2711851983261563e-08
dgamma error:  7.346136578328727e-11
dbeta error:  8.753834990721456e-12
```

4 Batch Normalization: Alternative Backward Pass

In class we talked about two different implementations for the sigmoid backward pass. One strategy is to write out a computation graph composed of simple operations and backprop through all intermediate values. Another strategy is to work out the derivatives on paper. For example, you

can derive a very simple formula for the sigmoid function's backward pass by simplifying gradients on paper.

Surprisingly, it turns out that you can do a similar simplification for the batch normalization backward pass too!

In the forward pass, given a set of inputs $X = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_N \end{bmatrix}$,

we first calculate the mean μ and variance v . With μ and v calculated, we can calculate the standard deviation σ and normalized data Y . The equations and graph illustration below describe the computation (y_i is the i -th element of the vector Y).

$$\mu = \frac{1}{N} \sum_{k=1}^N x_k \qquad v = \frac{1}{N} \sum_{k=1}^N (x_k - \mu)^2 \qquad (1)$$

$$\sigma = \sqrt{v + \epsilon} \qquad y_i = \frac{x_i - \mu}{\sigma} \qquad (2)$$

The meat of our problem during backpropagation is to compute $\frac{\partial L}{\partial X}$, given the upstream gradient we receive, $\frac{\partial L}{\partial Y}$. To do this, recall the chain rule in calculus gives us $\frac{\partial L}{\partial X} = \frac{\partial L}{\partial Y} \cdot \frac{\partial Y}{\partial X}$.

The unknown/hard part is $\frac{\partial Y}{\partial X}$. We can find this by first deriving step-by-step our local gradients at $\frac{\partial v}{\partial X}$, $\frac{\partial \mu}{\partial X}$, $\frac{\partial \sigma}{\partial v}$, $\frac{\partial Y}{\partial \sigma}$, and $\frac{\partial Y}{\partial \mu}$, and then use the chain rule to compose these gradients (which appear in the form of vectors!) appropriately to compute $\frac{\partial Y}{\partial X}$.

If it's challenging to directly reason about the gradients over X and Y which require matrix multiplication, try reasoning about the gradients in terms of individual elements x_i and y_i first: in that case, you will need to come up with the derivations for $\frac{\partial L}{\partial x_i}$, by relying on the Chain Rule to first calculate the intermediate $\frac{\partial \mu}{\partial x_i}$, $\frac{\partial v}{\partial x_i}$, $\frac{\partial \sigma}{\partial x_i}$, then assemble these pieces to calculate $\frac{\partial y_i}{\partial x_i}$.

You should make sure each of the intermediary gradient derivations are all as simplified as possible, for ease of implementation.

After doing so, implement the simplified batch normalization backward pass in the function `batchnorm_backward_alt` and compare the two implementations by running the following. Your two implementations should compute nearly identical results, but the alternative implementation should be a bit faster.

```
[ ]: np.random.seed(493)
N, D = 100, 500
x = 5 * np.random.randn(N, D) + 12
gamma = np.random.randn(D)
beta = np.random.randn(D)
dout = np.random.randn(N, D)

bn_param = {'mode': 'train'}
out, cache = batchnorm_forward(x, gamma, beta, bn_param)
```

```

t1 = time.time()
dx1, dgamma1, dbeta1 = batchnorm_backward(dout, cache)
t2 = time.time()
dx2, dgamma2, dbeta2 = batchnorm_backward_alt(dout, cache)
t3 = time.time()

print('dx difference: ', rel_error(dx1, dx2))
print('dgamma difference: ', rel_error(dgamma1, dgamma2))
print('dbeta difference: ', rel_error(dbeta1, dbeta2))
print('speedup: %.2fx' % ((t2 - t1) / (t3 - t2)))

```

```

dx difference:  4.206930160310874e-13
dgamma difference:  0.0
dbeta difference:  0.0
speedup: 0.53x

```

5 Fully Connected Networks with Batch Normalization

Now that you have a working implementation for batch normalization, go back to your `FullyConnectedNetImproved` in the file `cse493g1/classifiers/fc_net.py`. Modify your implementation to add batch normalization.

Concretely, when the `normalization` flag is set to `"batchnorm"` in the constructor, you should insert a batch normalization layer before each ReLU nonlinearity. The outputs from the last layer of the network should not be normalized. Once you are done, run the following to gradient-check your implementation.

Hint: You might find it useful to define an additional helper layer similar to those in the file `cse493g1/layer_utils.py`.

```

[ ]: np.random.seed(493)
N, D, H1, H2, C = 2, 15, 20, 30, 10
X = np.random.randn(N, D)
y = np.random.randint(C, size=(N,))

# You should expect losses between 1e-4~1e-10 for W,
# losses between 1e-08~1e-10 for b,
# and losses between 1e-08~1e-09 for beta and gammas.
for reg in [0, 3.14]:
    print('Running check with reg = ', reg)
    model = FullyConnectedNetImproved([H1, H2], input_dim=D, num_classes=C,
                                      reg=reg, weight_scale=5e-2, dtype=np.float64,
                                      normalization='batchnorm')

    loss, grads = model.loss(X, y)
    print('Initial loss: ', loss)

    for name in sorted(grads):

```

```

f = lambda _: model.loss(X, y)[0]
grad_num = eval_numerical_gradient(f, model.params[name], verbose=False,
↪h=1e-5)
print('%s relative error: %.2e' % (name, rel_error(grad_num, grads[name])))
if reg == 0: print()

```

```

Running check with reg = 0
Initial loss: 2.076971674776514
W1 relative error: 2.21e-05
W2 relative error: 2.47e-05
W3 relative error: 3.70e-10
b1 relative error: 2.22e-03
b2 relative error: 2.22e-08
b3 relative error: 9.10e-11
beta1 relative error: 8.16e-09
beta2 relative error: 1.04e-08
gamma1 relative error: 7.75e-09
gamma2 relative error: 2.17e-08

```

```

Running check with reg = 3.14
Initial loss: 7.013903417894839
W1 relative error: 1.05e-04
W2 relative error: 6.17e-06
W3 relative error: 4.96e-07
b1 relative error: 4.44e-08
b2 relative error: 1.11e-08
b3 relative error: 1.66e-10
beta1 relative error: 6.18e-08
beta2 relative error: 3.72e-09
gamma1 relative error: 6.36e-08
gamma2 relative error: 1.06e-08

```

6 Batch Normalization for Deep Networks

Run the following to train a six-layer network on a subset of 1000 training examples both with and without batch normalization.

```

[ ]: np.random.seed(493)

# Try training a very deep net with batchnorm.
hidden_dims = [100, 100, 100, 100, 100]

num_train = 1000
small_data = {
    'X_train': data['X_train'][:num_train],
    'y_train': data['y_train'][:num_train],
    'X_val': data['X_val'],

```

```

    'y_val': data['y_val'],
}

weight_scale = 2e-2
bn_model = FullyConnectedNetImproved(hidden_dims, weight_scale=weight_scale,
    ↪normalization='batchnorm')
model = FullyConnectedNetImproved(hidden_dims, weight_scale=weight_scale,
    ↪normalization=None)

print('Solver with batch norm:')
bn_solver = Solver(bn_model, small_data,
    num_epochs=10, batch_size=50,
    update_rule='adam',
    optim_config={
        'learning_rate': 1e-3,
    },
    verbose=True, print_every=20)
bn_solver.train()

print('\nSolver without batch norm:')
solver = Solver(model, small_data,
    num_epochs=10, batch_size=50,
    update_rule='adam',
    optim_config={
        'learning_rate': 1e-3,
    },
    verbose=True, print_every=20)
solver.train()

```

Solver with batch norm:

```

(Iteration 1 / 200) loss: 2.296650
(Epoch 0 / 10) train acc: 0.129000; val_acc: 0.137000
(Epoch 1 / 10) train acc: 0.318000; val_acc: 0.258000
(Iteration 21 / 200) loss: 1.963419
(Epoch 2 / 10) train acc: 0.434000; val_acc: 0.303000
(Iteration 41 / 200) loss: 1.820266
(Epoch 3 / 10) train acc: 0.528000; val_acc: 0.337000
(Iteration 61 / 200) loss: 1.513026
(Epoch 4 / 10) train acc: 0.564000; val_acc: 0.318000
(Iteration 81 / 200) loss: 1.448250
(Epoch 5 / 10) train acc: 0.649000; val_acc: 0.316000
(Iteration 101 / 200) loss: 1.231488
(Epoch 6 / 10) train acc: 0.641000; val_acc: 0.305000
(Iteration 121 / 200) loss: 1.188077
(Epoch 7 / 10) train acc: 0.692000; val_acc: 0.297000
(Iteration 141 / 200) loss: 1.334523
(Epoch 8 / 10) train acc: 0.747000; val_acc: 0.326000

```

```
(Iteration 161 / 200) loss: 0.648037
(Epoch 9 / 10) train acc: 0.779000; val_acc: 0.334000
(Iteration 181 / 200) loss: 0.881652
(Epoch 10 / 10) train acc: 0.831000; val_acc: 0.361000
```

Solver without batch norm:

```
(Iteration 1 / 200) loss: 2.303307
(Epoch 0 / 10) train acc: 0.168000; val_acc: 0.142000
(Epoch 1 / 10) train acc: 0.199000; val_acc: 0.177000
(Iteration 21 / 200) loss: 2.072841
(Epoch 2 / 10) train acc: 0.268000; val_acc: 0.240000
(Iteration 41 / 200) loss: 1.855935
(Epoch 3 / 10) train acc: 0.341000; val_acc: 0.292000
(Iteration 61 / 200) loss: 1.747505
(Epoch 4 / 10) train acc: 0.396000; val_acc: 0.314000
(Iteration 81 / 200) loss: 1.643815
(Epoch 5 / 10) train acc: 0.427000; val_acc: 0.307000
(Iteration 101 / 200) loss: 1.638902
(Epoch 6 / 10) train acc: 0.481000; val_acc: 0.307000
(Iteration 121 / 200) loss: 1.446964
(Epoch 7 / 10) train acc: 0.478000; val_acc: 0.284000
(Iteration 141 / 200) loss: 1.353077
(Epoch 8 / 10) train acc: 0.587000; val_acc: 0.322000
(Iteration 161 / 200) loss: 1.184632
(Epoch 9 / 10) train acc: 0.540000; val_acc: 0.313000
(Iteration 181 / 200) loss: 1.167104
(Epoch 10 / 10) train acc: 0.659000; val_acc: 0.321000
```

Run the following to visualize the results from two networks trained above. You should find that using batch normalization helps the network to converge much faster.

```
[ ]: def plot_training_history(title, label, baseline, bn_solvers, plot_fn,
    ↪bl_marker='.', bn_marker='.', labels=None):
    """utility function for plotting training history"""
    plt.title(title)
    plt.xlabel(label)
    bn_plots = [plot_fn(bn_solver) for bn_solver in bn_solvers]
    bl_plot = plot_fn(baseline)
    num_bn = len(bn_plots)
    for i in range(num_bn):
        label='with_norm'
        if labels is not None:
            label += str(labels[i])
        plt.plot(bn_plots[i], bn_marker, label=label)
    label='baseline'
    if labels is not None:
        label += str(labels[0])
    plt.plot(bl_plot, bl_marker, label=label)
```

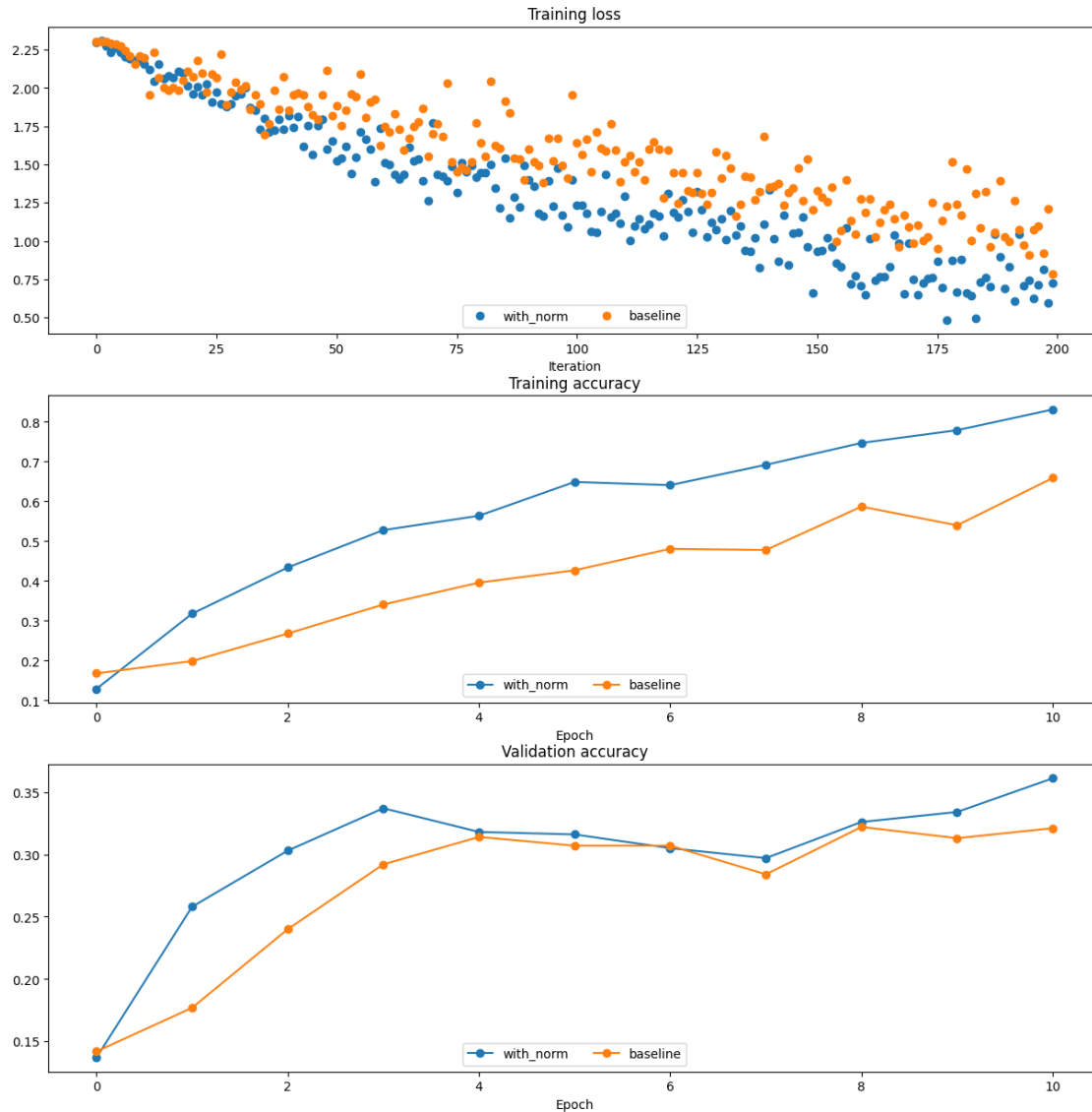
```

plt.legend(loc='lower center', ncol=num_bn+1)

plt.subplot(3, 1, 1)
plot_training_history('Training loss', 'Iteration', solver, [bn_solver], \
                      lambda x: x.loss_history, bl_marker='o', bn_marker='o')
plt.subplot(3, 1, 2)
plot_training_history('Training accuracy', 'Epoch', solver, [bn_solver], \
                      lambda x: x.train_acc_history, bl_marker='-o', \
                      bn_marker='-o')
plt.subplot(3, 1, 3)
plot_training_history('Validation accuracy', 'Epoch', solver, [bn_solver], \
                      lambda x: x.val_acc_history, bl_marker='-o', \
                      bn_marker='-o')

plt.gcf().set_size_inches(15, 15)
plt.show()

```

7 Batch Normalization and Initialization

We will now run a small experiment to study the interaction of batch normalization and weight initialization.

The first cell will train eight-layer networks both with and without batch normalization using different scales for weight initialization. The second layer will plot training accuracy, validation set accuracy, and training loss as a function of the weight initialization scale.

```
[ ]: np.random.seed(493)

# Try training a very deep net with batchnorm.
```

```

hidden_dims = [50, 50, 50, 50, 50, 50, 50]
num_train = 1000
small_data = {
    'X_train': data['X_train'][:num_train],
    'y_train': data['y_train'][:num_train],
    'X_val': data['X_val'],
    'y_val': data['y_val'],
}

bn_solvers_ws = {}
solvers_ws = {}
weight_scales = np.logspace(-4, 0, num=20)
for i, weight_scale in enumerate(weight_scales):
    print('Running weight scale %d / %d' % (i + 1, len(weight_scales)))
    bn_model = FullyConnectedNetImproved(hidden_dims,
    ↪weight_scale=weight_scale, normalization='batchnorm')
    model = FullyConnectedNetImproved(hidden_dims, weight_scale=weight_scale,
    ↪normalization=None)

    bn_solver = Solver(bn_model, small_data,
                        num_epochs=10, batch_size=50,
                        update_rule='adam',
                        optim_config={
                            'learning_rate': 1e-3,
                        },
                        verbose=False, print_every=200)
    bn_solver.train()
    bn_solvers_ws[weight_scale] = bn_solver

    solver = Solver(model, small_data,
                    num_epochs=10, batch_size=50,
                    update_rule='adam',
                    optim_config={
                        'learning_rate': 1e-3,
                    },
                    verbose=False, print_every=200)
    solver.train()
    solvers_ws[weight_scale] = solver

```

```

Running weight scale 1 / 20
Running weight scale 2 / 20
Running weight scale 3 / 20
Running weight scale 4 / 20
Running weight scale 5 / 20
Running weight scale 6 / 20
Running weight scale 7 / 20
Running weight scale 8 / 20

```

```

Running weight scale 9 / 20
Running weight scale 10 / 20
Running weight scale 11 / 20
Running weight scale 12 / 20
Running weight scale 13 / 20
Running weight scale 14 / 20
Running weight scale 15 / 20
Running weight scale 16 / 20

/content/drive/My Drive/cse493g1/assignments/assignment3/cse493g1/layers.py:151:
RuntimeWarning: overflow encountered in exp
    x = np.exp(x)
/content/drive/My Drive/cse493g1/assignments/assignment3/cse493g1/layers.py:153:
RuntimeWarning: invalid value encountered in divide
    x = x / row_sums[:,np.newaxis]
/content/drive/My Drive/cse493g1/assignments/assignment3/cse493g1/layers.py:156:
RuntimeWarning: divide by zero encountered in log
    loss = np.sum(-np.log(loss_array))

Running weight scale 17 / 20
Running weight scale 18 / 20
Running weight scale 19 / 20
Running weight scale 20 / 20

```

```

[ ]: # Plot results of weight scale experiment.
best_train_accs, bn_best_train_accs = [], []
best_val_accs, bn_best_val_accs = [], []
final_train_loss, bn_final_train_loss = [], []

for ws in weight_scales:
    best_train_accs.append(max(solvers_ws[ws].train_acc_history))
    bn_best_train_accs.append(max(bn_solvers_ws[ws].train_acc_history))

    best_val_accs.append(max(solvers_ws[ws].val_acc_history))
    bn_best_val_accs.append(max(bn_solvers_ws[ws].val_acc_history))

    final_train_loss.append(np.mean(solvers_ws[ws].loss_history[-100:]))
    bn_final_train_loss.append(np.mean(bn_solvers_ws[ws].loss_history[-100:]))

plt.subplot(3, 1, 1)
plt.title('Best val accuracy vs. weight initialization scale')
plt.xlabel('Weight initialization scale')
plt.ylabel('Best val accuracy')
plt.semilogx(weight_scales, best_val_accs, '-o', label='baseline')
plt.semilogx(weight_scales, bn_best_val_accs, '-o', label='batchnorm')
plt.legend(ncol=2, loc='lower right')

plt.subplot(3, 1, 2)

```

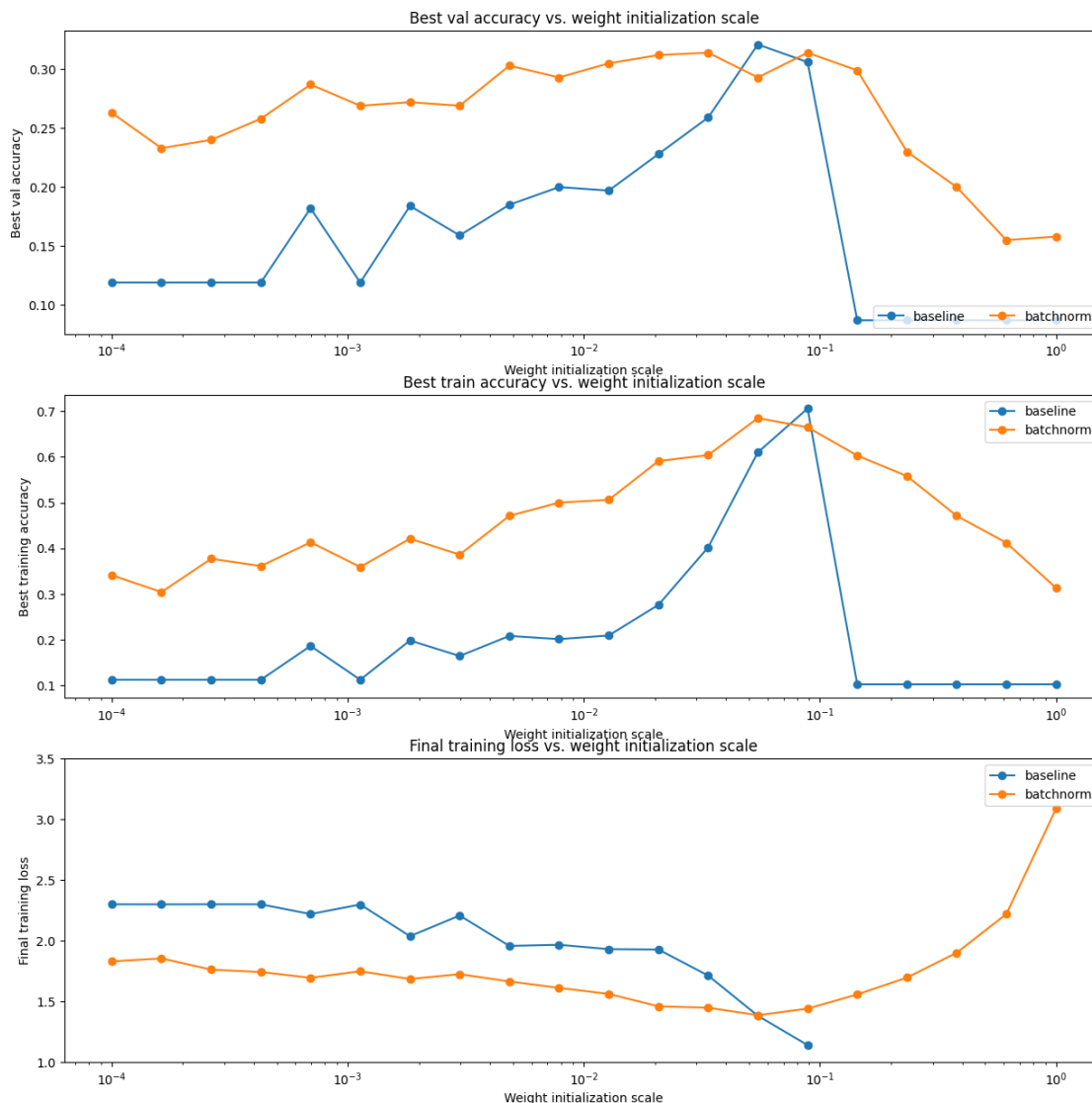
```

plt.title('Best train accuracy vs. weight initialization scale')
plt.xlabel('Weight initialization scale')
plt.ylabel('Best training accuracy')
plt.semilogx(weight_scales, best_train_accs, '-o', label='baseline')
plt.semilogx(weight_scales, bn_best_train_accs, '-o', label='batchnorm')
plt.legend()

plt.subplot(3, 1, 3)
plt.title('Final training loss vs. weight initialization scale')
plt.xlabel('Weight initialization scale')
plt.ylabel('Final training loss')
plt.semilogx(weight_scales, final_train_loss, '-o', label='baseline')
plt.semilogx(weight_scales, bn_final_train_loss, '-o', label='batchnorm')
plt.legend()
plt.gca().set_ylim(1.0, 3.5)

plt.gcf().set_size_inches(15, 15)
plt.show()

```



7.1 Inline Question 1:

Describe the results of this experiment. How does the weight initialization scale affect models with/without batch normalization differently, and why?

7.2 Answer:

The experiments show that training loss, validation accuracy and training accuracy vary a lot by weight initialization for our baseline model while they remain more flat for our batchnorm model. This implies that batchnorm model is more robust to changes in weight initialization. Slight changes in weight initialization do not cause accuracy to dip dramatically.

The reason this happens is because if x has a mean far away from the origin and high variance, slight changes to w will make a large change in classification of x . Normalizing x makes our

layers less sensitive to changes in w , which makes it less sensitive to our random initialization. Batchnormalization also helps with gradient flow and doesn't let gradients explode or vanish. This is because all dw are based on some factor of x , thus very high or very low values of x can cause dw to explode or vanish. Normalization makes sure x doesn't have these extreme values

8 Batch Normalization and Batch Size

We will now run a small experiment to study the interaction of batch normalization and batch size.

The first cell will train 6-layer networks both with and without batch normalization using different batch sizes. The second layer will plot training accuracy and validation set accuracy over time.

```
[ ]: def run_batchsize_experiments(normalization_mode):
    np.random.seed(493)

    # Try training a very deep net with batchnorm.
    hidden_dims = [100, 100, 100, 100, 100]
    num_train = 1000
    small_data = {
        'X_train': data['X_train'][:num_train],
        'y_train': data['y_train'][:num_train],
        'X_val': data['X_val'],
        'y_val': data['y_val'],
    }
    n_epochs=10
    weight_scale = 2e-2
    batch_sizes = [5,10,50]
    lr = 10**(-3.5)
    solver_bsize = batch_sizes[0]

    print('No normalization: batch size = ',solver_bsize)
    model = FullyConnectedNetImproved(hidden_dims, weight_scale=weight_scale,
    ↪normalization=None)
    solver = Solver(model, small_data,
                    num_epochs=n_epochs, batch_size=solver_bsize,
                    update_rule='adam',
                    optim_config={
                        'learning_rate': lr,
                    },
                    verbose=False)
    solver.train()

    bn_solvers = []
    for i in range(len(batch_sizes)):
        b_size=batch_sizes[i]
        print('Normalization: batch size = ',b_size)
        bn_model = FullyConnectedNetImproved(hidden_dims,
    ↪weight_scale=weight_scale, normalization=normalization_mode)
```

```

    bn_solver = Solver(bn_model, small_data,
                        num_epochs=n_epochs, batch_size=b_size,
                        update_rule='adam',
                        optim_config={
                            'learning_rate': lr,
                        },
                        verbose=False)
    bn_solver.train()
    bn_solvers.append(bn_solver)

    return bn_solvers, solver, batch_sizes

batch_sizes = [5,10,50]
bn_solvers_bsize, solver_bsize, batch_sizes = \
    ↪run_batchsize_experiments('batchnorm')

```

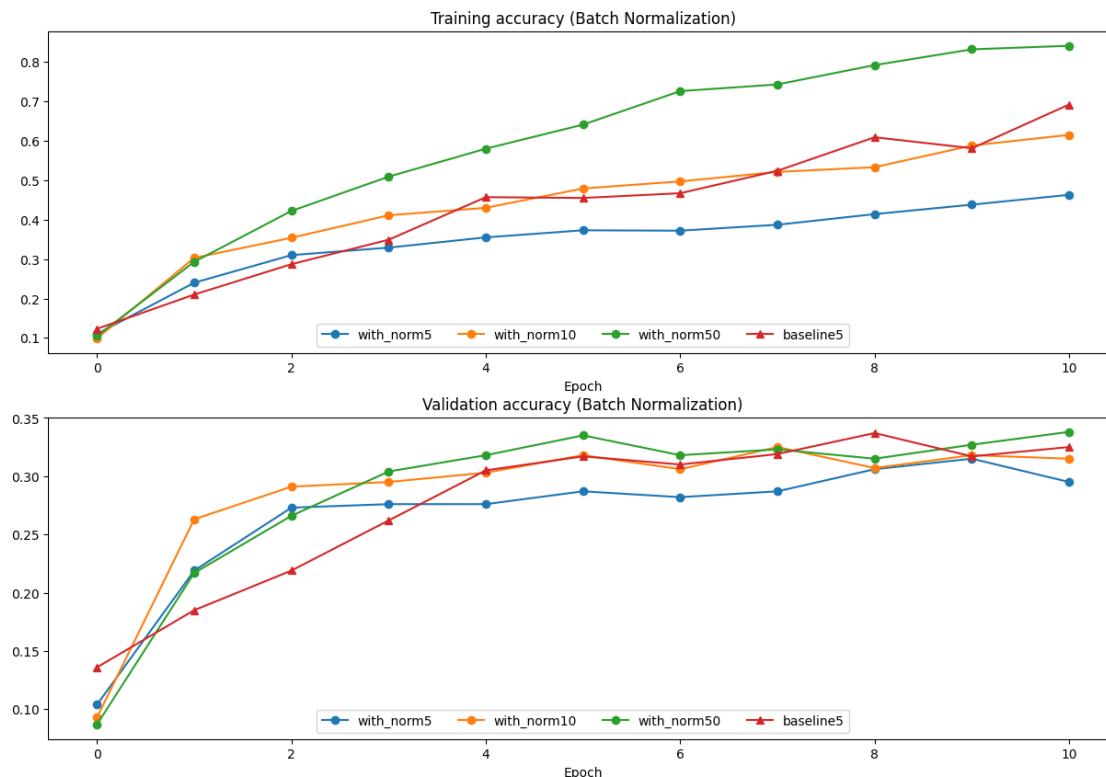
No normalization: batch size = 5
 Normalization: batch size = 5
 Normalization: batch size = 10
 Normalization: batch size = 50

```

[ ]: plt.subplot(2, 1, 1)
plot_training_history('Training accuracy (Batch Normalization)', 'Epoch', \
    ↪solver_bsize, bn_solvers_bsize, \
                        lambda x: x.train_acc_history, bl_marker='^-', \
    ↪bn_marker='-o', labels=batch_sizes)
plt.subplot(2, 1, 2)
plot_training_history('Validation accuracy (Batch Normalization)', 'Epoch', \
    ↪solver_bsize, bn_solvers_bsize, \
                        lambda x: x.val_acc_history, bl_marker='^-', \
    ↪bn_marker='-o', labels=batch_sizes)

plt.gcf().set_size_inches(15, 10)
plt.show()

```



8.1 Inline Question 2:

Describe the results of this experiment. What does this imply about the relationship between batch normalization and batch size? Why is this relationship observed?

8.2 Answer:

Our experiment tested validation and training accuracy of our model with different batch sizes. We observe that batchnorm with high batch size (50) performs better than our baseline model. However, batchnorm with lower batch sizes (5,10) perform worse than baseline. This indicated a high dependence of batchnorm performance to batch size. This relationship is observed because as we calculate mean and variance from a small batch, it is not representative enough of the training data. This causes our normalization to be inaccurate and reduces batchnorm performance.

9 Layer Normalization

Batch normalization has proved to be effective in making networks easier to train, but the dependency on batch size makes it less useful in complex networks which have a cap on the input batch size due to hardware limitations.

Several alternatives to batch normalization have been proposed to mitigate this problem; one such technique is Layer Normalization [2]. Instead of normalizing over the batch, we normalize over the

features. In other words, when using Layer Normalization, each feature vector corresponding to a single datapoint is normalized based on the sum of all terms within that feature vector.

[2] Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton. “Layer Normalization.” *stat 1050* (2016): 21.

9.1 Inline Question 3:

Which of these data preprocessing steps is analogous to batch normalization, and which is analogous to layer normalization?

1. Scaling each image in the dataset, so that the RGB channels for each row of pixels within an image sums up to 1.
2. Scaling each image in the dataset, so that the RGB channels for all pixels within an image sums up to 1.
3. Subtracting the mean image of the dataset from each image in the dataset.
4. Setting all RGB values to either 0 or 1 depending on a given threshold.

9.2 Answer:

2 is analogous to layer norm as it involve normalizing over features of a given training point. We are not looking at other training points to calculate mean or variance in these steps.

3 is analogous to batch norm as it involves normalizing over all images of the dataset. This step would be affected by batch sizes and would require keeping a running mean and variance for test-time.

10 Layer Normalization: Implementation

Now you'll implement layer normalization. This step should be relatively straightforward, as conceptually the implementation is almost identical to that of batch normalization. One significant difference though is that for layer normalization, we do not keep track of the moving moments, and the testing phase is identical to the training phase, where the mean and variance are directly calculated per datapoint.

Here's what you need to do:

- In `cse493g1/layers.py`, implement the forward pass for layer normalization in the function `layernorm_forward`.

Run the cell below to check your results. * In `cse493g1/layers.py`, implement the backward pass for layer normalization in the function `layernorm_backward`.

Run the second cell below to check your results. * Modify `cse493g1/classifiers/fc_net.py` to add layer normalization to the `FullyConnectedNetImproved`. When the `normalization` flag is set to "layernorm" in the constructor, you should insert a layer normalization layer before each ReLU nonlinearity.

Run the third cell below to run the batch size experiment on layer normalization.

```
[ ]: # Check the training-time forward pass by checking means and variances
# of features both before and after layer normalization.

# Simulate the forward pass for a two-layer network.
np.random.seed(493)
N, D1, D2, D3 = 4, 50, 60, 3
X = np.random.randn(N, D1)
W1 = np.random.randn(D1, D2)
W2 = np.random.randn(D2, D3)
a = np.maximum(0, X.dot(W1)).dot(W2)

print('Before layer normalization:')
print_mean_std(a,axis=1)

gamma = np.ones(D3)
beta = np.zeros(D3)

# Means should be close to zero and stds close to one.
print('After layer normalization (gamma=1, beta=0)')
a_norm, _ = layernorm_forward(a, gamma, beta, {'mode': 'train'})
print_mean_std(a_norm,axis=1)

gamma = np.asarray([3.0,3.0,3.0])
beta = np.asarray([5.0,5.0,5.0])

# Now means should be close to beta and stds close to gamma.
print('After layer normalization (gamma=', gamma, ', beta=', beta, ')')
a_norm, _ = layernorm_forward(a, gamma, beta, {'mode': 'train'})
print_mean_std(a_norm,axis=1)
```

Before layer normalization:

```
means: [-13.55016261  10.48803421   0.51090326  16.1646569 ]
stds:  [10.43227939  44.64705887  61.08455449  28.07932638]
```

After layer normalization (gamma=1, beta=0)

```
means: [0.00000000e+00 0.00000000e+00 0.00000000e+00 7.40148683e-17]
stds:  [0.99999995  1.          1.          0.99999999]
```

After layer normalization (gamma= [3. 3. 3.] , beta= [5. 5. 5.])

```
means: [5. 5. 5. 5.]
stds:  [2.99999986 2.99999999 3.          2.99999998]
```

```
[ ]: # Gradient check batchnorm backward pass.
np.random.seed(493)
N, D = 4, 5
x = 5 * np.random.randn(N, D) + 12
```

```

gamma = np.random.randn(D)
beta = np.random.randn(D)
dout = np.random.randn(N, D)

ln_param = {}
fx = lambda x: layernorm_forward(x, gamma, beta, ln_param)[0]
fg = lambda a: layernorm_forward(x, a, beta, ln_param)[0]
fb = lambda b: layernorm_forward(x, gamma, b, ln_param)[0]

dx_num = eval_numerical_gradient_array(fx, x, dout)
da_num = eval_numerical_gradient_array(fg, gamma.copy(), dout)
db_num = eval_numerical_gradient_array(fb, beta.copy(), dout)

_, cache = layernorm_forward(x, gamma, beta, ln_param)
dx, dgamma, dbeta = layernorm_backward(dout, cache)

# You should expect to see relative errors between 1e-12 and 1e-8.
print('dx error: ', rel_error(dx_num, dx))
print('dgamma error: ', rel_error(da_num, dgamma))
print('dbeta error: ', rel_error(db_num, dbeta))

```

```

dx error:  3.964278018989327e-10
dgamma error:  9.614319694248486e-12
dbeta error:  8.753834990721456e-12

```

11 Layer Normalization and Batch Size

We will now run the previous batch size experiment with layer normalization instead of batch normalization. Compared to the previous experiment, you should see a markedly smaller influence of batch size on the training history!

```

[ ]: ln_solvers_bsize, solver_bsize, batch_sizes = \
    ↪run_batchsize_experiments('layernorm')

plt.subplot(2, 1, 1)
plot_training_history('Training accuracy (Layer Normalization)', 'Epoch', \
    ↪solver_bsize, ln_solvers_bsize, \
    ↪lambda x: x.train_acc_history, bl_marker='^-', \
    ↪bn_marker='-o', labels=batch_sizes)
plt.subplot(2, 1, 2)
plot_training_history('Validation accuracy (Layer Normalization)', 'Epoch', \
    ↪solver_bsize, ln_solvers_bsize, \
    ↪lambda x: x.val_acc_history, bl_marker='^-', \
    ↪bn_marker='-o', labels=batch_sizes)

plt.gcf().set_size_inches(15, 10)
plt.show()

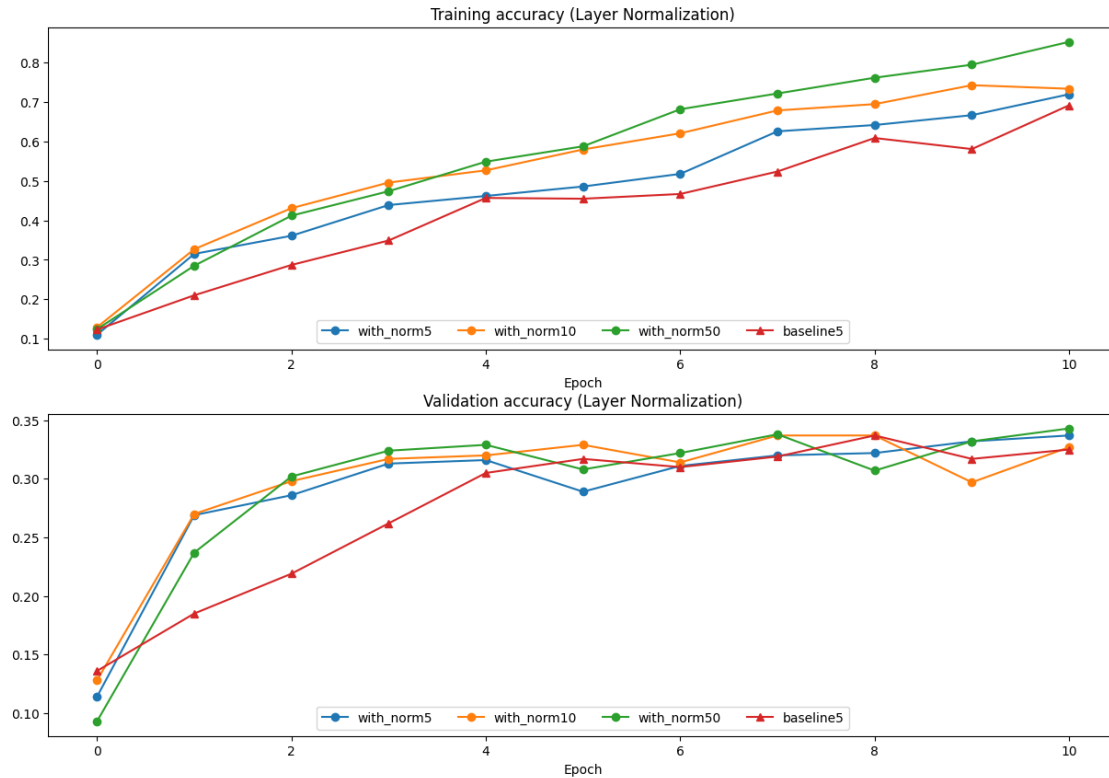
```

No normalization: batch size = 5

Normalization: batch size = 5

Normalization: batch size = 10

Normalization: batch size = 50



11.1 Inline Question 4:

When is layer normalization likely to not work well, and why?

1. Using it in a very deep network
2. Having a very small dimension of features
3. Having a high regularization term

11.2 Answer:

2. Layer norm might face issues when dimension of features is low. This could be because the calculations for mean and standard deviation across the features would be noisy when the number of features is low.

Dropout

February 16, 2024

```
[ ]: # This mounts your Google Drive to the Colab VM.
from google.colab import drive
drive.mount('/content/drive')

# TODO: Enter the foldername in your Drive where you have saved the unzipped
# assignment folder, e.g. 'cse493g1/assignments/assignment3/'
FOLDERNAME = 'cse493g1/assignments/assignment3/'
assert FOLDERNAME is not None, "[!] Enter the foldername."

# Now that we've mounted your Drive, this ensures that
# the Python interpreter of the Colab VM can load
# python files from within it.
import sys
sys.path.append('/content/drive/My Drive/{}'.format(FOLDERNAME))

# This downloads the COCO dataset to your Drive
# if it doesn't already exist.
%cd /content/drive/My\ Drive/$FOLDERNAME/cse493g1/datasets/
!bash get_datasets.sh
%cd /content/drive/My\ Drive/$FOLDERNAME
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call `drive.mount("/content/drive", force_remount=True)`.
/content/drive/My Drive/cse493g1/assignments/assignment3/cse493g1/datasets
/content/drive/My Drive/cse493g1/assignments/assignment3

1 Dropout

Dropout [1] is a technique for regularizing neural networks by randomly setting some output activations to zero during the forward pass. In this exercise, you will implement a dropout layer and modify your fully connected network to optionally use dropout.

[1] Geoffrey E. Hinton et al, “Improving neural networks by preventing co-adaptation of feature detectors”, arXiv 2012

```
[ ]: # Setup cell.
import time
import numpy as np
```

```

import matplotlib.pyplot as plt
from cse493g1.classifiers.fc_net import *
from cse493g1.data_utils import get_CIFAR10_data
from cse493g1.gradient_check import eval_numerical_gradient, \
    eval_numerical_gradient_array
from cse493g1.solver import Solver

%matplotlib inline
plt.rcParams["figure.figsize"] = (10.0, 8.0) # Set default size of plots.
plt.rcParams["image.interpolation"] = "nearest"
plt.rcParams["image.cmap"] = "gray"

%load_ext autoreload
%autoreload 2

def rel_error(x, y):
    """Returns relative error."""
    return np.max(np.abs(x - y) / (np.maximum(1e-8, np.abs(x) + np.abs(y))))

```

The autoreload extension is already loaded. To reload it, use:

```
%reload_ext autoreload
```

```

[ ]: # Load the (preprocessed) CIFAR-10 data.
data = get_CIFAR10_data()
for k, v in list(data.items()):
    print(f"{k}: {v.shape}")

```

```

X_train: (49000, 3, 32, 32)
y_train: (49000,)
X_val: (1000, 3, 32, 32)
y_val: (1000,)
X_test: (1000, 3, 32, 32)
y_test: (1000,)

```

2 Dropout: Forward Pass

In the file `cse493g1/layers.py`, implement the forward pass for dropout. Since dropout behaves differently during training and testing, make sure to implement the operation for both modes.

Once you have done so, run the cell below to test your implementation.

```

[ ]: np.random.seed(493)
x = np.random.randn(500, 500) + 10

for p in [0.25, 0.4, 0.7]:
    out, _ = dropout_forward(x, {'mode': 'train', 'p': p})
    out_test, _ = dropout_forward(x, {'mode': 'test', 'p': p})

```

```

print('Running tests with p = ', p)
print('Mean of input: ', x.mean())
print('Mean of train-time output: ', out.mean())
print('Mean of test-time output: ', out_test.mean())
print('Fraction of train-time output set to zero: ', (out == 0).mean())
print('Fraction of test-time output set to zero: ', (out_test == 0).mean())
print()

```

```

Running tests with p = 0.25
Mean of input: 9.998754292315606
Mean of train-time output: 10.013822700086228
Mean of test-time output: 9.998754292315606
Fraction of train-time output set to zero: 0.749592
Fraction of test-time output set to zero: 0.0

```

```

Running tests with p = 0.4
Mean of input: 9.998754292315606
Mean of train-time output: 9.961259703367766
Mean of test-time output: 9.998754292315606
Fraction of train-time output set to zero: 0.601452
Fraction of test-time output set to zero: 0.0

```

```

Running tests with p = 0.7
Mean of input: 9.998754292315606
Mean of train-time output: 9.983016036268745
Mean of test-time output: 9.998754292315606
Fraction of train-time output set to zero: 0.3011
Fraction of test-time output set to zero: 0.0

```

3 Dropout: Backward Pass

In the file `cse493g1/layers.py`, implement the backward pass for dropout. After doing so, run the following cell to numerically gradient-check your implementation.

```

[ ]: np.random.seed(493)
x = np.random.randn(10, 10) + 10
dout = np.random.randn(*x.shape)

dropout_param = {'mode': 'train', 'p': 0.2, 'seed': 123}
out, cache = dropout_forward(x, dropout_param)
dx = dropout_backward(dout, cache)
dx_num = eval_numerical_gradient_array(lambda xx: dropout_forward(xx,
    ↪ dropout_param)[0], x, dout)

# Error should be around e-10 or less.
print('dx relative error: ', rel_error(dx, dx_num))

```

dx relative error: 5.445599254826896e-11

3.1 Inline Question 1:

What happens if we do not divide the values being passed through inverse dropout by p in the dropout layer? Why does that happen?

3.2 Answer:

If we do not divide values passed through inverse dropout by p , we will get very different outputs at train-time vs test-time. This is because at train time we get an output considering only $p\%$ of the neurons which are active. While at test time, we use all the neurons to calculate the output and hence it will be scaled up by the probability of keeping the neurons. Hence, we scale down our train time outputs for consistency across train and test time outputs.

4 Fully Connected Networks with Dropout

In the file `cse493g1/classifiers/fc_net.py`, copy your `FullyConnectedNetBasic` into `FullyConnectedNetImproved` and modify this new net to use dropout. Specifically, if the constructor of the network receives a value that is not 1 for the `dropout_keep_ratio` parameter, then the net should add a dropout layer immediately after every ReLU nonlinearity. After doing so, run the following to numerically gradient-check your implementation.

```
[ ]: np.random.seed(493)
N, D, H1, H2, C = 2, 15, 20, 30, 10
X = np.random.randn(N, D)
y = np.random.randint(C, size=(N,))

for dropout_keep_ratio in [1, 0.75, 0.5]:
    print('Running check with dropout = ', dropout_keep_ratio)
    model = FullyConnectedNetImproved(
        [H1, H2],
        input_dim=D,
        num_classes=C,
        weight_scale=5e-2,
        dtype=np.float64,
        dropout_keep_ratio=dropout_keep_ratio,
        seed=123
    )

    loss, grads = model.loss(X, y)
    print('Initial loss: ', loss)

    # Relative errors should be around e-6 or less.
    # Note that it's fine if for dropout_keep_ratio=1 you have W2 error be on
    ↳ the order of e-5.
    for name in sorted(grads):
        f = lambda _: model.loss(X, y)[0]
```



```

        grad_num = eval_numerical_gradient(f, model.params[name],
↪ verbose=False, h=1e-5)
        print('%s relative error: %.2e' % (name, rel_error(grad_num,
↪ grads[name])))
    print()

```

```

Running check with dropout = 1
Initial loss: 2.299821914918452
W1 relative error: 9.66e-08
W2 relative error: 4.43e-06
W3 relative error: 1.41e-06
b1 relative error: 9.10e-09
b2 relative error: 1.36e-07
b3 relative error: 1.34e-10

```

```

Running check with dropout = 0.75
Initial loss: 2.303014228386485
W1 relative error: 2.04e-07
W2 relative error: 8.49e-08
W3 relative error: 2.07e-07
b1 relative error: 3.90e-08
b2 relative error: 2.71e-09
b3 relative error: 1.00e-10

```

```

Running check with dropout = 0.5
Initial loss: 2.30486036957328
W1 relative error: 6.66e-08
W2 relative error: 1.09e-08
W3 relative error: 3.47e-08
b1 relative error: 2.08e-09
b2 relative error: 8.34e-10
b3 relative error: 8.54e-11

```

5 Regularization Experiment

As an experiment, we will train a pair of two-layer networks on 500 training examples: one will use no dropout, and one will use a keep probability of 0.25. We will then visualize the training and validation accuracies of the two networks over time.

```

[ ]: # Train two identical nets, one with dropout and one without.
np.random.seed(493)
num_train = 500
small_data = {
    'X_train': data['X_train'][:num_train],
    'y_train': data['y_train'][:num_train],
    'X_val': data['X_val'],

```

```

        'y_val': data['y_val'],
    }

    solvers = {}
    dropout_choices = [1, 0.25]
    for dropout_keep_ratio in dropout_choices:
        model = FullyConnectedNetImproved(
            [500],
            dropout_keep_ratio=dropout_keep_ratio
        )
        print(dropout_keep_ratio)

        solver = Solver(
            model,
            small_data,
            num_epochs=25,
            batch_size=100,
            update_rule='adam',
            optim_config={'learning_rate': 5e-4},
            verbose=True,
            print_every=100
        )
        solver.train()
        solvers[dropout_keep_ratio] = solver
    print()

```

1

(Iteration 1 / 125) loss: 7.287606

(Epoch 0 / 25) train acc: 0.212000; val_acc: 0.120000

/content/drive/My Drive/cse493g1/assignments/assignment3/cse493g1/layers.py:156:

RuntimeWarning: divide by zero encountered in log

loss = np.sum(-np.log(loss_array))

(Epoch 1 / 25) train acc: 0.410000; val_acc: 0.272000

(Epoch 2 / 25) train acc: 0.494000; val_acc: 0.226000

(Epoch 3 / 25) train acc: 0.630000; val_acc: 0.258000

(Epoch 4 / 25) train acc: 0.676000; val_acc: 0.255000

(Epoch 5 / 25) train acc: 0.710000; val_acc: 0.270000

(Epoch 6 / 25) train acc: 0.780000; val_acc: 0.292000

(Epoch 7 / 25) train acc: 0.872000; val_acc: 0.265000

(Epoch 8 / 25) train acc: 0.846000; val_acc: 0.288000

(Epoch 9 / 25) train acc: 0.878000; val_acc: 0.259000

(Epoch 10 / 25) train acc: 0.914000; val_acc: 0.302000

(Epoch 11 / 25) train acc: 0.932000; val_acc: 0.302000

(Epoch 12 / 25) train acc: 0.932000; val_acc: 0.263000

(Epoch 13 / 25) train acc: 0.928000; val_acc: 0.267000

(Epoch 14 / 25) train acc: 0.964000; val_acc: 0.270000

```
(Epoch 15 / 25) train acc: 0.932000; val_acc: 0.276000
(Epoch 16 / 25) train acc: 0.926000; val_acc: 0.292000
(Epoch 17 / 25) train acc: 0.974000; val_acc: 0.272000
(Epoch 18 / 25) train acc: 0.966000; val_acc: 0.272000
(Epoch 19 / 25) train acc: 0.946000; val_acc: 0.290000
(Epoch 20 / 25) train acc: 0.978000; val_acc: 0.301000
(Iteration 101 / 125) loss: 0.321758
(Epoch 21 / 25) train acc: 0.988000; val_acc: 0.302000
(Epoch 22 / 25) train acc: 0.986000; val_acc: 0.310000
(Epoch 23 / 25) train acc: 0.988000; val_acc: 0.302000
(Epoch 24 / 25) train acc: 0.984000; val_acc: 0.293000
(Epoch 25 / 25) train acc: 0.998000; val_acc: 0.304000
```

0.25

```
(Iteration 1 / 125) loss: 16.330096
(Epoch 0 / 25) train acc: 0.202000; val_acc: 0.153000
(Epoch 1 / 25) train acc: 0.380000; val_acc: 0.243000
(Epoch 2 / 25) train acc: 0.452000; val_acc: 0.251000
(Epoch 3 / 25) train acc: 0.518000; val_acc: 0.291000
(Epoch 4 / 25) train acc: 0.580000; val_acc: 0.307000
(Epoch 5 / 25) train acc: 0.652000; val_acc: 0.306000
(Epoch 6 / 25) train acc: 0.682000; val_acc: 0.300000
(Epoch 7 / 25) train acc: 0.702000; val_acc: 0.313000
(Epoch 8 / 25) train acc: 0.708000; val_acc: 0.306000
(Epoch 9 / 25) train acc: 0.752000; val_acc: 0.313000
(Epoch 10 / 25) train acc: 0.792000; val_acc: 0.294000
(Epoch 11 / 25) train acc: 0.800000; val_acc: 0.295000
(Epoch 12 / 25) train acc: 0.826000; val_acc: 0.300000
(Epoch 13 / 25) train acc: 0.826000; val_acc: 0.315000
(Epoch 14 / 25) train acc: 0.850000; val_acc: 0.327000
(Epoch 15 / 25) train acc: 0.826000; val_acc: 0.314000
(Epoch 16 / 25) train acc: 0.864000; val_acc: 0.320000
(Epoch 17 / 25) train acc: 0.872000; val_acc: 0.306000
(Epoch 18 / 25) train acc: 0.890000; val_acc: 0.304000
(Epoch 19 / 25) train acc: 0.892000; val_acc: 0.311000
(Epoch 20 / 25) train acc: 0.884000; val_acc: 0.309000
(Iteration 101 / 125) loss: 5.844120
(Epoch 21 / 25) train acc: 0.848000; val_acc: 0.294000
(Epoch 22 / 25) train acc: 0.882000; val_acc: 0.322000
(Epoch 23 / 25) train acc: 0.864000; val_acc: 0.315000
(Epoch 24 / 25) train acc: 0.914000; val_acc: 0.310000
(Epoch 25 / 25) train acc: 0.912000; val_acc: 0.304000
```

```
[ ]: # Plot train and validation accuracies of the two models.
train_accs = []
val_accs = []
```

```

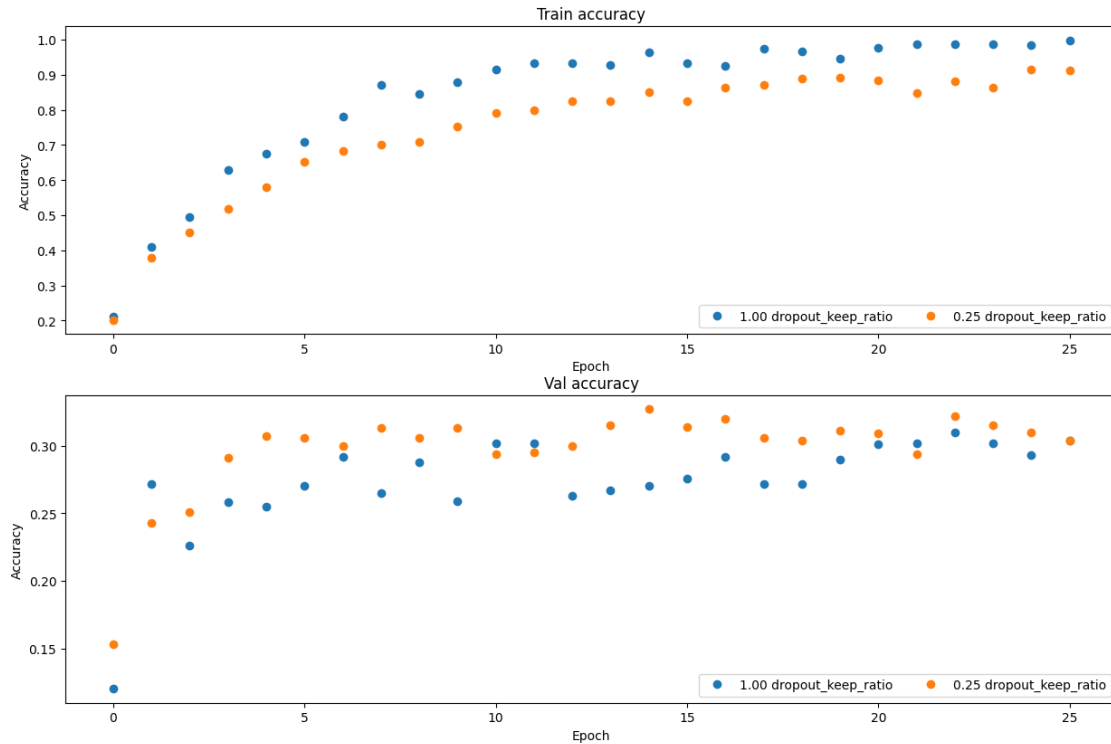
for dropout_keep_ratio in dropout_choices:
    solver = solvers[dropout_keep_ratio]
    train_accs.append(solver.train_acc_history[-1])
    val_accs.append(solver.val_acc_history[-1])

plt.subplot(3, 1, 1)
for dropout_keep_ratio in dropout_choices:
    plt.plot(
        solvers[dropout_keep_ratio].train_acc_history, 'o', label='%0.2f_
↳dropout_keep_ratio' % dropout_keep_ratio)
plt.title('Train accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(ncol=2, loc='lower right')

plt.subplot(3, 1, 2)
for dropout_keep_ratio in dropout_choices:
    plt.plot(
        solvers[dropout_keep_ratio].val_acc_history, 'o', label='%0.2f_
↳dropout_keep_ratio' % dropout_keep_ratio)
plt.title('Val accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(ncol=2, loc='lower right')

plt.gcf().set_size_inches(15, 15)
plt.show()

```



5.1 Inline Question 2:

Compare the validation and training accuracies with and without dropout – what do your results suggest about dropout as a regularizer?

5.2 Answer:

Training accuracies for no dropout are higher than training accuracies with dropout. On the other hand, Validation accuracies with no dropout are lower than validation accuracies with dropout. This means that dropout is acting as a regularizer and preventing our model from overfitting the training data so it can perform better on validation data.

However, from our current accuracy graphs there doesn't seem to be a big improvement to our validation accuracy from using dropout - we might want to use dropout in combination with other regularization methods

Convolutional Networks

February 16, 2024

```
[ ]: # This mounts your Google Drive to the Colab VM.
from google.colab import drive
drive.mount('/content/drive')

# TODO: Enter the foldername in your Drive where you have saved the unzipped
# assignment folder, e.g. 'cse493g1/assignments/assignment3/'
FOLDERNAME = 'cse493g1/assignments/assignment3/'
assert FOLDERNAME is not None, "[!] Enter the foldername."

# Now that we've mounted your Drive, this ensures that
# the Python interpreter of the Colab VM can load
# python files from within it.
import sys
sys.path.append('/content/drive/My Drive/{}'.format(FOLDERNAME))

# This downloads the COCO dataset to your Drive
# if it doesn't already exist.
%cd /content/drive/My\ Drive/$FOLDERNAME/cse493g1/datasets/
!bash get_datasets.sh
%cd /content/drive/My\ Drive/$FOLDERNAME
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call `drive.mount("/content/drive", force_remount=True)`.

/content/drive/My Drive/cse493g1/assignments/assignment3/cse493g1/datasets
/content/drive/My Drive/cse493g1/assignments/assignment3

1 Convolutional Networks

So far we have worked with deep fully connected networks, using them to explore different optimization strategies and network architectures. Fully connected networks are a good testbed for experimentation because they are very computationally efficient, but in practice all state-of-the-art results use convolutional networks instead.

First you will implement several layer types that are used in convolutional networks. You will then use these layers to train a convolutional network on the CIFAR-10 dataset.

```
[ ]: # Setup cell.
import numpy as np
```

```

import matplotlib.pyplot as plt
from cse493g1.classifiers.cnn import *
from cse493g1.data_utils import get_CIFAR10_data
from cse493g1.gradient_check import eval_numerical_gradient_array,
    ↪eval_numerical_gradient
from cse493g1.layers import *
from cse493g1.fast_layers import *
from cse493g1.solver import Solver

%matplotlib inline
plt.rcParams['figure.figsize'] = (10.0, 8.0) # set default size of plots
plt.rcParams['image.interpolation'] = 'nearest'
plt.rcParams['image.cmap'] = 'gray'

# for auto-reloading external modules
# see http://stackoverflow.com/questions/1907993/
    ↪autoreload-of-modules-in-ipython
%load_ext autoreload
%autoreload 2

def rel_error(x, y):
    """ returns relative error """
    return np.max(np.abs(x - y) / (np.maximum(1e-8, np.abs(x) + np.abs(y))))

```

```

[ ]: # Load the (preprocessed) CIFAR-10 data.
data = get_CIFAR10_data()
for k, v in list(data.items()):
    print(f"{k}: {v.shape}")

```

```

X_train: (49000, 3, 32, 32)
y_train: (49000,)
X_val: (1000, 3, 32, 32)
y_val: (1000,)
X_test: (1000, 3, 32, 32)
y_test: (1000,)

```

2 Convolution: Naive Forward Pass

The core of a convolutional network is the convolution operation. In the file `cse493g1/layers.py`, implement the forward pass for the convolution layer in the function `conv_forward_naive`.

You don't have to worry too much about efficiency at this point; just write the code in whatever way you find most clear.

You can test your implementation by running the following:

```

[ ]: x_shape = (2, 3, 4, 4)
     w_shape = (3, 3, 4, 4)

```

```

x = np.linspace(-0.1, 0.5, num=np.prod(x_shape)).reshape(x_shape)
w = np.linspace(-0.2, 0.3, num=np.prod(w_shape)).reshape(w_shape)
b = np.linspace(-0.1, 0.2, num=3)

conv_param = {'stride': 2, 'pad': 1}
out, _ = conv_forward_naive(x, w, b, conv_param)
correct_out = np.array([[[[-0.08759809, -0.10987781],
                           [-0.18387192, -0.2109216 ]],
                          [[ 0.21027089,  0.21661097],
                           [ 0.22847626,  0.23004637]],
                          [[ 0.50813986,  0.54309974],
                           [ 0.64082444,  0.67101435]]],
                         [[[-0.98053589, -1.03143541],
                           [-1.19128892, -1.24695841]],
                          [[ 0.69108355,  0.66880383],
                           [ 0.59480972,  0.56776003]],
                          [[ 2.36270298,  2.36904306],
                           [ 2.38090835,  2.38247847]]]])

# Compare your output to ours; difference should be around e-8
print('Testing conv_forward_naive')
print('difference: ', rel_error(out, correct_out))

```

```

Testing conv_forward_naive
difference:  2.2121476417505994e-08

```

2.1 Aside: Image Processing via Convolutions

As fun way to both check your implementation and gain a better understanding of the type of operation that convolutional layers can perform, we will set up an input containing two images and manually set up filters that perform common image processing operations (grayscale conversion and edge detection). The convolution forward pass will apply these operations to each of the input images. We can then visualize the results as a sanity check.

```

[ ]: from imageio import imread
     from PIL import Image

     kitten = imread('cse493g1/notebook_images/kitten.jpg')
     puppy = imread('cse493g1/notebook_images/puppy.jpg')
     # kitten is wide, and puppy is already square
     d = kitten.shape[1] - kitten.shape[0]
     kitten_cropped = kitten[:, d//2:-d//2, :]

     img_size = 200 # Make this smaller if it runs too slow
     resized_puppy = np.array(Image.fromarray(puppy).resize((img_size, img_size)))
     resized_kitten = np.array(Image.fromarray(kitten_cropped).resize((img_size,
     ↪img_size)))
     x = np.zeros((2, 3, img_size, img_size))

```



```

x[0, :, :, :] = resized_puppy.transpose((2, 0, 1))
x[1, :, :, :] = resized_kitten.transpose((2, 0, 1))

# Set up a convolutional weights holding 2 filters, each 3x3
w = np.zeros((2, 3, 3, 3))

# The first filter converts the image to grayscale.
# Set up the red, green, and blue channels of the filter.
w[0, 0, :, :] = [[0, 0, 0], [0, 0.3, 0], [0, 0, 0]]
w[0, 1, :, :] = [[0, 0, 0], [0, 0.6, 0], [0, 0, 0]]
w[0, 2, :, :] = [[0, 0, 0], [0, 0.1, 0], [0, 0, 0]]

# Second filter detects horizontal edges in the blue channel.
w[1, 2, :, :] = [[1, 2, 1], [0, 0, 0], [-1, -2, -1]]

# Vector of biases. We don't need any bias for the grayscale
# filter, but for the edge detection filter we want to add 128
# to each output so that nothing is negative.
b = np.array([0, 128])

# Compute the result of convolving each input in x with each filter in w,
# offsetting by b, and storing the results in out.
out, _ = conv_forward_naive(x, w, b, {'stride': 1, 'pad': 1})

def imshow_no_ax(img, normalize=True):
    """ Tiny helper to show images as uint8 and remove axis labels """
    if normalize:
        img_max, img_min = np.max(img), np.min(img)
        img = 255.0 * (img - img_min) / (img_max - img_min)
    plt.imshow(img.astype('uint8'))
    plt.gca().axis('off')

# Show the original images and the results of the conv operation
plt.subplot(2, 3, 1)
imshow_no_ax(puppy, normalize=False)
plt.title('Original image')
plt.subplot(2, 3, 2)
imshow_no_ax(out[0, 0])
plt.title('Grayscale')
plt.subplot(2, 3, 3)
imshow_no_ax(out[0, 1])
plt.title('Edges')
plt.subplot(2, 3, 4)
imshow_no_ax(kitten_cropped, normalize=False)
plt.subplot(2, 3, 5)
imshow_no_ax(out[1, 0])
plt.subplot(2, 3, 6)

```

```
imshow_no_ax(out[1, 1])
plt.show()
```

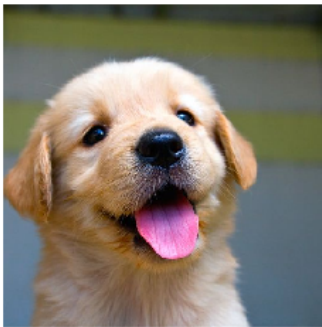
<ipython-input-8-ed6e0fc1d3d2>:4: DeprecationWarning: Starting with ImageIO v3 the behavior of this function will switch to that of iio.v3.imread. To keep the current behavior (and make this warning disappear) use `import imageio.v2 as imageio` or call `imageio.v2.imread` directly.

```
kitten = imread('cse493g1/notebook_images/kitten.jpg')
```

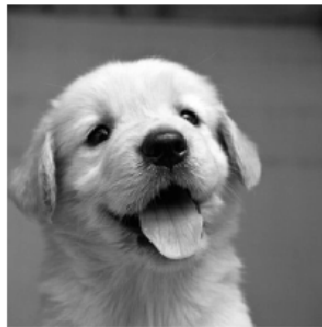
<ipython-input-8-ed6e0fc1d3d2>:5: DeprecationWarning: Starting with ImageIO v3 the behavior of this function will switch to that of iio.v3.imread. To keep the current behavior (and make this warning disappear) use `import imageio.v2 as imageio` or call `imageio.v2.imread` directly.

```
puppy = imread('cse493g1/notebook_images/puppy.jpg')
```

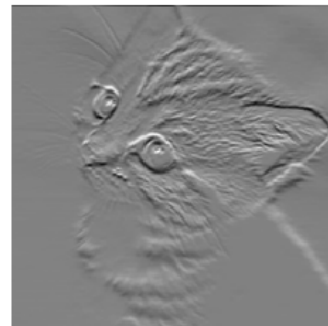
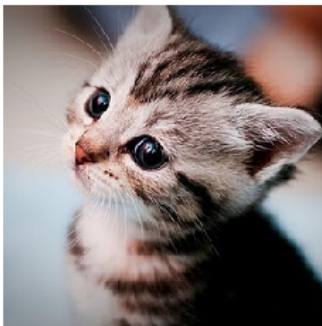
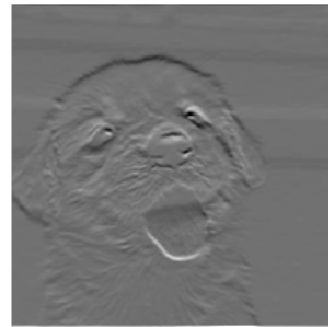
Original image



Grayscale



Edges



3 Convolution: Naive Backward Pass

Implement the backward pass for the convolution operation in the function `conv_backward_naive` in the file `cse493g1/layers.py`. Again, you don't need to worry too much about computational efficiency.

When you are done, run the following to check your backward pass with a numeric gradient check.

```
[ ]: np.random.seed(493)
x = np.random.randn(4, 3, 5, 5)
w = np.random.randn(2, 3, 3, 3)
b = np.random.randn(2,)
dout = np.random.randn(4, 2, 5, 5)
conv_param = {'stride': 1, 'pad': 1}

dx_num = eval_numerical_gradient_array(lambda x: conv_forward_naive(x, w, b,
    ↪conv_param)[0], x, dout)
dw_num = eval_numerical_gradient_array(lambda w: conv_forward_naive(x, w, b,
    ↪conv_param)[0], w, dout)
db_num = eval_numerical_gradient_array(lambda b: conv_forward_naive(x, w, b,
    ↪conv_param)[0], b, dout)

out, cache = conv_forward_naive(x, w, b, conv_param)
dx, dw, db = conv_backward_naive(dout, cache)

# Your errors should be around e-8 or less.
print('Testing conv_backward_naive function')
print('dx error: ', rel_error(dx, dx_num))
print('dw error: ', rel_error(dw, dw_num))
print('db error: ', rel_error(db, db_num))
```

```
Testing conv_backward_naive function
dx error:  3.001996916775403e-09
dw error:  3.4403223291170234e-10
db error:  4.314875022006533e-10
```

4 Max-Pooling: Naive Forward Pass

Implement the forward pass for the max-pooling operation in the function `max_pool_forward_naive` in the file `cse493g1/layers.py`. Again, don't worry too much about computational efficiency.

Check your implementation by running the following:

```
[ ]: x_shape = (2, 3, 4, 4)
x = np.linspace(-0.3, 0.4, num=np.prod(x_shape)).reshape(x_shape)
pool_param = {'pool_width': 2, 'pool_height': 2, 'stride': 2}

out, _ = max_pool_forward_naive(x, pool_param)

correct_out = np.array([[[[-0.26315789, -0.24842105],
                           [-0.20421053, -0.18947368]],
                          [[-0.14526316, -0.13052632],
                           [-0.08631579, -0.07157895]]],
```

```

[[[-0.02736842, -0.01263158],
  [ 0.03157895,  0.04631579]]],
[[[ 0.09052632,  0.10526316],
  [ 0.14947368,  0.16421053]],
[[ 0.20842105,  0.22315789],
  [ 0.26736842,  0.28210526]],
[[ 0.32631579,  0.34105263],
  [ 0.38526316,  0.4         ]]]])

# Compare your output with ours. Difference should be on the order of e-8.
print('Testing max_pool_forward_naive function:')
print('difference: ', rel_error(out, correct_out))

```

Testing max_pool_forward_naive function:
 difference: 4.1666665157267834e-08

5 Max-Pooling: Naive Backward

Implement the backward pass for the max-pooling operation in the function `max_pool_backward_naive` in the file `cse493g1/layers.py`. You don't need to worry about computational efficiency.

Check your implementation with numeric gradient checking by running the following:

```

[ ]: np.random.seed(493)
x = np.random.randn(3, 2, 8, 8)
dout = np.random.randn(3, 2, 4, 4)
pool_param = {'pool_height': 2, 'pool_width': 2, 'stride': 2}

dx_num = eval_numerical_gradient_array(lambda x: max_pool_forward_naive(x,
    ↪pool_param)[0], x, dout)

out, cache = max_pool_forward_naive(x, pool_param)
dx = max_pool_backward_naive(dout, cache)

# Your error should be on the order of e-12
print('Testing max_pool_backward_naive function:')
print('dx error: ', rel_error(dx, dx_num))

```

Testing max_pool_backward_naive function:
 dx error: 3.2756208694894018e-12

6 Fast Layers

Making convolution and pooling layers fast can be challenging. To spare you the pain, we've provided fast implementations of the forward and backward passes for convolution and pooling layers in the file `cse493g1/fast_layers.py`.

6.0.1 Execute the below cell, save the notebook, and restart the runtime

The fast convolution implementation depends on a Cython extension; to compile it, run the cell below. Next, save the Colab notebook (File > Save) and **restart the runtime** (Runtime > Restart runtime). You can then re-execute the preceeding cells from top to bottom and skip the cell below as you only need to run it once for the compilation step.

```
[ ]: # Remember to restart the runtime after executing this cell!
%cd /content/drive/My\ Drive/$FOLDERNAME/cse493g1/
!python setup.py build_ext --inplace
%cd /content/drive/My\ Drive/$FOLDERNAME/
```

```
/content/drive/My Drive/cse493g1/assignments/assignment3/cse493g1
/content/drive/My Drive/cse493g1/assignments/assignment3
```

The API for the fast versions of the convolution and pooling layers is exactly the same as the naive versions that you implemented above: the forward pass receives data, weights, and parameters and produces outputs and a cache object; the backward pass receives upstream derivatives and the cache object and produces gradients with respect to the data and weights.

Note: The fast implementation for pooling will only perform optimally if the pooling regions are non-overlapping and tile the input. If these conditions are not met then the fast pooling implementation will not be much faster than the naive implementation.

You can compare the performance of the naive and fast versions of these layers by running the following:

```
[ ]: # Rel errors should be around e-9 or less.
from cse493g1.fast_layers import conv_forward_fast, conv_backward_fast
from time import time
np.random.seed(493)
x = np.random.randn(100, 3, 31, 31)
w = np.random.randn(25, 3, 3, 3)
b = np.random.randn(25,)
dout = np.random.randn(100, 25, 16, 16)
conv_param = {'stride': 2, 'pad': 1}

t0 = time()
out_naive, cache_naive = conv_forward_naive(x, w, b, conv_param)
t1 = time()
out_fast, cache_fast = conv_forward_fast(x, w, b, conv_param)
t2 = time()

print('Testing conv_forward_fast:')
print('Naive: %fs' % (t1 - t0))
print('Fast: %fs' % (t2 - t1))
print('Speedup: %fx' % ((t1 - t0) / (t2 - t1)))
print('Difference: ', rel_error(out_naive, out_fast))

t0 = time()
```

```

dx_naive, dw_naive, db_naive = conv_backward_naive(dout, cache_naive)
t1 = time()
dx_fast, dw_fast, db_fast = conv_backward_fast(dout, cache_fast)
t2 = time()

print('\nTesting conv_backward_fast:')
print('Naive: %fs' % (t1 - t0))
print('Fast: %fs' % (t2 - t1))
print('Speedup: %fx' % ((t1 - t0) / (t2 - t1)))
print('dx difference: ', rel_error(dx_naive, dx_fast))
print('dw difference: ', rel_error(dw_naive, dw_fast))
print('db difference: ', rel_error(db_naive, db_fast))

```

```

Testing conv_forward_fast:
Naive: 3.464671s
Fast: 0.009616s
Speedup: 360.288689x
Difference: 6.9816349604656895e-12

```

```

Testing conv_backward_fast:
Naive: 6.089191s
Fast: 0.008278s
Speedup: 735.596745x
dx difference: 1.4129232514043081e-11
dw difference: 3.092666522381916e-13
db difference: 0.0

```

```

[ ]: # Relative errors should be close to 0.0.
from cse493g1.fast_layers import max_pool_forward_fast, max_pool_backward_fast
np.random.seed(493)
x = np.random.randn(100, 3, 32, 32)
dout = np.random.randn(100, 3, 16, 16)
pool_param = {'pool_height': 2, 'pool_width': 2, 'stride': 2}

t0 = time()
out_naive, cache_naive = max_pool_forward_naive(x, pool_param)
t1 = time()
out_fast, cache_fast = max_pool_forward_fast(x, pool_param)
t2 = time()

print('Testing pool_forward_fast:')
print('Naive: %fs' % (t1 - t0))
print('fast: %fs' % (t2 - t1))
print('speedup: %fx' % ((t1 - t0) / (t2 - t1)))
print('difference: ', rel_error(out_naive, out_fast))

t0 = time()

```

```

dx_naive = max_pool_backward_naive(dout, cache_naive)
t1 = time()
dx_fast = max_pool_backward_fast(dout, cache_fast)
t2 = time()

print('\nTesting pool_backward_fast:')
print('Naive: %fs' % (t1 - t0))
print('fast: %fs' % (t2 - t1))
print('speedup: %fx' % ((t1 - t0) / (t2 - t1)))
print('dx difference: ', rel_error(dx_naive, dx_fast))

```

```

Testing pool_forward_fast:
Naive: 0.278426s
fast: 0.003428s
speedup: 81.210153x
difference: 0.0

```

```

Testing pool_backward_fast:
Naive: 0.398285s
fast: 0.010458s
speedup: 38.084283x
dx difference: 0.0

```

7 Convolutional “Sandwich” Layers

In the previous assignment, we introduced the concept of “sandwich” layers that combine multiple operations into commonly used patterns. In the file `cse493g1/layer_utils.py` you will find sandwich layers that implement a few commonly used patterns for convolutional networks. Run the cells below to sanity check their usage.

```

[ ]: from cse493g1.layer_utils import conv_relu_pool_forward, conv_relu_pool_backward
np.random.seed(493)
x = np.random.randn(2, 3, 16, 16)
w = np.random.randn(3, 3, 3, 3)
b = np.random.randn(3,)
dout = np.random.randn(2, 3, 8, 8)
conv_param = {'stride': 1, 'pad': 1}
pool_param = {'pool_height': 2, 'pool_width': 2, 'stride': 2}

out, cache = conv_relu_pool_forward(x, w, b, conv_param, pool_param)
dx, dw, db = conv_relu_pool_backward(dout, cache)

dx_num = eval_numerical_gradient_array(lambda x: conv_relu_pool_forward(x, w,
    ↪b, conv_param, pool_param)[0], x, dout)
dw_num = eval_numerical_gradient_array(lambda w: conv_relu_pool_forward(x, w,
    ↪b, conv_param, pool_param)[0], w, dout)

```

```

db_num = eval_numerical_gradient_array(lambda b: conv_relu_pool_forward(x, w,
    ↪b, conv_param, pool_param)[0], b, dout)

# Relative errors should be around e-8 or less
print('Testing conv_relu_pool')
print('dx error: ', rel_error(dx_num, dx))
print('dw error: ', rel_error(dw_num, dw))
print('db error: ', rel_error(db_num, db))

```

```

Testing conv_relu_pool
dx error:  3.0063520770488714e-08
dw error:  6.194086751504152e-10
db error:  3.2819548156748916e-07

```

```

[ ]: from cse493g1.layer_utils import conv_relu_forward, conv_relu_backward
np.random.seed(493)
x = np.random.randn(2, 3, 8, 8)
w = np.random.randn(3, 3, 3, 3)
b = np.random.randn(3,)
dout = np.random.randn(2, 3, 8, 8)
conv_param = {'stride': 1, 'pad': 1}

out, cache = conv_relu_forward(x, w, b, conv_param)
dx, dw, db = conv_relu_backward(dout, cache)

dx_num = eval_numerical_gradient_array(lambda x: conv_relu_forward(x, w, b,
    ↪conv_param)[0], x, dout)
dw_num = eval_numerical_gradient_array(lambda w: conv_relu_forward(x, w, b,
    ↪conv_param)[0], w, dout)
db_num = eval_numerical_gradient_array(lambda b: conv_relu_forward(x, w, b,
    ↪conv_param)[0], b, dout)

# Relative errors should be around e-8 or less
print('Testing conv_relu:')
print('dx error: ', rel_error(dx_num, dx))
print('dw error: ', rel_error(dw_num, dw))
print('db error: ', rel_error(db_num, db))

```

```

Testing conv_relu:
dx error:  2.55653568336329e-09
dw error:  1.5835432103810795e-09
db error:  1.926243366624518e-10

```

8 Three-Layer Convolutional Network

Now that you have implemented all the necessary layers, we can put them together into a simple convolutional network.

Open the file `cse493g1/classifiers/cnn.py` and complete the implementation of the `ThreeLayerConvNet` class. Remember you can use the `fast/sandwich` layers (already imported for you) in your implementation. Run the following cells to help you debug:

8.1 Sanity Check Loss

After you build a new network, one of the first things you should do is sanity check the loss. When we use the softmax loss, we expect the loss for random weights (and no regularization) to be about $\log(C)$ for C classes. When we add regularization the loss should go up slightly.

```
[ ]: model = ThreeLayerConvNet()

N = 50
X = np.random.randn(N, 3, 32, 32)
y = np.random.randint(10, size=N)

loss, grads = model.loss(X, y)
print('Initial loss (no regularization): ', loss)

model.reg = 0.5
loss, grads = model.loss(X, y)
print('Initial loss (with regularization): ', loss)
```

Initial loss (no regularization): 2.3025841266964537

Initial loss (with regularization): 2.50868095468031

8.2 Gradient Check

After the loss looks reasonable, use numeric gradient checking to make sure that your backward pass is correct. When you use numeric gradient checking you should use a small amount of artificial data and a small number of neurons at each layer. Note: correct implementations may still have relative errors up to the order of e^{-2} .

```
[ ]: num_inputs = 2
input_dim = (3, 16, 16)
reg = 0.0
num_classes = 10
np.random.seed(493)
X = np.random.randn(num_inputs, *input_dim)
y = np.random.randint(num_classes, size=num_inputs)

model = ThreeLayerConvNet(
    num_filters=3,
    filter_size=3,
    input_dim=input_dim,
    hidden_dim=7,
    dtype=np.float64
)
loss, grads = model.loss(X, y)
```

```

# Errors should be small, but correct implementations may have
# relative errors up to the order of e-2
for param_name in sorted(grads):
    f = lambda _: model.loss(X, y)[0]
    param_grad_num = eval_numerical_gradient(f, model.params[param_name],
    ↪ verbose=False, h=1e-6)
    e = rel_error(param_grad_num, grads[param_name])
    print('%s max relative error: %e' % (param_name, rel_error(param_grad_num,
    ↪ grads[param_name])))

```

```

W1 max relative error: 1.142114e-01
W2 max relative error: 1.046013e-02
W3 max relative error: 4.801769e-05
b1 max relative error: 3.922210e-05
b2 max relative error: 2.879157e-07
b3 max relative error: 1.078649e-09

```

8.3 Overfit Small Data

A nice trick is to train your model with just a few training samples. You should be able to overfit small datasets, which will result in very high training accuracy and comparatively low validation accuracy.

```

[ ]: np.random.seed(493)

num_train = 100
small_data = {
    'X_train': data['X_train'][:num_train],
    'y_train': data['y_train'][:num_train],
    'X_val': data['X_val'],
    'y_val': data['y_val'],
}

model = ThreeLayerConvNet(weight_scale=1e-2)

solver = Solver(
    model,
    small_data,
    num_epochs=15,
    batch_size=50,
    update_rule='adam',
    optim_config={'learning_rate': 1e-3},
    verbose=True,
    print_every=1
)
solver.train()

```

(Iteration 1 / 30) loss: 2.377935
(Epoch 0 / 15) train acc: 0.140000; val_acc: 0.110000
(Iteration 2 / 30) loss: 3.507474
(Epoch 1 / 15) train acc: 0.170000; val_acc: 0.119000
(Iteration 3 / 30) loss: 4.381055
(Iteration 4 / 30) loss: 3.571722
(Epoch 2 / 15) train acc: 0.240000; val_acc: 0.097000
(Iteration 5 / 30) loss: 2.401453
(Iteration 6 / 30) loss: 2.270795
(Epoch 3 / 15) train acc: 0.230000; val_acc: 0.098000
(Iteration 7 / 30) loss: 2.451352
(Iteration 8 / 30) loss: 2.472043
(Epoch 4 / 15) train acc: 0.290000; val_acc: 0.156000
(Iteration 9 / 30) loss: 2.028178
(Iteration 10 / 30) loss: 1.889899
(Epoch 5 / 15) train acc: 0.450000; val_acc: 0.182000
(Iteration 11 / 30) loss: 1.845204
(Iteration 12 / 30) loss: 1.650750
(Epoch 6 / 15) train acc: 0.310000; val_acc: 0.164000
(Iteration 13 / 30) loss: 1.722978
(Iteration 14 / 30) loss: 1.817188
(Epoch 7 / 15) train acc: 0.500000; val_acc: 0.200000
(Iteration 15 / 30) loss: 1.494970
(Iteration 16 / 30) loss: 1.549230
(Epoch 8 / 15) train acc: 0.580000; val_acc: 0.178000
(Iteration 17 / 30) loss: 1.428538
(Iteration 18 / 30) loss: 1.449242
(Epoch 9 / 15) train acc: 0.620000; val_acc: 0.180000
(Iteration 19 / 30) loss: 1.337344
(Iteration 20 / 30) loss: 1.045957
(Epoch 10 / 15) train acc: 0.610000; val_acc: 0.182000
(Iteration 21 / 30) loss: 1.052689
(Iteration 22 / 30) loss: 1.299899
(Epoch 11 / 15) train acc: 0.670000; val_acc: 0.185000
(Iteration 23 / 30) loss: 1.076239
(Iteration 24 / 30) loss: 0.885635
(Epoch 12 / 15) train acc: 0.750000; val_acc: 0.203000
(Iteration 25 / 30) loss: 0.777443
(Iteration 26 / 30) loss: 0.860241
(Epoch 13 / 15) train acc: 0.820000; val_acc: 0.212000
(Iteration 27 / 30) loss: 0.725782
(Iteration 28 / 30) loss: 0.645265
(Epoch 14 / 15) train acc: 0.870000; val_acc: 0.202000
(Iteration 29 / 30) loss: 0.480345
(Iteration 30 / 30) loss: 0.449025
(Epoch 15 / 15) train acc: 0.890000; val_acc: 0.197000

```
[ ]: # Print final training accuracy.
print(
    "Small data training accuracy:",
    solver.check_accuracy(small_data['X_train'], small_data['y_train'])
)
```

Small data training accuracy: 0.82

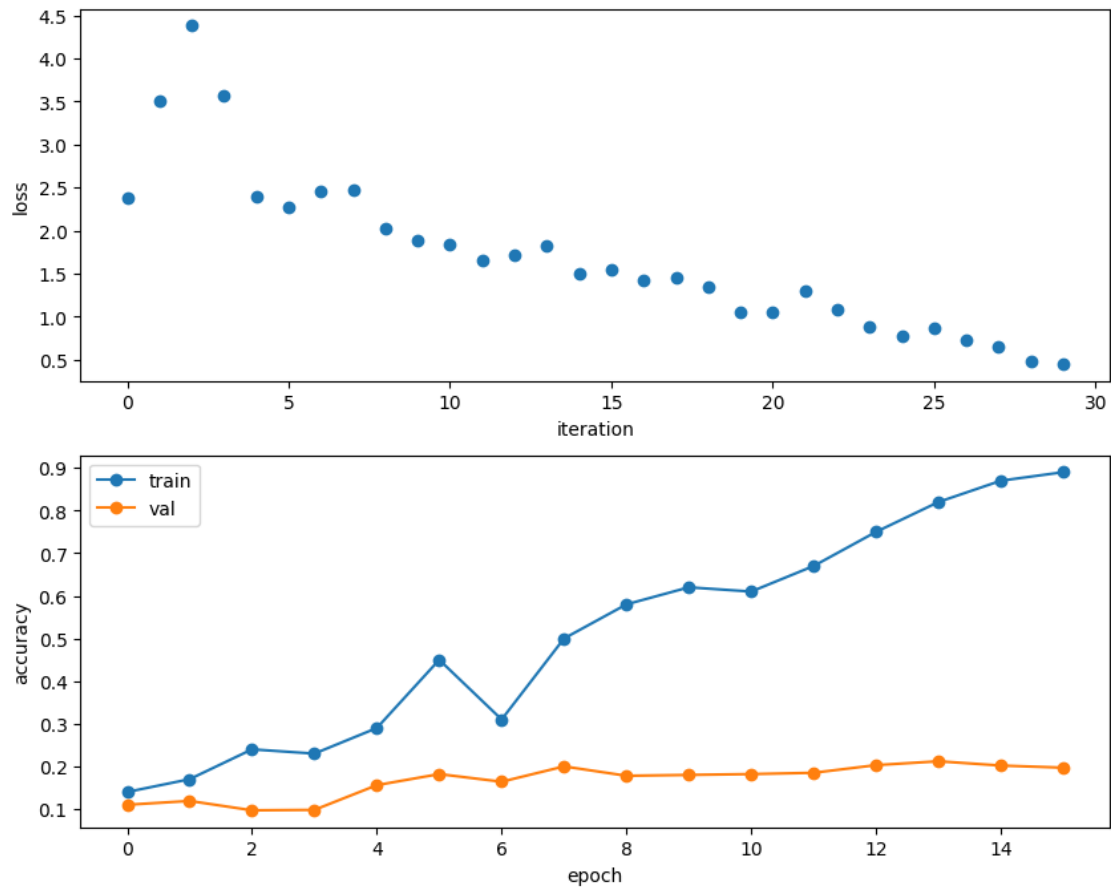
```
[ ]: # Print final validation accuracy.
print(
    "Small data validation accuracy:",
    solver.check_accuracy(small_data['X_val'], small_data['y_val'])
)
```

Small data validation accuracy: 0.212

Plotting the loss, training accuracy, and validation accuracy should show clear overfitting:

```
[ ]: plt.subplot(2, 1, 1)
plt.plot(solver.loss_history, 'o')
plt.xlabel('iteration')
plt.ylabel('loss')

plt.subplot(2, 1, 2)
plt.plot(solver.train_acc_history, '-o')
plt.plot(solver.val_acc_history, '-o')
plt.legend(['train', 'val'], loc='upper left')
plt.xlabel('epoch')
plt.ylabel('accuracy')
plt.show()
```



8.4 Train the Network

By training the three-layer convolutional network for one epoch, you should achieve greater than 40% accuracy on the training set:

```
[ ]: model = ThreeLayerConvNet(weight_scale=0.001, hidden_dim=500, reg=0.001)

solver = Solver(
    model,
    data,
    num_epochs=1,
    batch_size=50,
    update_rule='adam',
    optim_config={'learning_rate': 1e-3},
    verbose=True,
    print_every=20
)
solver.train()
```

(Iteration 1 / 980) loss: 2.304620
(Epoch 0 / 1) train acc: 0.100000; val_acc: 0.104000
(Iteration 21 / 980) loss: 2.133366
(Iteration 41 / 980) loss: 2.210642
(Iteration 61 / 980) loss: 1.698347
(Iteration 81 / 980) loss: 2.180361
(Iteration 101 / 980) loss: 1.833051
(Iteration 121 / 980) loss: 2.064002
(Iteration 141 / 980) loss: 1.889457
(Iteration 161 / 980) loss: 1.794145
(Iteration 181 / 980) loss: 1.818827
(Iteration 201 / 980) loss: 1.938266
(Iteration 221 / 980) loss: 1.629249
(Iteration 241 / 980) loss: 1.739791
(Iteration 261 / 980) loss: 1.860865
(Iteration 281 / 980) loss: 1.911879
(Iteration 301 / 980) loss: 1.773437
(Iteration 321 / 980) loss: 1.812807
(Iteration 341 / 980) loss: 1.554410
(Iteration 361 / 980) loss: 1.358729
(Iteration 381 / 980) loss: 1.790137
(Iteration 401 / 980) loss: 1.907026
(Iteration 421 / 980) loss: 1.957498
(Iteration 441 / 980) loss: 1.289243
(Iteration 461 / 980) loss: 1.904827
(Iteration 481 / 980) loss: 1.568783
(Iteration 501 / 980) loss: 1.478035
(Iteration 521 / 980) loss: 1.389411
(Iteration 541 / 980) loss: 1.525054
(Iteration 561 / 980) loss: 1.445444
(Iteration 581 / 980) loss: 1.601799
(Iteration 601 / 980) loss: 1.921848
(Iteration 621 / 980) loss: 1.470561
(Iteration 641 / 980) loss: 2.109136
(Iteration 661 / 980) loss: 1.305093
(Iteration 681 / 980) loss: 1.765340
(Iteration 701 / 980) loss: 1.591652
(Iteration 721 / 980) loss: 1.633411
(Iteration 741 / 980) loss: 1.609258
(Iteration 761 / 980) loss: 1.392624
(Iteration 781 / 980) loss: 1.853354
(Iteration 801 / 980) loss: 1.576398
(Iteration 821 / 980) loss: 1.296810
(Iteration 841 / 980) loss: 1.436183
(Iteration 861 / 980) loss: 1.788521
(Iteration 881 / 980) loss: 1.639770
(Iteration 901 / 980) loss: 1.407738
(Iteration 921 / 980) loss: 1.841744

```
(Iteration 941 / 980) loss: 1.590494
(Iteration 961 / 980) loss: 1.354923
(Epoch 1 / 1) train acc: 0.490000; val_acc: 0.499000
```

```
[ ]: # Print final training accuracy.
print(
    "Full data training accuracy:",
    solver.check_accuracy(data['X_train'], data['y_train'])
)
```

Full data training accuracy: 0.4910204081632653

```
[ ]: # Print final validation accuracy.
print(
    "Full data validation accuracy:",
    solver.check_accuracy(data['X_val'], data['y_val'])
)
```

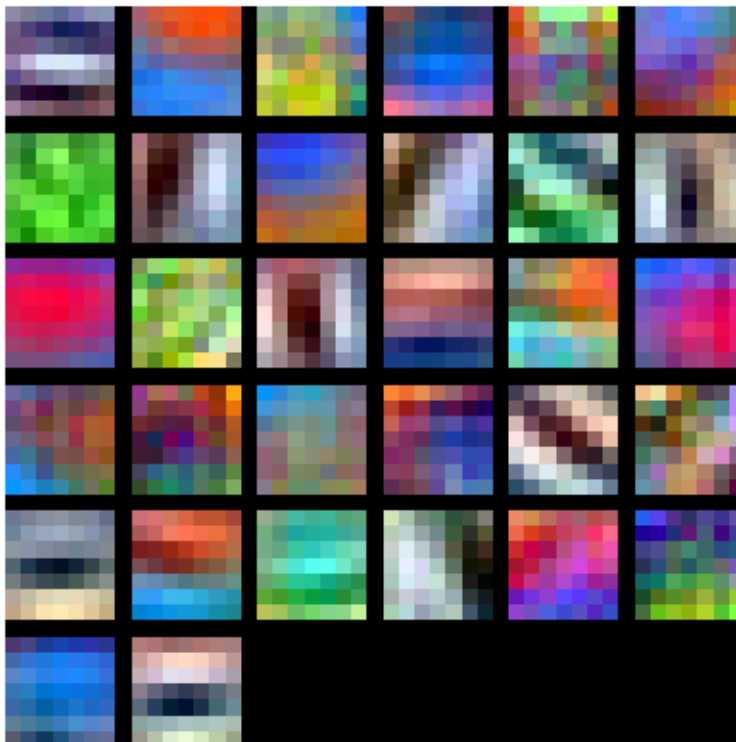
Full data validation accuracy: 0.499

8.5 Visualize Filters

You can visualize the first-layer convolutional filters from the trained network by running the following:

```
[ ]: from cse493g1.vis_utils import visualize_grid

grid = visualize_grid(model.params['W1'].transpose(0, 2, 3, 1))
plt.imshow(grid.astype('uint8'))
plt.axis('off')
plt.gcf().set_size_inches(5, 5)
plt.show()
```



9 EXTRA CREDIT: Spatial Batch Normalization

We already saw that batch normalization is a very useful technique for training deep fully connected networks. As proposed in the original paper (link in `BatchNormalization.ipynb`), batch normalization can also be used for convolutional networks, but we need to tweak it a bit; the modification will be called “spatial batch normalization.”

Normally, batch-normalization accepts inputs of shape (N, D) and produces outputs of shape (N, D) , where we normalize across the minibatch dimension N . For data coming from convolutional layers, batch normalization needs to accept inputs of shape (N, C, H, W) and produce outputs of shape (N, C, H, W) where the N dimension gives the minibatch size and the (H, W) dimensions give the spatial size of the feature map.

If the feature map was produced using convolutions, then we expect every feature channel’s statistics e.g. mean, variance to be relatively consistent both between different images, and different locations within the same image – after all, every feature channel is produced by the same convolutional filter! Therefore, spatial batch normalization computes a mean and variance for each of the C feature channels by computing statistics over the minibatch dimension N as well the spatial dimensions H and W .

[1] Sergey Ioffe and Christian Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”, ICML 2015.

10 EXTRA CREDIT: Spatial Batch Normalization: Forward Pass

In the file `cse493g1/layers.py`, implement the forward pass for spatial batch normalization in the function `spatial_batchnorm_forward`. Check your implementation by running the following:

```
[ ]: np.random.seed(493)

# Check the training-time forward pass by checking means and variances
# of features both before and after spatial batch normalization.
N, C, H, W = 2, 3, 4, 5
x = 4 * np.random.randn(N, C, H, W) + 10

print('Before spatial batch normalization:')
print('  shape: ', x.shape)
print('  means: ', x.mean(axis=(0, 2, 3)))
print('  stds: ', x.std(axis=(0, 2, 3)))

# Means should be close to zero and stds close to one
gamma, beta = np.ones(C), np.zeros(C)
bn_param = {'mode': 'train'}
out, _ = spatial_batchnorm_forward(x, gamma, beta, bn_param)
print('After spatial batch normalization:')
print('  shape: ', out.shape)
print('  means: ', out.mean(axis=(0, 2, 3)))
print('  stds: ', out.std(axis=(0, 2, 3)))

# Means should be close to beta and stds close to gamma
gamma, beta = np.asarray([3, 4, 5]), np.asarray([6, 7, 8])
out, _ = spatial_batchnorm_forward(x, gamma, beta, bn_param)
print('After spatial batch normalization (nontrivial gamma, beta):')
print('  shape: ', out.shape)
print('  means: ', out.mean(axis=(0, 2, 3)))
print('  stds: ', out.std(axis=(0, 2, 3)))
```

Before spatial batch normalization:

```
shape: (2, 3, 4, 5)
means: [9.2630585  9.90177254 9.94773619]
stds:  [3.57082624 3.61057967 4.35995843]
```

After spatial batch normalization:

```
shape: (2, 3, 4, 5)
means: [-1.26842981e-15 -6.88338275e-16 -1.22124533e-16]
stds:  [0.99999961 0.99999962 0.99999974]
```

After spatial batch normalization (nontrivial gamma, beta):

```
shape: (2, 3, 4, 5)
means: [6. 7. 8.]
stds:  [2.99999882 3.99999847 4.99999868]
```

```
[ ]: np.random.seed(493)

# Check the test-time forward pass by running the training-time
# forward pass many times to warm up the running averages, and then
# checking the means and variances of activations after a test-time
# forward pass.
N, C, H, W = 10, 4, 11, 12

bn_param = {'mode': 'train'}
gamma = np.ones(C)
beta = np.zeros(C)
for t in range(50):
    x = 2.3 * np.random.randn(N, C, H, W) + 13
    spatial_batchnorm_forward(x, gamma, beta, bn_param)
bn_param['mode'] = 'test'
x = 2.3 * np.random.randn(N, C, H, W) + 13
a_norm, _ = spatial_batchnorm_forward(x, gamma, beta, bn_param)

# Means should be close to zero and stds close to one, but will be
# noisier than training-time forward passes.
print('After spatial batch normalization (test-time):')
print('  means: ', a_norm.mean(axis=(0, 2, 3)))
print('  stds: ', a_norm.std(axis=(0, 2, 3)))
```

```
After spatial batch normalization (test-time):
means: [ 0.04646792  0.0557893  0.02508963 -0.01599525]
stds:  [0.97801211 0.99502029 1.0071852  1.00599827]
```

11 EXTRA CREDIT: Spatial Batch Normalization: Backward Pass

In the file `cse493g1/layers.py`, implement the backward pass for spatial batch normalization in the function `spatial_batchnorm_backward`. Run the following to check your implementation using a numeric gradient check:

```
[ ]: np.random.seed(493)
N, C, H, W = 2, 3, 4, 5
x = 5 * np.random.randn(N, C, H, W) + 12
gamma = np.random.randn(C)
beta = np.random.randn(C)
dout = np.random.randn(N, C, H, W)

bn_param = {'mode': 'train'}
fx = lambda x: spatial_batchnorm_forward(x, gamma, beta, bn_param)[0]
fg = lambda a: spatial_batchnorm_forward(x, gamma, beta, bn_param)[0]
fb = lambda b: spatial_batchnorm_forward(x, gamma, beta, bn_param)[0]
```

```

dx_num = eval_numerical_gradient_array(fx, x, dout)
da_num = eval_numerical_gradient_array(fg, gamma, dout)
db_num = eval_numerical_gradient_array(fb, beta, dout)

#You should expect errors of magnitudes between 1e-12~1e-06
_, cache = spatial_batchnorm_forward(x, gamma, beta, bn_param)
dx, dgamma, dbeta = spatial_batchnorm_backward(dout, cache)
print('dx error: ', rel_error(dx_num, dx))
print('dgamma error: ', rel_error(da_num, dgamma))
print('dbeta error: ', rel_error(db_num, dbeta))

```

```

dx error:  1.0642080342276615e-07
dgamma error:  8.741765051571405e-12
dbeta error:  3.2761583425136353e-12

```

12 EXTRA CREDIT: Spatial Group Normalization

In the previous notebook, we mentioned that Layer Normalization is an alternative normalization technique that mitigates the batch size limitations of Batch Normalization. However, as the authors of [2] observed, Layer Normalization does not perform as well as Batch Normalization when used with Convolutional Layers:

With fully connected layers, all the hidden units in a layer tend to make similar contributions to the final prediction, and re-centering and rescaling the summed inputs to a layer works well. However, the assumption of similar contributions is no longer true for convolutional neural networks. The large number of the hidden units whose receptive fields lie near the boundary of the image are rarely turned on and thus have very different statistics from the rest of the hidden units within the same layer.

The authors of [3] propose an intermediary technique. In contrast to Layer Normalization, where you normalize over the entire feature per-datapoint, they suggest a consistent splitting of each per-datapoint feature into G groups and a per-group per-datapoint normalization instead.

Visual comparison of the normalization techniques discussed so far (image edited from [3])

Even though an assumption of equal contribution is still being made within each group, the authors hypothesize that this is not as problematic, as innate grouping arises within features for visual recognition. One example they use to illustrate this is that many high-performance handcrafted features in traditional computer vision have terms that are explicitly grouped together. Take for example Histogram of Oriented Gradients [4] – after computing histograms per spatially local block, each per-block histogram is normalized before being concatenated together to form the final feature vector.

You will now implement Group Normalization.

[2] Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton. “Layer Normalization.” *stat* 1050 (2016): 21.

[3] Wu, Yuxin, and Kaiming He. “Group Normalization.” *arXiv preprint arXiv:1803.08494* (2018).

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In Computer Vision and Pattern Recognition (CVPR), 2005.

13 EXTRA CREDIT: Spatial Group Normalization: Forward Pass

In the file `cse493g1/layers.py`, implement the forward pass for group normalization in the function `spatial_groupnorm_forward`. Check your implementation by running the following:

```
[ ]: np.random.seed(493)

# Check the training-time forward pass by checking means and variances
# of features both before and after spatial batch normalization.
N, C, H, W = 2, 6, 4, 5
G = 2
x = 4 * np.random.randn(N, C, H, W) + 10
x_g = x.reshape((N*G,-1))
print('Before spatial group normalization:')
print('  shape: ', x.shape)
print('  means: ', x_g.mean(axis=1))
print('  stds: ', x_g.std(axis=1))

# Means should be close to zero and stds close to one
gamma, beta = np.ones((1,C,1,1)), np.zeros((1,C,1,1))
bn_param = {'mode': 'train'}

out, _ = spatial_groupnorm_forward(x, gamma, beta, G, bn_param)
out_g = out.reshape((N*G,-1))
print('After spatial group normalization:')
print('  shape: ', out.shape)
print('  means: ', out_g.mean(axis=1))
print('  stds: ', out_g.std(axis=1))
```

Before spatial group normalization:

```
shape: (2, 6, 4, 5)
means: [10.20725758  9.20112057 10.3034963   9.39619814]
stds:  [3.83224573  3.85580764  4.51500812  3.82824937]
```

After spatial group normalization:

```
shape: (2, 6, 4, 5)
means: [ 3.76550643e-16  0.00000000e+00 -2.85839061e-16 -5.92118946e-17]
stds:  [0.99999966  0.99999966  0.99999975  0.99999966]
```

14 EXTRA CREDIT: Spatial Group Normalization: Backward Pass

In the file `cse493g1/layers.py`, implement the backward pass for spatial batch normalization in the function `spatial_groupnorm_backward`. Run the following to check your implementation

using a numeric gradient check:

```
[ ]: np.random.seed(493)
N, C, H, W = 2, 6, 4, 5
G = 2
x = 5 * np.random.randn(N, C, H, W) + 12
gamma = np.random.randn(1,C,1,1)
beta = np.random.randn(1,C,1,1)
dout = np.random.randn(N, C, H, W)

gn_param = {}
fx = lambda x: spatial_groupnorm_forward(x, gamma, beta, G, gn_param)[0]
fg = lambda a: spatial_groupnorm_forward(x, gamma, beta, G, gn_param)[0]
fb = lambda b: spatial_groupnorm_forward(x, gamma, beta, G, gn_param)[0]

dx_num = eval_numerical_gradient_array(fx, x, dout)
da_num = eval_numerical_gradient_array(fg, gamma, dout)
db_num = eval_numerical_gradient_array(fb, beta, dout)

_, cache = spatial_groupnorm_forward(x, gamma, beta, G, gn_param)
dx, dgamma, dbeta = spatial_groupnorm_backward(dout, cache)

# You should expect errors of magnitudes between 1e-12 and 1e-07.
print('dx error: ', rel_error(dx_num, dx))
print('dgamma error: ', rel_error(da_num, dgamma))
print('dbeta error: ', rel_error(db_num, dbeta))
```

```
dx error:  6.671661060989952e-08
dgamma error:  4.422382985907405e-12
dbeta error:  3.2755312644649976e-12
```

```
[36]: np.random.seed(493)
N, C, H, W = 2, 6, 4, 5
G = 2
x = 5 * np.random.randn(N, C, H, W) + 12
gamma = np.random.randn(1,C,1,1)
beta = np.random.randn(1,C,1,1)
dout = np.random.randn(N, C, H, W)

gn_param = {}
fx = lambda x: spatial_groupnorm_forward(x, gamma, beta, G, gn_param)[0]
fg = lambda a: spatial_groupnorm_forward(x, gamma, beta, G, gn_param)[0]
fb = lambda b: spatial_groupnorm_forward(x, gamma, beta, G, gn_param)[0]

dx_num = eval_numerical_gradient_array(fx, x, dout)
da_num = eval_numerical_gradient_array(fg, gamma, dout)
db_num = eval_numerical_gradient_array(fb, beta, dout)
```

```

_, cache = spatial_groupnorm_forward(x, gamma, beta, G, gn_param)

N, C, H, W = dout.shape
G = len(cache)

split_matrices = np.split(dout, G, axis = 1)
dx_list = [None] * G
dgamma_list = [None] * G
dbeta_list = [None] * G

for i, split_dout in enumerate(split_matrices):
    split_dout_reshaped = split_dout.reshape(N,-1)
    dx_list[i], dgamma_list[i], dbeta_list[i] = ␣
    ↪ layernorm_backward(split_dout_reshaped, cache[i])
    dx_list[i] = dx_list[i].reshape(N, C//G, H, W)

    #Explicitly calculating dgamma and dbeta to avoid drama with dimensions
    #cache[i][1] is xcap for the chosen group
    dgamma_list[i] = np.sum(split_dout * cache[i][1].reshape(N,C//G,H,W), axis = ␣
    ↪ (0,2,3), keepdims= True)
    dbeta_list[i] = np.sum(split_dout, axis = (0,2,3), keepdims= True)

```

```
[38]: split_dout * cache[i][1]
```

```

-----
ValueError                                Traceback (most recent call last)
<ipython-input-38-c491740f8bcf> in <cell line: 1>()
----> 1 split_dout * cache[i][1]

ValueError: operands could not be broadcast together with shapes (2,3,4,5) (2,6)

```