CNN_without_preprocess

December 10, 2015

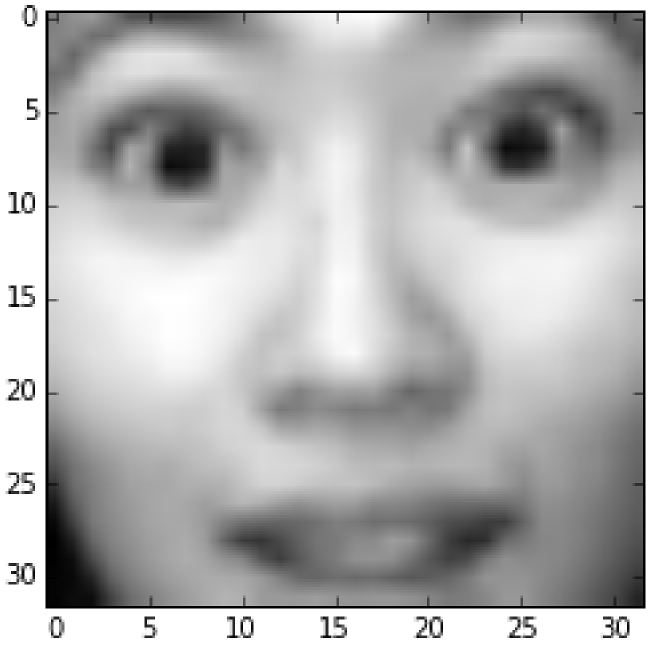
```
In [1]: import os
        import matplotlib.pyplot as plt
        %pylab inline
        import numpy as np
        from lasagne.layers import DenseLayer
        from lasagne.layers import InputLayer
        from lasagne.layers import DropoutLayer
        from lasagne.layers import Conv2DLayer
        from lasagne.layers import MaxPool2DLayer
        from lasagne.nonlinearities import softmax
        from lasagne.updates import adam
        from lasagne.layers import get_all_params
        from nolearn.lasagne import NeuralNet
        from nolearn.lasagne import TrainSplit
        from nolearn.lasagne import objective
```

```
Populating the interactive namespace from numpy and matplotlib
```

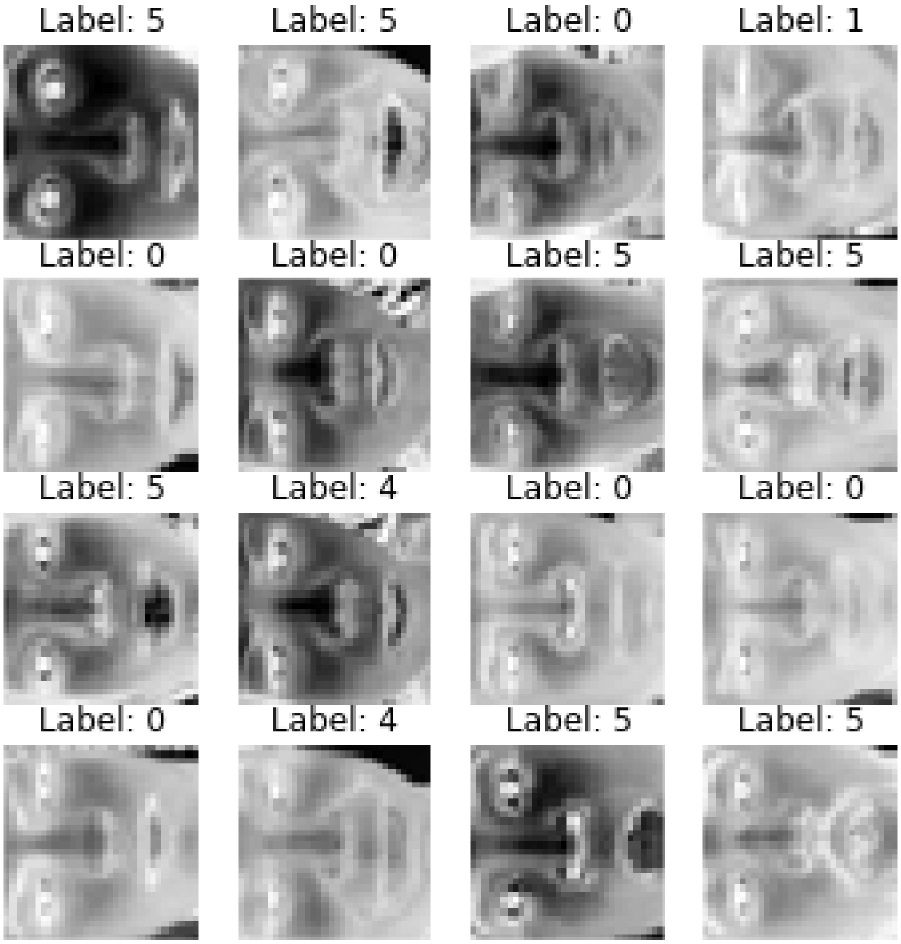
```
In [2]: import scipy.io
        train = scipy.io.loadmat('labeled_images.mat')
        print "Shape of tr_images is: ", train["tr_images"].shape
        (x_size, y_size, n_images) = train["tr_images"].shape
        X = np.reshape(np.swapaxes(train["tr_images"], 0, 2), (n_images, 1, x_size, y_size))
        y = train["tr_labels"].ravel()-1
        print X.shape
        print y.shape
        X = np.array(X).astype(np.float32)
        y = np.array(y).astype(np.int32)
        # Normalization
        X -= X.mean()
        X /= X.std()
        print X[0].shape
        print y.shape


        plt.imshow(np.swapaxes(np.reshape(X[0], (y_size, x_size)), 0, 1), cmap=pylab.gray())
        plt.show()
```

```
Shape of tr_images is:  (32, 32, 2925)
(2925, 1, 32, 32)
(2925,)
(1, 32, 32)
(2925,)
```



```
In [3]: # Show labels of the dataset
        figs, axes = plt.subplots(4, 4, figsize=(6, 6))
        for i in range(4):
            for j in range(4):
                axes[i, j].imshow(-X[i + 4 * j].reshape(32, 32), cmap='gray', interpolation='none')
                axes[i, j].set_xticks([])
                axes[i, j].set_yticks([])
                axes[i, j].set_title("Label: {}".format(y[i + 4 * j]))
                axes[i, j].axis('off')
```



```
In [43]: layers0 = [
             # layer dealing with the input data
             (InputLayer, {'shape': (None, X.shape[1], X.shape[2], X.shape[3])}),

             # first stage of our convolutional layers
             (Conv2DLayer, {'num_filters': 96, 'filter_size': 5}),
             (Conv2DLayer, {'num_filters': 96, 'filter_size': 3}),
             (Conv2DLayer, {'num_filters': 96, 'filter_size': 3}),
             (Conv2DLayer, {'num_filters': 96, 'filter_size': 3}),
             (Conv2DLayer, {'num_filters': 96, 'filter_size': 3}),
             (MaxPool2DLayer, {'pool_size': 2}),

             # second stage of our convolutional layers
             (Conv2DLayer, {'num_filters': 128, 'filter_size': 3}),
             (Conv2DLayer, {'num_filters': 128, 'filter_size': 3}),
             (Conv2DLayer, {'num_filters': 128, 'filter_size': 3}),
             (MaxPool2DLayer, {'pool_size': 2}),

             # two dense layers with dropout
             (DenseLayer, {'num_units': 64}),
             (DropoutLayer, {}),
             (DenseLayer, {'num_units': 64}),

             # the output layer
             (DenseLayer, {'num_units': 7, 'nonlinearity': softmax}),
         ]

         layers1 = [
             # layer dealing with the input data
             (InputLayer, {'shape': (None, X.shape[1], X.shape[2], X.shape[3])}),

             # first stage of our convolutional layers
             (Conv2DLayer, {'num_filters': 48, 'filter_size': 5}),
             (Conv2DLayer, {'num_filters': 48, 'filter_size': 3}),
             (Conv2DLayer, {'num_filters': 48, 'filter_size': 3}),
             (MaxPool2DLayer, {'pool_size': 2}),

             # second stage of our convolutional layers
             (Conv2DLayer, {'num_filters': 64, 'filter_size': 5}),
             (Conv2DLayer, {'num_filters': 64, 'filter_size': 3}),
             (MaxPool2DLayer, {'pool_size': 2}),

             # two dense layers with dropout
             (DenseLayer, {'num_units': 32}),
             (DropoutLayer, {}),
             (DenseLayer, {'num_units': 32}),

             # the output layer
             (DenseLayer, {'num_units': 7, 'nonlinearity': softmax}),
         ]

         layers2 = [
             (InputLayer, {'shape': (None, X.shape[1], X.shape[2], X.shape[3])}),

             (Conv2DLayer, {'num_filters': 32, 'filter_size': (3, 3)}),
             (MaxPool2DLayer, {'pool_size': (2, 2)}),

             (Conv2DLayer, {'num_filters': 64, 'filter_size': (3, 3)}),
             (Conv2DLayer, {'num_filters': 64, 'filter_size': (3, 3)}),
             (MaxPool2DLayer, {'pool_size': (2, 2)}),

             (Conv2DLayer, {'num_filters': 96, 'filter_size': (3, 3)}),
             (MaxPool2DLayer, {'pool_size': (2, 2)}),

             (DenseLayer, {'num_units': 64}),
             (DropoutLayer, {}),
             (DenseLayer, {'num_units': 64}),

             (DenseLayer, {'num_units': 7, 'nonlinearity': softmax}),
         ]

         layers3 = [
             (InputLayer, {'shape': (None, X.shape[1], X.shape[2], X.shape[3])}),
             (Conv2DLayer, {'num_filters': 96, 'filter_size': (5, 5)}),
             (MaxPool2DLayer, {'pool_size': (2, 2)}),
             (Conv2DLayer, {'num_filters': 64, 'filter_size': (5, 5)}),
             (MaxPool2DLayer, {'pool_size': (2, 2)}),
             (DenseLayer, {'num_units': 64}),
             (DenseLayer, {'num_units': 7, 'nonlinearity': softmax}),
         ]

         layers4 = [
             (InputLayer, {'shape': (None, X.shape[1], X.shape[2], X.shape[3])}),
             (Conv2DLayer, {'num_filters': 14, 'filter_size': (5, 5)}),
             (MaxPool2DLayer, {'pool_size': (2, 2)}),
             (Conv2DLayer, {'num_filters': 14, 'filter_size': (11, 11)}),
             (MaxPool2DLayer, {'pool_size': (2, 2)}),
             (DenseLayer, {'num_units': 32}),
             (DropoutLayer, {}),
             (DenseLayer, {'num_units': 32}),
             (DenseLayer, {'num_units': 7, 'nonlinearity': softmax}),
         ]

         layers5 = [
             (InputLayer, {'shape': (None, X.shape[1], X.shape[2], X.shape[3])}),
             (DenseLayer, {'num_units': 100}),
             (DenseLayer, {'num_units': 20}),
             (DenseLayer, {'num_units': 7, 'nonlinearity': softmax}),
         ]
```

```
In [44]: def regularization_objective(layers, lambda1=0., lambda2=0., *args, **kwargs):
             # default loss
             losses = objective(layers, *args, **kwargs)
             # get the layers' weights, but only those that should be regularized
             # (i.e. not the biases)
             weights = get_all_params(layers[-1], regularizable=True)
             # sum of absolute weights for L1
             sum_abs_weights = sum([abs(w).sum() for w in weights])
             # sum of squared weights for L2
             sum_squared_weights = sum([(w ** 2).sum() for w in weights])
             # add weights to regular loss
             losses += lambda1 * sum_abs_weights + lambda2 * sum_squared_weights
             return losses
```

```
In [45]: net1 = NeuralNet(
             layers=layers4,
             max_epochs=400,

             update=adam,
             update_learning_rate=0.0003,

             objective=regularization_objective,
             objective_lambda2=0.0025,

             train_split=TrainSplit(eval_size=0.005),
             verbose=4,
         )
```

```
In [46]: net1.fit(X, y)
```

```
# Neural Network with 27205 learnable parameters

## Layer information

name        size        total    cap.Y    cap.X    cov.Y    cov.X    filter Y    filter X    field Y   
----------  --------  -------  -------  -------  -------  -------  ----------  ----------  ---------  -
input0      1x32x32      1024   100.00   100.00   100.00   100.00          32          32         32
conv2d1     14x28x28    10976   100.00   100.00    15.62    15.62           5           5          5
maxpool2d2  14x14x14     2744   100.00   100.00    15.62    15.62           5           5          5
conv2d3     14x4x4        224    88.00    88.00    78.12    78.12          22          22         25
maxpool2d4  14x2x2         56    88.00    88.00    78.12    78.12          22          22         25
dense5      32             32   100.00   100.00   100.00   100.00          32          32         32
dropout6    32             32   100.00   100.00   100.00   100.00          32          32         32
dense7      32             32   100.00   100.00   100.00   100.00          32          32         32
dense8      7               7   100.00   100.00   100.00   100.00          32          32         32

Explanation
    X, Y:    image dimensions
    cap.:    learning capacity
    cov.:    coverage of image
    magenta: capacity too low (<1/6)
    cyan:    image coverage too high (>100%)
    red:     capacity too low and coverage too high


  epoch    train loss    valid loss    train/val    valid acc  dur
-------  ------------  ------------  -----------  -----------  -----
      1       2.19820       2.16019      1.01759      0.16667  8.22s
      2       2.15592       2.13053      1.01192      0.27778  8.61s
      3       2.13207       2.10978      1.01056      0.16667  9.22s
      4       2.10675       2.07756      1.01405      0.27778  9.93s
      5       2.09024       2.08239      1.00377      0.27778  9.44s
      6       2.07430       2.02401      1.02485      0.27778  8.27s
      7       2.03045       2.02117      1.00459      0.33333  8.66s
      8       2.01804       1.96153      1.02881      0.44444  9.25s
      9       1.97238       1.95389      1.00946      0.33333  10.98s
     10       1.94025       1.90566      1.01816      0.33333  9.55s
     11       1.88858       1.88571      1.00152      0.33333  8.46s
     12       1.85812       1.83472      1.01275      0.33333  8.24s
     13       1.80618       1.80991      0.99794      0.33333  8.27s
     14       1.77533       1.71711      1.03391      0.44444  8.16s
     15       1.73105       1.69035      1.02408      0.44444  8.01s
     16       1.68908       1.66910      1.01196      0.44444  8.02s
     17       1.66170       1.60432      1.03576      0.50000  7.88s
     18       1.63672       1.59940      1.02334      0.44444  8.01s
     19       1.60816       1.54364      1.04180      0.55556  7.70s
     20       1.58574       1.55594      1.01915      0.55556  7.89s
     21       1.55536       1.52566      1.01947      0.55556  7.75s
     22       1.51485       1.51099      1.00255      0.55556  7.76s
     23       1.50940       1.48630      1.01554      0.61111  7.84s
     24       1.48521       1.48626      0.99930      0.55556  7.78s
     25       1.45762       1.45078      1.00471      0.55556  7.75s
     26       1.44827       1.45042      0.99852      0.55556  8.06s
     27       1.43102       1.48331      0.96475      0.55556  7.76s
```

28	1.40994	1.45446	0.96939	0.55556	7.88s
29	1.38623	1.46662	0.94518	0.55556	7.76s
30	1.36298	1.43979	0.94665	0.55556	7.68s
31	1.38064	1.45388	0.94963	0.55556	7.71s
32	1.32844	1.41416	0.93939	0.55556	9.90s
33	1.32563	1.41635	0.93595	0.55556	8.10s
34	1.30796	1.41295	0.92570	0.55556	9.26s
35	1.30296	1.40105	0.92998	0.55556	9.71s
36	1.28888	1.48789	0.86625	0.55556	9.52s
37	1.28586	1.41749	0.90714	0.55556	9.04s
38	1.26826	1.43034	0.88668	0.50000	9.23s
39	1.23855	1.40687	0.88036	0.50000	9.14s
40	1.24152	1.44427	0.85962	0.55556	8.76s
41	1.23172	1.44915	0.84996	0.61111	14.38s
42	1.23123	1.44015	0.85493	0.55556	10.18s
43	1.21420	1.36780	0.88770	0.55556	9.55s
44	1.21570	1.41043	0.86194	0.61111	8.43s
45	1.20009	1.36742	0.87763	0.55556	8.39s
46	1.18752	1.35579	0.87589	0.61111	8.02s
47	1.20435	1.35494	0.88886	0.55556	8.31s
48	1.18842	1.33907	0.88750	0.55556	8.17s
49	1.18369	1.41513	0.83645	0.50000	8.36s
50	1.16726	1.37597	0.84832	0.50000	8.05s
51	1.15609	1.39911	0.82631	0.50000	8.07s
52	1.14306	1.39311	0.82051	0.55556	7.89s
53	1.13591	1.43439	0.79191	0.50000	7.75s
54	1.15043	1.46007	0.78793	0.55556	7.89s
55	1.13216	1.42863	0.79248	0.55556	8.15s
56	1.09843	1.44413	0.76062	0.55556	8.10s
57	1.12782	1.41309	0.79812	0.55556	7.89s
58	1.11660	1.39306	0.80155	0.55556	7.84s
59	1.10782	1.38615	0.79920	0.55556	7.85s
60	1.10758	1.44731	0.76527	0.55556	7.99s
61	1.08780	1.32830	0.81895	0.55556	7.99s
62	1.08298	1.36979	0.79062	0.55556	7.83s
63	1.08189	1.35756	0.79693	0.55556	8.14s
64	1.06791	1.34865	0.79184	0.55556	8.06s
65	1.07493	1.42180	0.75604	0.55556	8.10s
66	1.07329	1.39809	0.76769	0.55556	8.15s
67	1.07570	1.39973	0.76850	0.55556	7.99s
68	1.05061	1.39259	0.75443	0.55556	8.06s
69	1.03725	1.42297	0.72893	0.55556	8.52s
70	1.05082	1.39814	0.75158	0.55556	8.04s
71	1.02987	1.44228	0.71406	0.55556	8.30s
72	1.04445	1.43153	0.72960	0.55556	8.25s
73	1.03817	1.41544	0.73346	0.55556	8.39s
74	1.01949	1.50302	0.67829	0.55556	8.15s
75	1.00439	1.48164	0.67789	0.55556	8.07s
76	0.99348	1.46152	0.67976	0.55556	8.08s
77	1.00385	1.41404	0.70992	0.55556	8.09s
78	1.01830	1.36582	0.74556	0.55556	8.07s
79	1.01741	1.45219	0.70061	0.55556	8.33s
80	1.00581	1.42013	0.70825	0.55556	7.98s
81	1.00820	1.40671	0.71670	0.55556	7.96s

82	0.96423	1.48363	0.64992	0.55556	7.95s
83	0.96486	1.41804	0.68042	0.55556	7.89s
84	0.97445	1.40989	0.69115	0.55556	7.95s
85	0.96555	1.46150	0.66066	0.55556	8.01s
86	0.96620	1.48238	0.65179	0.55556	8.11s
87	0.95978	1.50425	0.63804	0.55556	8.13s
88	0.97880	1.40914	0.69461	0.55556	8.06s
89	0.95322	1.42975	0.66670	0.55556	8.05s
90	0.95723	1.52790	0.62650	0.55556	7.83s
91	0.93723	1.49352	0.62753	0.55556	8.24s
92	0.93957	1.46053	0.64331	0.55556	7.88s
93	0.93984	1.41435	0.66450	0.55556	8.19s
94	0.94996	1.48466	0.63986	0.55556	9.63s
95	0.95666	1.44896	0.66024	0.55556	10.87s
96	0.94493	1.44392	0.65442	0.55556	9.63s
97	0.92390	1.45521	0.63489	0.55556	8.23s
98	0.91525	1.48907	0.61465	0.55556	7.90s
99	0.92213	1.54430	0.59712	0.55556	8.09s
100	0.93146	1.47603	0.63105	0.55556	9.27s
101	0.91226	1.48720	0.61341	0.55556	8.73s
102	0.90430	1.53111	0.59062	0.55556	10.26s
103	0.89795	1.50446	0.59686	0.55556	8.50s
104	0.91182	1.46185	0.62374	0.55556	8.19s
105	0.89536	1.47642	0.60644	0.55556	8.07s
106	0.89655	1.52654	0.58731	0.55556	8.37s
107	0.89209	1.49762	0.59567	0.55556	8.17s
108	0.89308	1.55334	0.57494	0.55556	8.12s
109	0.88453	1.50773	0.58666	0.55556	8.28s
110	0.87551	1.55064	0.56461	0.55556	8.19s
111	0.88357	1.64155	0.53826	0.55556	8.21s
112	0.86616	1.50163	0.57681	0.55556	8.65s
113	0.87526	1.55845	0.56162	0.55556	7.78s
114	0.87591	1.60296	0.54643	0.55556	7.98s
115	0.86243	1.54421	0.55849	0.55556	7.94s
116	0.87957	1.44915	0.60696	0.55556	8.12s
117	0.87077	1.44102	0.60427	0.55556	8.42s
118	0.86029	1.44426	0.59566	0.55556	7.97s
119	0.86408	1.47793	0.58466	0.55556	8.11s
120	0.85902	1.47530	0.58226	0.55556	7.98s
121	0.85879	1.51114	0.56831	0.55556	8.01s
122	0.84086	1.44109	0.58349	0.55556	8.63s
123	0.84334	1.40280	0.60119	0.55556	10.35s
124	0.85228	1.39232	0.61213	0.55556	8.84s
125	0.83057	1.44087	0.57643	0.55556	8.99s
126	0.84004	1.50270	0.55902	0.55556	8.39s
127	0.84452	1.36674	0.61791	0.55556	8.45s
128	0.83858	1.40307	0.59768	0.55556	8.47s
129	0.82761	1.46548	0.56473	0.55556	7.83s
130	0.81947	1.47882	0.55414	0.55556	8.11s
131	0.83744	1.43090	0.58525	0.55556	7.75s
132	0.82138	1.42058	0.57820	0.55556	8.00s
133	0.81968	1.35815	0.60353	0.55556	8.14s
134	0.81836	1.30734	0.62597	0.61111	7.81s
135	0.82070	1.37326	0.59763	0.55556	8.07s

136	0.80854	1.43456	0.56361	0.55556	8.10s
137	0.81280	1.33868	0.60717	0.55556	8.07s
138	0.81826	1.39139	0.58809	0.55556	8.18s
139	0.82404	1.32647	0.62123	0.55556	8.10s
140	0.81186	1.39746	0.58095	0.55556	8.01s
141	0.79522	1.55863	0.51021	0.55556	7.86s
142	0.80458	1.38594	0.58053	0.55556	7.80s
143	0.80121	1.40149	0.57169	0.55556	7.95s
144	0.80585	1.34017	0.60131	0.61111	7.90s
145	0.79231	1.40463	0.56407	0.55556	7.94s
146	0.80789	1.35090	0.59804	0.55556	7.93s
147	0.78967	1.37744	0.57329	0.61111	7.89s
148	0.80536	1.44784	0.55625	0.55556	7.92s
149	0.79396	1.42665	0.55652	0.55556	7.89s
150	0.79557	1.46860	0.54172	0.55556	7.88s
151	0.78070	1.47762	0.52835	0.55556	8.05s
152	0.79509	1.43431	0.55433	0.55556	7.98s
153	0.78932	1.58860	0.49686	0.55556	8.03s
154	0.80475	1.49296	0.53903	0.55556	8.09s
155	0.77399	1.38898	0.55724	0.55556	7.98s
156	0.78154	1.50279	0.52006	0.55556	8.05s
157	0.75942	1.50631	0.50416	0.55556	8.01s
158	0.76346	1.53359	0.49783	0.55556	7.78s
159	0.77363	1.50804	0.51300	0.55556	7.64s
160	0.75960	1.47309	0.51565	0.55556	7.75s
161	0.76981	1.56983	0.49038	0.55556	7.71s
162	0.76592	1.52900	0.50093	0.55556	7.95s
163	0.77073	1.55727	0.49493	0.55556	7.76s
164	0.76132	1.64158	0.46377	0.55556	7.82s
165	0.75301	1.65495	0.45500	0.55556	7.99s
166	0.75662	1.70702	0.44324	0.55556	7.75s
167	0.75758	1.64908	0.45940	0.55556	7.85s
168	0.74017	1.55867	0.47488	0.55556	7.88s
169	0.73713	1.49057	0.49453	0.55556	7.87s
170	0.75775	1.61018	0.47060	0.55556	8.14s
171	0.73657	1.49830	0.49160	0.55556	8.00s
172	0.73209	1.55352	0.47125	0.55556	8.04s
173	0.74166	1.41848	0.52285	0.55556	8.02s
174	0.73247	1.48393	0.49360	0.61111	7.78s
175	0.73201	1.36250	0.53726	0.55556	7.96s
176	0.74463	1.44351	0.51585	0.55556	7.70s
177	0.74603	1.42591	0.52320	0.50000	7.65s
178	0.73373	1.41780	0.51751	0.61111	7.78s
179	0.72618	1.33584	0.54361	0.50000	7.74s
180	0.72693	1.37609	0.52826	0.50000	7.83s
181	0.71862	1.46650	0.49002	0.50000	8.06s
182	0.70855	1.36752	0.51813	0.55556	7.97s
183	0.72226	1.51539	0.47662	0.50000	7.89s
184	0.70004	1.52733	0.45834	0.55556	8.05s
185	0.69361	1.54650	0.44850	0.50000	8.04s
186	0.71170	1.54975	0.45923	0.50000	7.99s
187	0.72359	1.45353	0.49781	0.55556	7.96s
188	0.71340	1.34702	0.52961	0.50000	8.04s
189	0.70531	1.38834	0.50802	0.55556	8.22s

190	0.71406	1.43713	0.49686	0.50000	7.96s
191	0.69524	1.40373	0.49528	0.50000	7.91s
192	0.70394	1.52274	0.46229	0.50000	8.01s
193	0.71563	1.32358	0.54068	0.61111	7.77s
194	0.71879	1.37464	0.52289	0.61111	8.01s
195	0.70561	1.43758	0.49083	0.50000	7.86s
196	0.71753	1.74446	0.41132	0.50000	8.15s
197	0.68875	1.61180	0.42732	0.50000	8.21s
198	0.69320	1.49843	0.46262	0.50000	8.04s
199	0.68290	1.63920	0.41660	0.50000	7.96s
200	0.69277	1.61466	0.42905	0.50000	8.08s
201	0.70301	1.57658	0.44591	0.50000	7.97s
202	0.69403	1.78106	0.38967	0.50000	7.85s
203	0.70343	1.65558	0.42488	0.55556	7.89s
204	0.68802	1.56523	0.43957	0.50000	7.75s
205	0.68765	1.64107	0.41903	0.50000	8.06s
206	0.68070	1.77345	0.38383	0.50000	7.78s
207	0.68594	1.66518	0.41193	0.55556	7.89s
208	0.67805	1.65582	0.40949	0.50000	7.84s
209	0.67074	1.73445	0.38671	0.55556	7.82s
210	0.66579	1.74991	0.38047	0.50000	7.96s
211	0.68645	1.75434	0.39129	0.50000	7.90s
212	0.66252	1.73564	0.38171	0.50000	7.96s
213	0.68220	1.75888	0.38786	0.50000	8.43s
214	0.67788	1.90104	0.35658	0.50000	8.10s
215	0.65544	1.77176	0.36994	0.50000	8.06s
216	0.66936	1.78656	0.37467	0.50000	8.03s
217	0.65665	1.75816	0.37349	0.50000	8.07s
218	0.67845	1.84192	0.36834	0.55556	8.18s
219	0.66494	1.91986	0.34635	0.50000	8.27s
220	0.65994	1.81821	0.36296	0.50000	8.25s
221	0.66088	1.84849	0.35752	0.50000	8.58s
222	0.65234	1.76474	0.36965	0.50000	8.08s
223	0.64950	2.14181	0.30325	0.50000	8.48s
224	0.67500	1.86855	0.36124	0.50000	8.64s
225	0.65972	1.95162	0.33804	0.50000	8.02s
226	0.65452	1.97529	0.33135	0.50000	8.00s
227	0.66365	1.85540	0.35768	0.50000	8.07s
228	0.68034	1.93125	0.35228	0.50000	8.40s
229	0.70279	1.95369	0.35973	0.50000	8.26s
230	0.70125	1.78807	0.39218	0.55556	8.22s
231	0.73765	1.76926	0.41693	0.55556	8.18s
232	0.71218	1.57708	0.45158	0.55556	8.12s
233	0.71977	1.28274	0.56112	0.66667	7.87s
234	0.70687	1.07449	0.65786	0.61111	7.80s
235	0.72409	0.98428	0.73566	0.66667	7.80s
236	0.76573	1.00692	0.76046	0.61111	7.87s
237	0.79776	1.27031	0.62801	0.61111	7.91s
238	0.75353	1.60093	0.47068	0.55556	7.86s
239	0.68760	1.84927	0.37182	0.50000	7.72s
240	0.67545	1.54785	0.43638	0.55556	7.91s
241	0.66187	1.38168	0.47903	0.66667	8.16s
242	0.64386	1.27916	0.50335	0.66667	8.13s
243	0.65394	1.18096	0.55373	0.66667	8.12s

244	0.64046	1.31945	0.48540	0.61111	8.15s
245	0.65364	1.38967	0.47035	0.66667	8.17s
246	0.64454	1.51654	0.42500	0.66667	8.18s
247	0.61750	1.60123	0.38564	0.55556	8.18s
248	0.62440	1.67212	0.37342	0.61111	8.31s
249	0.61448	1.55269	0.39576	0.66667	8.01s
250	0.62708	1.74056	0.36027	0.55556	7.68s
251	0.62276	1.60114	0.38895	0.66667	7.91s
252	0.60517	1.61413	0.37492	0.66667	8.30s
253	0.62585	1.42556	0.43902	0.66667	7.98s
254	0.61471	1.23583	0.49741	0.72222	8.02s
255	0.61394	1.26862	0.48394	0.66667	7.85s
256	0.61990	1.22366	0.50659	0.61111	7.78s
257	0.61507	1.43063	0.42993	0.61111	7.76s
258	0.61030	1.40036	0.43582	0.72222	7.78s
259	0.60451	1.42589	0.42395	0.66667	7.72s
260	0.60978	1.60608	0.37967	0.72222	7.89s
261	0.62833	1.62003	0.38785	0.66667	8.06s
262	0.61915	1.72265	0.35942	0.61111	7.86s
263	0.60644	1.62578	0.37301	0.66667	7.85s
264	0.61295	1.71047	0.35835	0.66667	8.02s
265	0.60189	1.69213	0.35570	0.61111	7.95s
266	0.59635	1.76162	0.33853	0.61111	7.93s
267	0.60020	1.70346	0.35234	0.61111	8.01s
268	0.58447	1.59386	0.36670	0.55556	8.10s
269	0.59681	1.54798	0.38554	0.66667	7.90s
270	0.57977	1.50189	0.38603	0.66667	7.80s
271	0.61393	1.49178	0.41154	0.55556	7.96s
272	0.60168	1.28223	0.46924	0.66667	8.06s
273	0.60276	1.38007	0.43676	0.66667	7.99s
274	0.58924	1.33996	0.43974	0.61111	7.97s
275	0.58857	1.38772	0.42413	0.61111	7.98s
276	0.58038	1.45139	0.39988	0.61111	8.05s
277	0.57779	1.43978	0.40130	0.61111	8.33s
278	0.57307	1.52714	0.37525	0.61111	8.22s
279	0.58084	1.59038	0.36522	0.61111	7.78s
280	0.57252	1.65014	0.34695	0.61111	7.70s
281	0.56248	1.64765	0.34138	0.61111	7.85s
282	0.57812	1.60657	0.35985	0.61111	7.73s
283	0.55163	1.73456	0.31802	0.61111	7.70s
284	0.58355	1.61399	0.36156	0.61111	7.88s
285	0.57220	1.79095	0.31950	0.61111	8.08s
286	0.57039	1.83052	0.31160	0.55556	7.97s
287	0.56829	1.67799	0.33867	0.61111	8.01s
288	0.56917	1.58570	0.35894	0.61111	7.85s
289	0.57028	1.78743	0.31905	0.61111	9.68s
290	0.57264	1.78671	0.32050	0.61111	9.99s
291	0.55771	1.57641	0.35378	0.66667	9.07s
292	0.56753	1.86764	0.30388	0.61111	9.15s
293	0.55303	1.58570	0.34876	0.61111	8.06s
294	0.55987	1.66217	0.33683	0.61111	8.22s
295	0.56201	1.56524	0.35906	0.61111	9.67s
296	0.55556	1.51827	0.36592	0.66667	10.38s
297	0.55928	1.64357	0.34029	0.61111	8.29s

298	0.55435	1.81798	0.30493	0.61111	8.09s
299	0.55548	1.71795	0.32334	0.61111	8.56s
300	0.55959	1.66398	0.33630	0.66667	8.05s
301	0.56144	1.78661	0.31425	0.55556	8.01s
302	0.56889	1.78067	0.31948	0.61111	7.93s
303	0.55271	1.62516	0.34009	0.61111	7.95s
304	0.53911	1.69613	0.31785	0.55556	8.01s
305	0.54903	1.63097	0.33663	0.61111	7.93s
306	0.55294	1.46857	0.37652	0.72222	7.81s
307	0.57035	1.86471	0.30586	0.61111	8.05s
308	0.53562	1.57477	0.34013	0.61111	7.75s
309	0.54615	1.54492	0.35352	0.61111	7.88s
310	0.53536	1.44513	0.37046	0.66667	7.78s
311	0.54106	1.69795	0.31866	0.61111	7.73s
312	0.55142	1.60625	0.34329	0.66667	7.73s
313	0.56857	1.45473	0.39084	0.61111	7.79s
314	0.54418	1.51465	0.35928	0.72222	7.84s
315	0.56776	1.68462	0.33702	0.61111	8.03s
316	0.54235	1.57154	0.34511	0.72222	7.76s
317	0.54181	1.52736	0.35473	0.55556	7.79s
318	0.54127	1.42444	0.37999	0.66667	7.76s
319	0.54995	1.57605	0.34894	0.55556	7.74s
320	0.54091	1.45897	0.37075	0.55556	7.76s
321	0.54236	1.42418	0.38083	0.72222	7.89s
322	0.54291	1.47633	0.36774	0.66667	7.84s
323	0.55153	1.76668	0.31218	0.55556	7.88s
324	0.54467	1.55831	0.34953	0.66667	7.91s
325	0.56535	1.39941	0.40400	0.61111	7.79s
326	0.55312	1.60302	0.34505	0.61111	7.85s
327	0.56716	1.62100	0.34988	0.61111	7.95s
328	0.55039	1.78705	0.30799	0.61111	7.71s
329	0.55892	1.70980	0.32689	0.61111	7.77s
330	0.55598	1.64908	0.33714	0.61111	7.78s
331	0.53352	1.65208	0.32294	0.66667	7.77s
332	0.54494	1.70349	0.31990	0.61111	8.00s
333	0.54933	1.93434	0.28399	0.55556	7.86s
334	0.53465	1.72996	0.30905	0.61111	7.79s
335	0.54468	1.69825	0.32073	0.61111	16.21s
336	0.52467	2.05010	0.25593	0.61111	9.72s
337	0.55113	1.99914	0.27568	0.55556	10.05s
338	0.53974	1.90053	0.28400	0.55556	8.64s
339	0.53878	1.96030	0.27485	0.55556	8.83s
340	0.52595	2.06641	0.25452	0.55556	8.35s
341	0.54196	2.01958	0.26836	0.55556	8.34s
342	0.54369	1.97273	0.27560	0.55556	8.90s
343	0.57889	1.56778	0.36924	0.55556	10.08s
344	0.60370	1.37588	0.43877	0.66667	9.66s
345	0.59703	1.10050	0.54250	0.72222	8.89s
346	0.55639	1.15430	0.48202	0.72222	10.22s
347	0.59522	1.17366	0.50714	0.66667	10.80s
348	0.58680	1.29953	0.45155	0.66667	8.76s
349	0.59669	1.39747	0.42698	0.61111	8.92s
350	0.58512	1.61168	0.36305	0.61111	8.48s
351	0.55556	1.74832	0.31777	0.61111	10.07s

```
352       0.53584       2.06527      0.25945      0.61111  10.35s
353       0.55360       1.99004      0.27818      0.55556  9.19s
354       0.56907       1.76980      0.32154      0.55556  9.96s
355       0.56713       1.35054      0.41993      0.55556  9.50s
356       0.53659       1.35030      0.39738      0.66667  8.20s
357       0.54420       1.36812      0.39777      0.66667  9.10s
358       0.55036       1.49883      0.36719      0.61111  8.23s
359       0.55686       1.74084      0.31988      0.61111  9.20s
360       0.53444       1.99799      0.26749      0.61111  8.62s
361       0.52272       1.86899      0.27968      0.61111  8.47s
362       0.51380       1.78182      0.28836      0.61111  8.37s
363       0.51998       1.75053      0.29704      0.61111  8.43s
364       0.52774       1.50838      0.34987      0.66667  8.26s
365       0.52833       1.27380      0.41477      0.66667  8.54s
366       0.51285       1.20751      0.42472      0.72222  8.55s
367       0.53154       1.31542      0.40409      0.72222  8.45s
368       0.53729       1.62488      0.33067      0.61111  8.28s
369       0.52548       1.73773      0.30239      0.61111  8.18s
370       0.50670       1.73534      0.29199      0.66667  8.13s
371       0.52296       1.67672      0.31190      0.61111  8.23s
372       0.51345       1.55175      0.33088      0.61111  8.12s
373       0.54104       1.80061      0.30048      0.61111  8.27s
374       0.50388       1.53817      0.32759      0.66667  8.33s
375       0.50595       1.31499      0.38476      0.66667  8.09s
376       0.51360       1.16607      0.44045      0.66667  8.22s
377       0.50904       1.22271      0.41632      0.55556  7.88s
378       0.51769       1.25915      0.41114      0.72222  8.22s
379       0.51683       1.44999      0.35644      0.72222  9.80s
380       0.52881       1.56989      0.33685      0.66667  9.22s
381       0.51231       1.63586      0.31317      0.66667  10.19s
382       0.51517       1.58113      0.32583      0.66667  8.83s
383       0.49368       1.47843      0.33392      0.66667  8.41s
384       0.52803       1.44782      0.36471      0.72222  8.07s
385       0.53807       1.33780      0.40221      0.61111  8.31s
386       0.55124       1.36473      0.40392      0.66667  8.08s
387       0.53065       1.28969      0.41146      0.61111  8.08s
388       0.55192       1.23866      0.44558      0.66667  8.00s
389       0.55685       1.39716      0.39856      0.66667  8.28s
390       0.55153       1.40568      0.39236      0.66667  8.33s
391       0.58123       1.59542      0.36431      0.66667  8.25s
392       0.58260       1.46764      0.39697      0.66667  8.37s
393       0.60664       1.68764      0.35946      0.66667  8.47s
394       0.60647       1.77455      0.34176      0.55556  8.39s
395       0.61471       1.94533      0.31599      0.61111  8.44s
396       0.59099       2.05297      0.28787      0.55556  9.71s
397       0.59406       1.84235      0.32245      0.61111  9.91s
398       0.57785       1.73547      0.33297      0.61111  9.49s
399       0.57234       1.63097      0.35092      0.61111  8.83s
400       0.60429       1.70867      0.35366      0.72222  8.36s
```

```
Out[46]: NeuralNet(X_tensor_type=None,
              batch_iterator_test=<nolearn.lasagne.base.BatchIterator object at 0x1077bf210>,
              batch_iterator_train=<nolearn.lasagne.base.BatchIterator object at 0x1077bf190>,
              custom_score=None,
              layers=[(<class 'lasagne.layers.input.InputLayer'>, {'shape': (None, 1, 32, 32)}), (<class
              loss=None, max_epochs=400, more_params={},
              objective=<function regularization_objective at 0x110df1c08>,
              objective_lambda2=0.0025,
              objective_loss_function=<function categorical_crossentropy at 0x1074b1758>,
              on_batch_finished=[],
              on_epoch_finished=[<nolearn.lasagne.handlers.PrintLog instance at 0x11c7e29e0>],
              on_training_finished=[],
              on_training_started=[<nolearn.lasagne.handlers.PrintLayerInfo instance at 0x11c7e2cf8>],
              regression=False,
              train_split=<nolearn.lasagne.base.TrainSplit object at 0x11c831250>,
              update=<function adam at 0x1074b9b18>, update_learning_rate=0.0003,
              use_label_encoder=False, verbose=4,
              y_tensor_type=TensorType(int32, vector))
```

```
In [47]: net1.save_params_to ('CNN_two_layers_small_filters')
```

```
In [48]: def classify_pub_test(classifier):
             pub_test = scipy.io.loadmat('public_test_images.mat')
             hid_test = scipy.io.loadmat('hidden_test_images.mat')

             (x, y, n_images) = pub_test["public_test_images"].shape
             test_img = np.reshape(np.swapaxes(pub_test["public_test_images"], 0, 2), (n_images, 1, x,
             test_img = np.array(test_img).astype(np.float32)
             test_img -= test_img.mean()
             test_img /= test_img.std()
             pub_res = list(classifier.predict(test_img)+1)

             (x, y, n_images) = hid_test["hidden_test_images"].shape
             test_img = np.reshape(np.swapaxes(hid_test["hidden_test_images"], 0, 2), (n_images, 1, x,
             test_img = np.array(test_img).astype(np.float32)
             test_img -= test_img.mean()
             test_img /= test_img.std()
             hid_res = list(classifier.predict(test_img)+1)
             return pub_res+hid_res
```

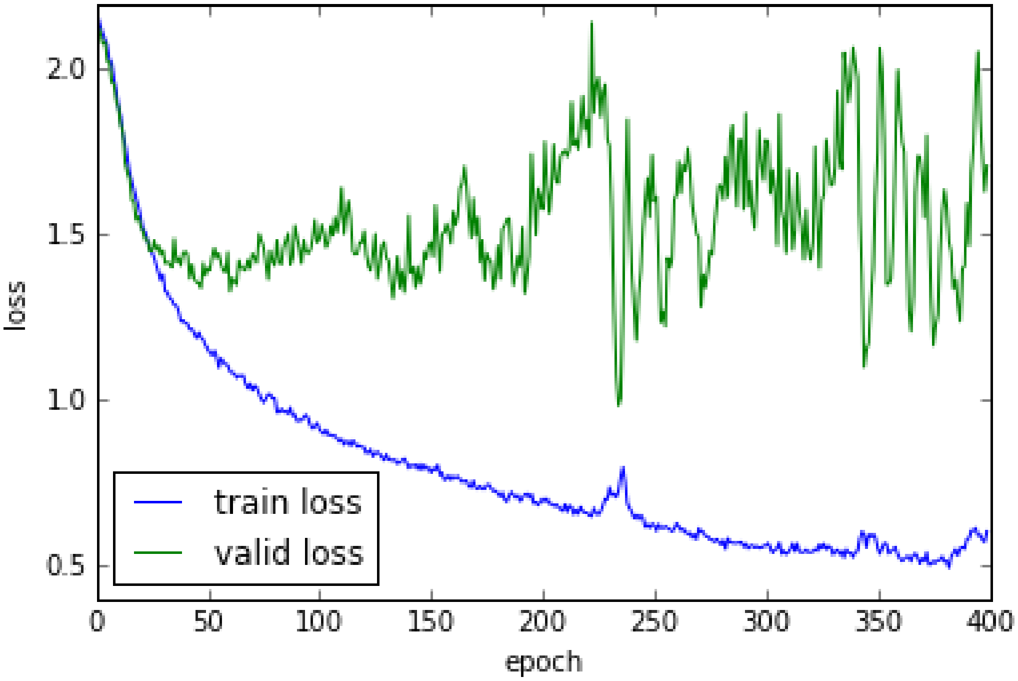
```
In [49]: classify_result = classify_pub_test(net1)
         cls_res_list = list(classify_result)
         print cls_res_list
         with open('cnn_with_normalization_small_filters.csv', 'w') as f:
             f.write('Id,Prediction\n')
             index = 1
             for pred in cls_res_list:
                 f.write('%d,%d\n'%(index, pred))
                 index += 1
             while index<=1253:
                 f.write('%d,0\n'%(index))
                 index+=1
```

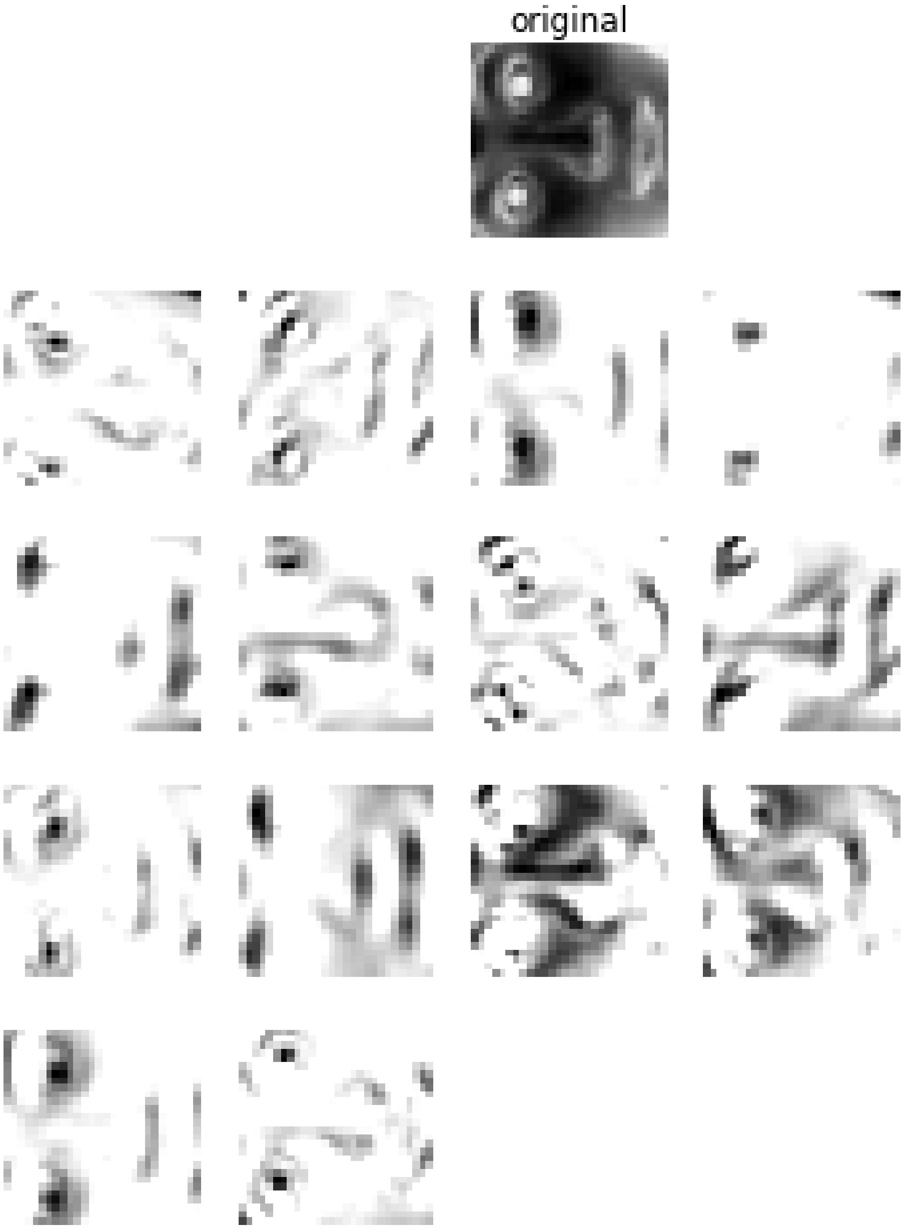
```
[7, 7, 4, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 4, 7, 4, 4, 7, 4, 7, 4, 7, 7, 4, 7, 4, 7, 4, 4, 4, 3,
```

```
In [50]: from nolearn.lasagne.visualize import plot_loss
         from nolearn.lasagne.visualize import plot_conv_weights
         from nolearn.lasagne.visualize import plot_conv_activity
         from nolearn.lasagne.visualize import plot_occlusion
```

```
plot_loss(net1)
plot_conv_activity(net1.layers_[1], X[0:1])
```

```
Out[50]: <module 'matplotlib.pyplot' from '/Users/zexuanwang/anaconda/lib/python2.7/site-packages/matpl
```





original

In [ ]:

In [ ]:

In [ ]:

In [ ]: