In [9]:

```python
# Reload our package (for use if modified).
import importlib
importlib.reload(load)
importlib.reload(normalize)
importlib.reload(process)
importlib.reload(batch)
importlib.reload(cnn)
importlib.reload(metrics)
```

Out[9]:

```
<module 'src.classification.metrics' from '../../src/classification/me
trics.py'>
```

In [1]:

```python
# Put the project's root directory on our PYTHONPATH.
import sys
sys.path.insert(0, "../..")
```

In [2]:

```python
# Import from our package.
from src.preprocessing import load, normalize, process, batch
from src.classification import cnn, metrics
```

In [3]:

```python
# Import from other libraries.
import tensorflow as tf
import random
import numpy as np
```

In [4]:

```python
# Define the architecture of our net as a list of layers.
layers = [
    cnn.ConvolutionalLayer(3, 5, 2, "VALID", "convlayer"),
    cnn.TransferLayer(lambda x, name: tf.sigmoid(x, name=name), "trans1layer"),
    cnn.FlatLayer("flatlayer"),
    cnn.FullLayer(15, "full1layer"),
    cnn.TransferLayer(lambda x, name: tf.sigmoid(x, name=name), "trans2layer"),
    cnn.FullLayer(2, "full2layer")
]
```

In [5]:

```
# Construct the net.
sess = tf.Session()
net = cnn.NeuralNet(
    layers,
    lambda labels, predictions: tf.losses.softmax_cross_entropy(
        labels,
        predictions,
        weights=1,
        label_smoothing=0,
        scope=None,
        loss_collection=tf.GraphKeys.LOSSES,
        reduction=tf.losses.Reduction.SUM_BY_NONZERO_WEIGHTS
    ),
    {
        "accuracy": metrics.accuracy
    },
    lambda: tf.train.GradientDescentOptimizer(0.005),
    "models/" + "4",
    "nn"
)
```

```
INFO:tensorflow:Using config: {'_tf_random_seed': 42114, '_session_con
fig': None, '_keep_checkpoint_every_n_hours': 1, '_model_dir': 'models
/4', '_keep_checkpoint_max': 10, '_log_step_count_steps': 1000, '_save
_summary_steps': 100, '_save_checkpoints_steps': None, '_save_checkpoi
nts_secs': 60}
```

In [6]:

```
# Load and preprocess training data.
train_images, train_masks = load.load_images_and_masks("data/train", ("1", "10"))
train_images = normalize.smdm_normalize(train_images, 51, "REFLECT")
train_patches, train_classes = process.patchify(train_images, train_masks, 51, 11)
train_classes = process.one_hot_encode(train_classes)
train_input_fn = batch.input_fn(train_patches, train_classes, 3000)
```

In [7]:

```
# Train the net for 5 minutes.
with tf.Session().as_default():
    net.train_duration_(train_input_fn, 300)
```

```
INFO:tensorflow:Create CheckpointSaverHook.
INFO:tensorflow:Restoring parameters from models/4/model.ckpt-115
INFO:tensorflow:Saving checkpoints for 116 into models/4/model.ckpt.
INFO:tensorflow:step = 116, loss = 0.161084
INFO:tensorflow:Loss for final step: 0.161084.
INFO:tensorflow:Create CheckpointSaverHook.
INFO:tensorflow:Restoring parameters from models/4/model.ckpt-116
INFO:tensorflow:Saving checkpoints for 117 into models/4/model.ckpt.
INFO:tensorflow:step = 117, loss = 0.164938
INFO:tensorflow:Loss for final step: 0.164938.
INFO:tensorflow:Create CheckpointSaverHook.
INFO:tensorflow:Restoring parameters from models/4/model.ckpt-117
INFO:tensorflow:Saving checkpoints for 118 into models/4/model.ckpt.
INFO:tensorflow:step = 118, loss = 0.160524
INFO:tensorflow:Loss for final step: 0.160524.
INFO:tensorflow:Create CheckpointSaverHook.
INFO:tensorflow:Restoring parameters from models/4/model.ckpt-118
INFO:tensorflow:Saving checkpoints for 119 into models/4/model.ckpt.
INFO:tensorflow:step = 119, loss = 0.158074
INFO:tensorflow:Loss for final step: 0.158074
```

In [11]:

```
# Load and preprocess test data.
test_images, test_masks = load.load_images_and_masks("data/test", ("7"))
test_images = normalize.smdm_normalize(test_images, 51, "REFLECT")
test_patches, test_classes = process.patchify(test_images, test_masks, 51, 11)
test_classes = process.one_hot_encode(test_classes)
test_input_fn = batch.input_fn(test_patches, test_classes, 3000, 1)
```

In [12]:

```
# Evaluate the net.
metric_dict = net.evaluate_(test_input_fn)
```

```
INFO:tensorflow:Starting evaluation at 2017-10-15-18:39:57
INFO:tensorflow:Restoring parameters from models/4/model.ckpt-243
INFO:tensorflow:Finished evaluation at 2017-10-15-18:39:59
INFO:tensorflow:Saving dict for global step 243: accuracy = 0.997496,
global_step = 243, loss = 0.0842894
```

In [13]:

```
# Examine the test results.
metric_dict
```

Out[13]:

```
{'accuracy': 0.99749601, 'global_step': 243, 'loss': 0.084289394}
```