In [1]:
```
# Configure matplotlib.
%matplotlib inline
```

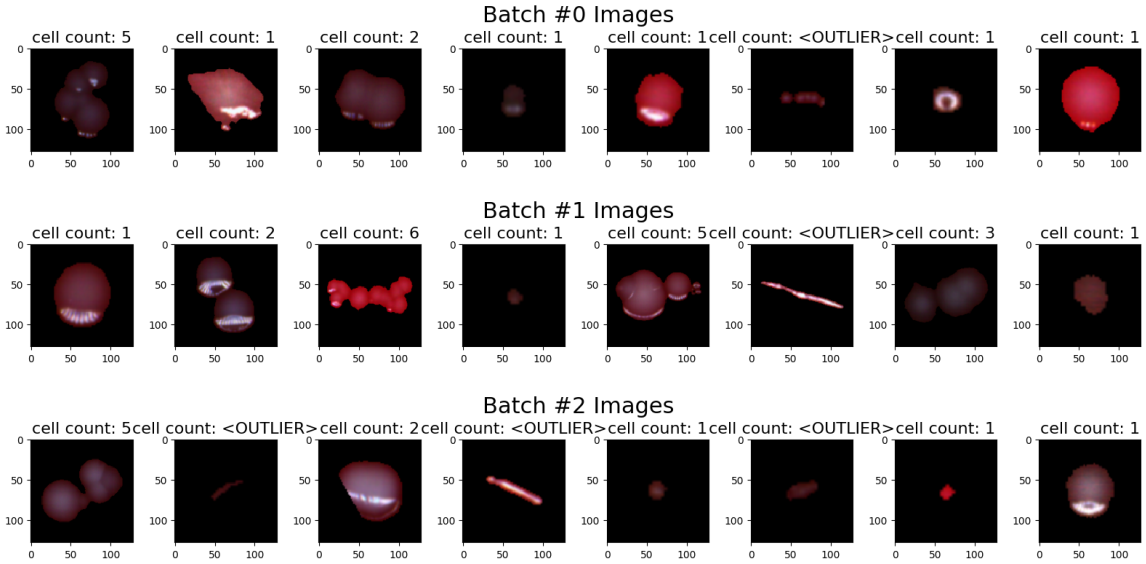
In [2]:
```
# Import our package.
import sys, importlib
sys.path.append("/home/ubuntu/cell_counting")

from src import dataset, visualization, preprocess, metric
from src.model import model
from src.model import neural_net
from src.model.segment_counting.convnet2 import convnet2
```

```
/home/ubuntu/anaconda3/envs/tensorflow_p36/lib/python3.6/importlib/_bootstrap.py
:219: RuntimeWarning: compiletime version 3.5 of module 'tensorflow.python.frame
work.fast_tensor_util' does not match runtime version 3.6
  return f(*args, **kwds)
```

In [3]:
```
# (if changes are made) Re-import our package.
for module in (dataset, visualization, preprocess, metric, model, neural_net, conv
net2):
    importlib.reload(module)
```

In [4]:
```
# Load the microbia_segments dataset.
def image_path_getter(example_metadata):
    return "/home/ubuntu/cell_counting/data/microbia_segments/raw/" + example_meta
data["Segment Relative Path"]
def mask_path_getter(example_metadata):
    return "/home/ubuntu/cell_counting/data/microbia_segments/raw/" + example_meta
data["Binary Segment Relative Path"]
def label_getter(example_metadata):
    return example_metadata["data"]["segment_type"]["data"]
microbia_segments = dataset.Dataset(256)
microbia_segments.load_images_masks_labels_from_json(
    "/home/ubuntu/cell_counting/data/microbia_segments/raw/enumeration_segments.js
on", image_path_getter,
    mask_path_getter, label_getter, (128, 128))
```

In [5]:
```
# Plot a few batches.
for batch in range(3):
    inputs, outputs = microbia_segments.get_batch(8)
    visualization.show_image_grid(inputs, 1, 8, 2.5, 16, "Batch #{0} Images".forma
t(batch),
        ["cell count: {0}".format(count + 1 if count != 7 else "<OUTLIER>") for co
unt in outputs])
```



In [6]:
```
# Make the labels one-hot.
def to_one_hot(examples):
    inputs, outputs = examples
    outputs = preprocess.one_hot_encode(outputs, 7)
    return inputs, outputs
microbia_segments.map_batch(to_one_hot)
```

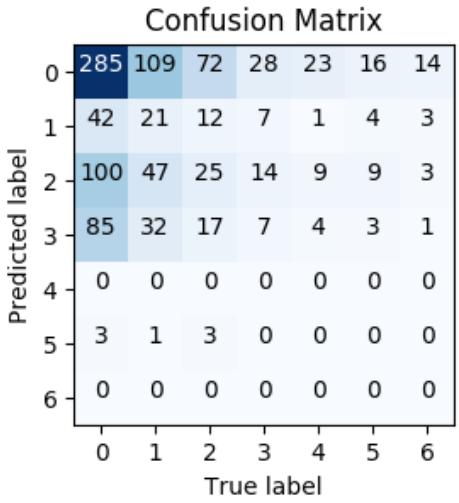
In [7]:
```
# Split the dataset.
train, test = microbia_segments.split(0.1)
```

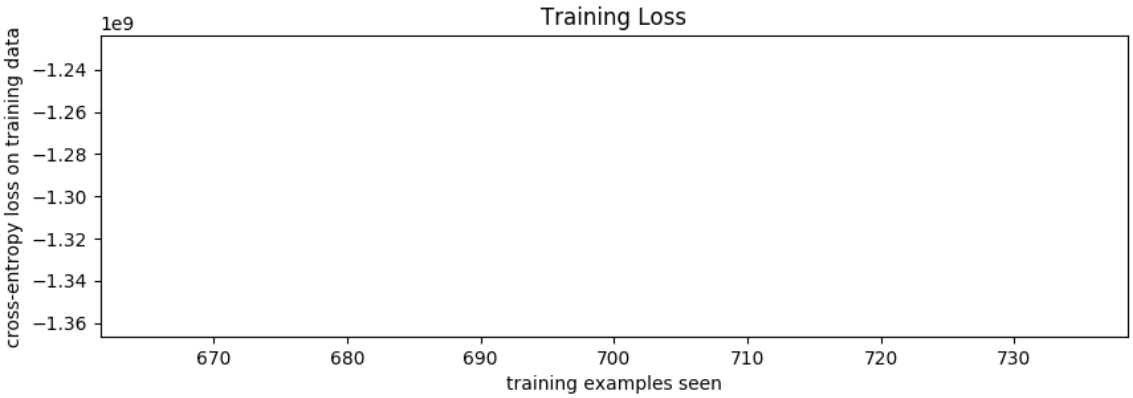
In [22]:
```
# Create the net.
import tensorflow as tf
net = convnet2.ConvNet2("saves/17-11-27-AM-09-43", 120)
```

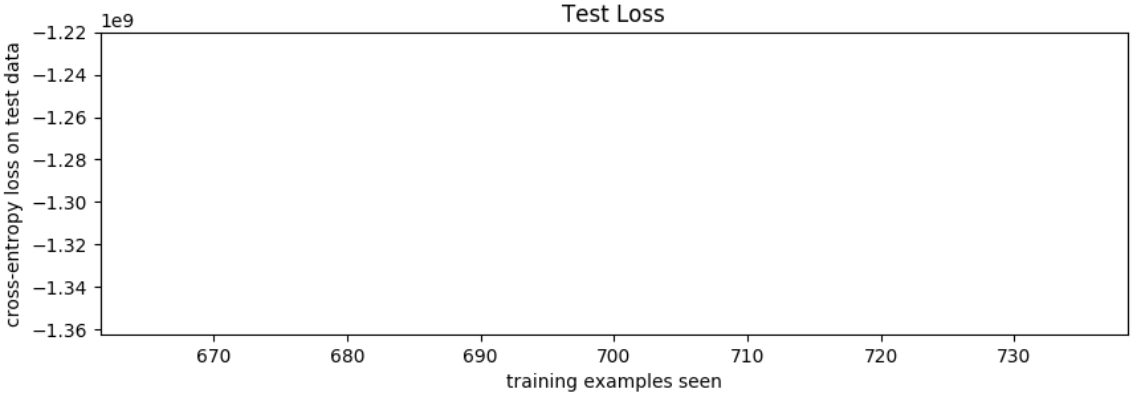
In [23]:
```
# Create some metrics.
from src import losses

train_data = train.get_batch(1000)
test_data = test.get_batch(1000)
def loss_fn(actual, pred):
    with tf.Session() as sess:
        actual = tf.constant(actual)
        pred = tf.constant(pred)
        loss = tf.losses.softmax_cross_entropy(actual, pred, reduction=tf.losses.R
eduction.SUM)
        loss = sess.run(loss)
    return loss
metrics = {
    "conf_mtx": metric.ConfusionMatrixMetric(test_data, 7),
    "train_loss": metric.LossMetric(train_data, loss_fn),
    "test_loss": metric.LossMetric(test_data, loss_fn),
    "off_by_counts": metric.OffByCountMetric(test_data, 7),
    "pred_thpt": metric.PredictionThroughputMetric(test_data)
}
```

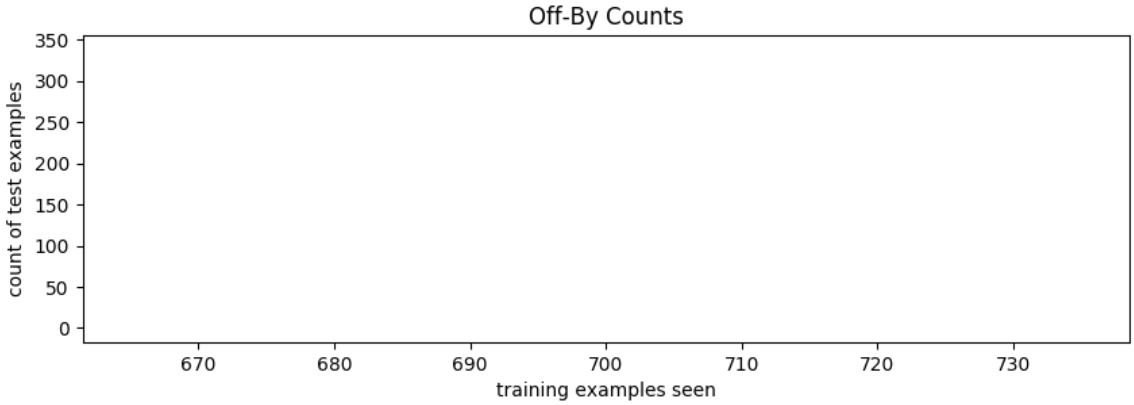
In [24]:
```
# Make a function for plotting the metrics.
def plot_metrics():
    mtx = metrics["conf_mtx"].get_results()[1][-1]
    visualization.plot_confusion_matrix(mtx, "Confusion Matrix", 3, 10)
    xs, ys = metrics["train_loss"].get_results()
    visualization.plot_line(xs, ys, "Training Loss", "training examples seen", "cr
oss-entropy loss on training data",
                            3, 10)
    xs, ys = metrics["test_loss"].get_results()
    visualization.plot_line(xs, ys, "Test Loss", "training examples seen", "cross-
entropy loss on test data", 3, 10)
    xs, sets_of_ys = metrics["off_by_counts"].get_results()
    visualization.plot_lines(xs, sets_of_ys, "Off-By Counts", "training examples s
een", "count of test examples",
                            ["off by {0}".format(x) for x in range(-7, 7 + 1)], 3,
10)
    xs, ys = metrics["pred_thpt"].get_results()
    visualization.plot_line(xs, ys, "Training Throughput", "training examples seen
", "speed of training in examples/s",
                            3, 10)
```

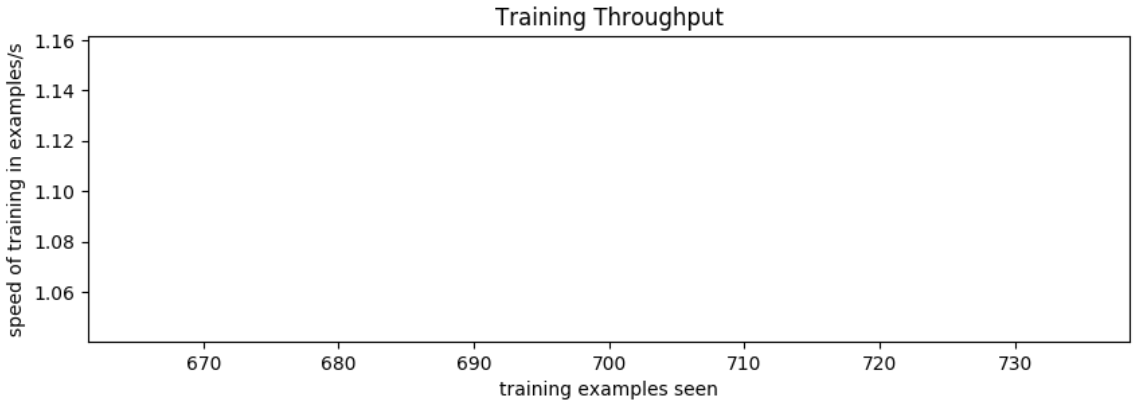
In [ ]:
```
# Alternately train and evaluate the net for 30 minutes.
for _ in range(30//3):
    net.train(train, 3*60)
    net.evaluate(metrics)
    plot_metrics()
```

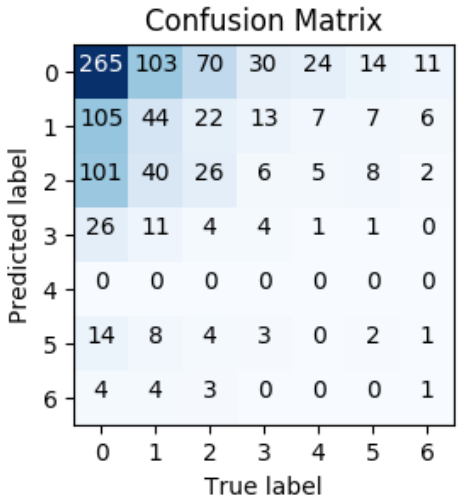

Confusion Matrix
Predicted label
True label
0 1 2 3 4 5 6
285 109 72 28 23 16 14
42 21 12 7 1 4 3
100 47 25 14 9 9 3
85 32 17 7 4 3 1
0 0 0 0 0 0 0
3 1 3 0 0 0 0
0 0 0 0 0 0 0

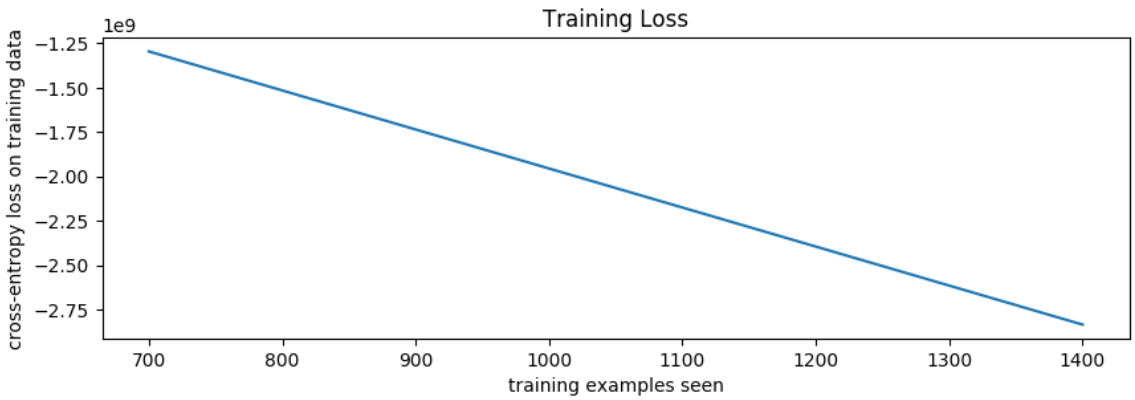

Training Loss
1e9
cross-entropy loss on training data
−1.24
−1.26
−1.28
−1.30
−1.32
−1.34
−1.36
670 680 690 700 710 720 730
training examples seen

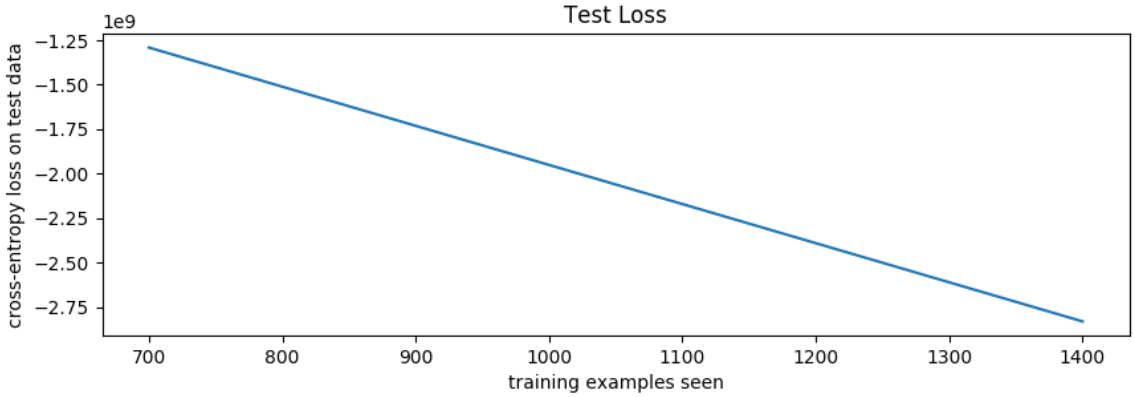

Test Loss
1e9
cross-entropy loss on test data
−1.22
−1.24
−1.26
−1.28
−1.30
−1.32
−1.34
−1.36
670 680 690 700 710 720 730
training examples seen

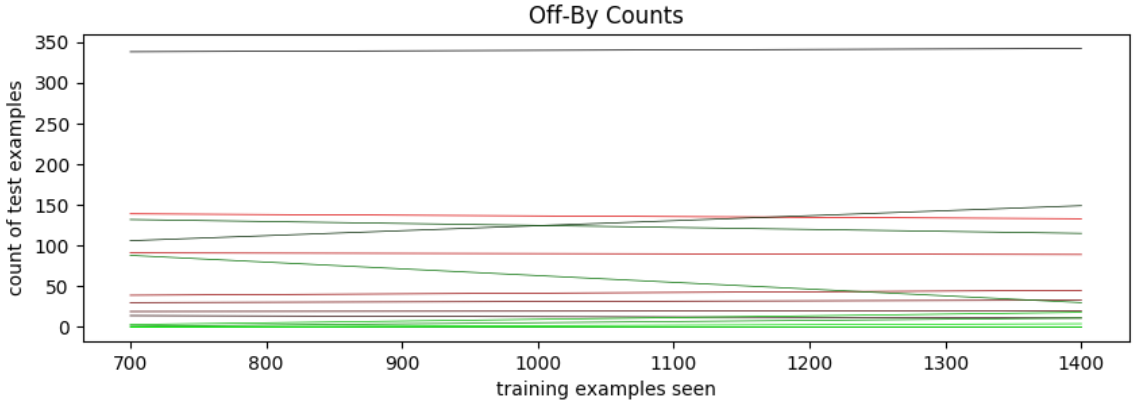

Off-By Counts
count of test examples
350
300
250
200
150
100
50
0
670
680
690
700
710
720
730
training examples seen

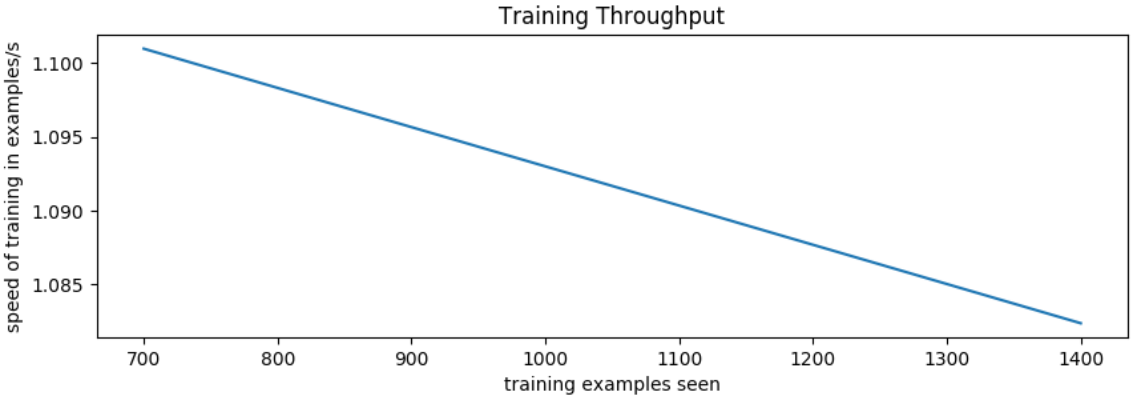

Training Throughput
speed of training in examples/s
1.16
1.14
1.12
1.10
1.08
1.06
670
680
690
700
710
720
730
training examples seen

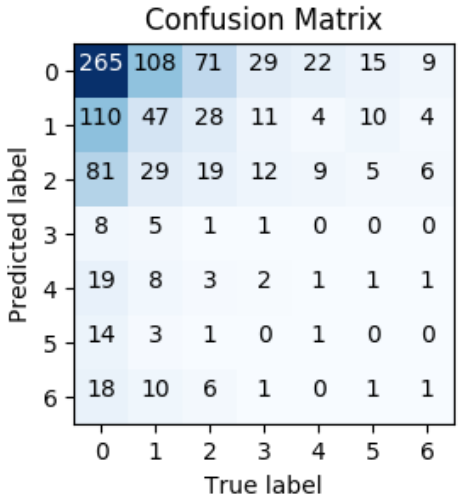

Confusion Matrix
Predicted label
True label
0 1 2 3 4 5 6
265 103 70 30 24 14 11
105 44 22 13 7 7 6
101 40 26 6 5 8 2
26 11 4 4 1 1 0
0 0 0 0 0 0 0
14 8 4 3 0 2 1
4 4 3 0 0 0 1

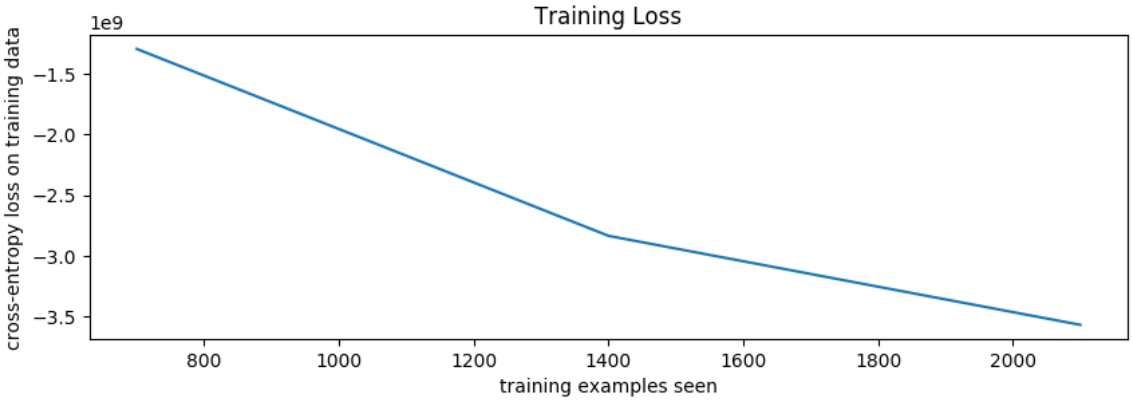

Training Loss
1e9
−1.25
−1.50
−1.75
−2.00
−2.25
−2.50
−2.75
cross-entropy loss on training data
700
800
900
1000
1100
1200
1300
1400
training examples seen

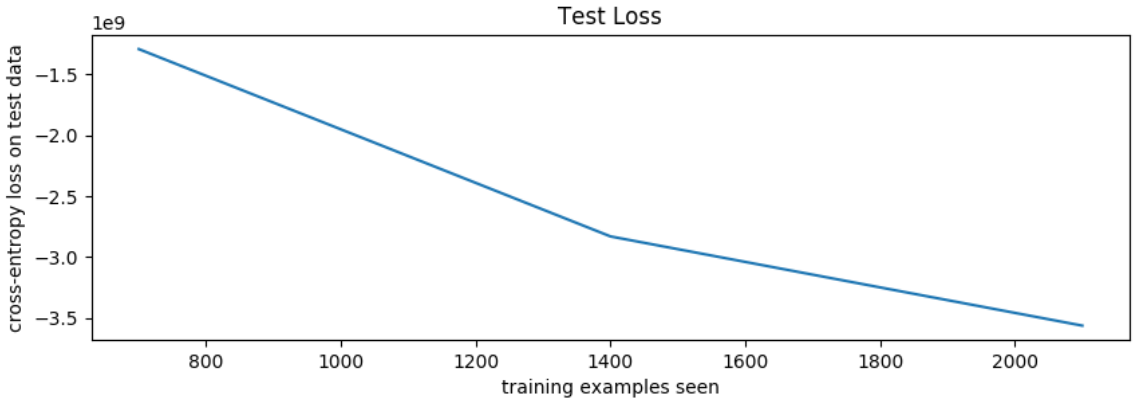

Test Loss
1e9
−1.25
−1.50
−1.75
−2.00
−2.25
−2.50
−2.75
cross-entropy loss on test data
700
800
900
1000
1100
1200
1300
1400
training examples seen

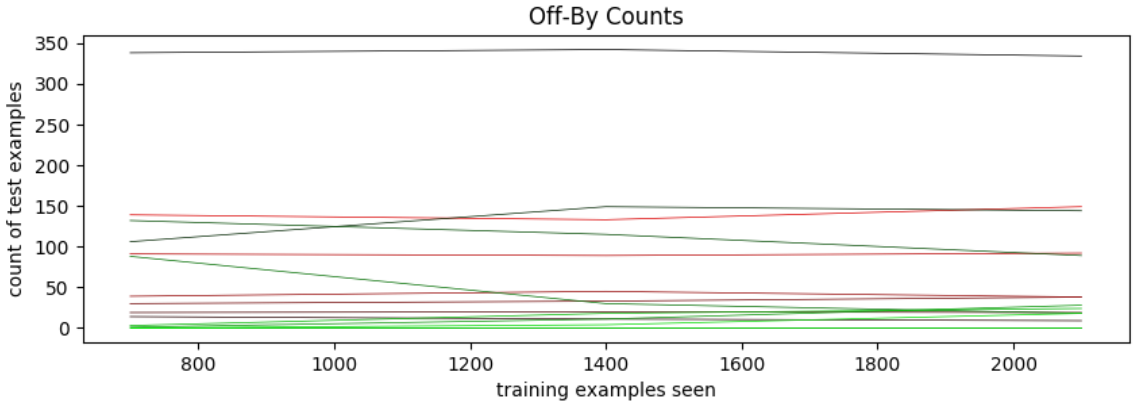

Off-By Counts
350
300
250
200
150
100
50
0
count of test examples
700
800
900
1000
1100
1200
1300
1400
training examples seen

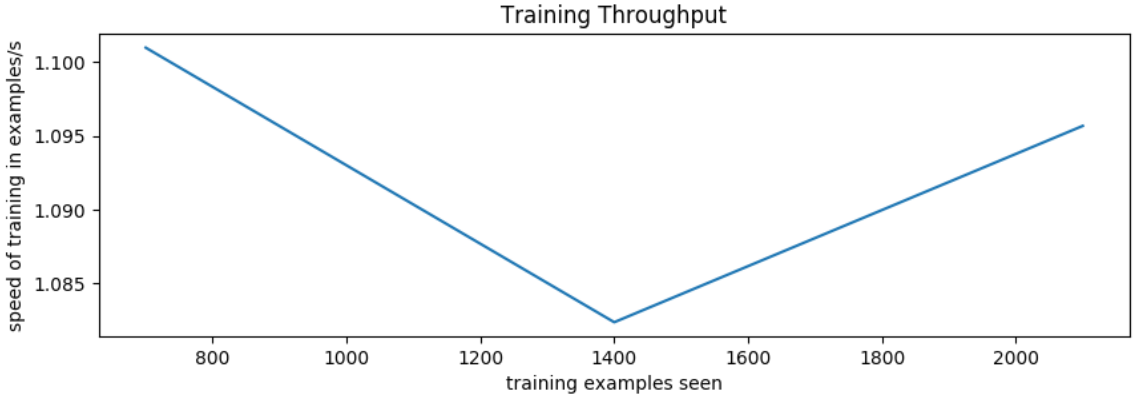

Training Throughput
speed of training in examples/s
1.100
1.095
1.090
1.085
700
800
900
1000
1100
1200
1300
1400
training examples seen

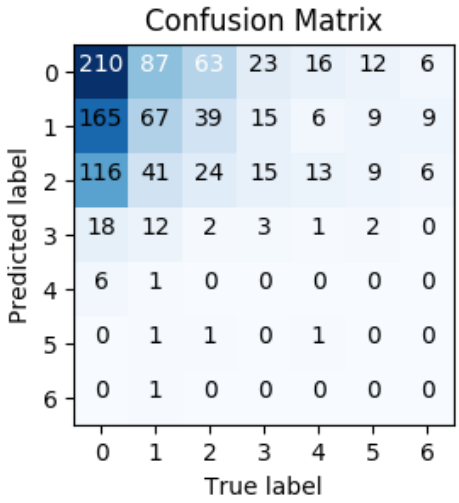

Confusion Matrix
Predicted label
0 265 108 71 29 22 15 9
1 110 47 28 11 4 10 4
2 81 29 19 12 9 5 6
3 8 5 1 1 0 0 0
4 19 8 3 2 1 1 1
5 14 3 1 0 1 0 0
6 18 10 6 1 0 1 1
0 1 2 3 4 5 6
True label

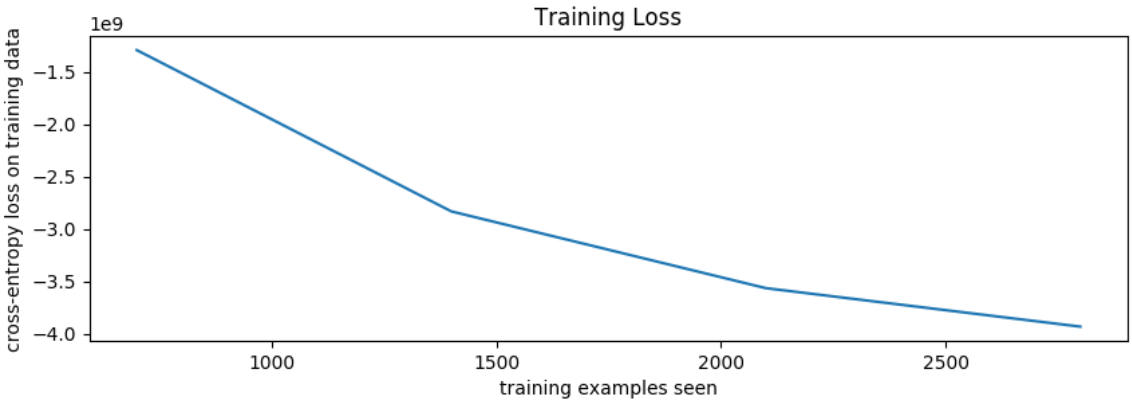

Training Loss
1e9
cross-entropy loss on training data
−1.5
−2.0
−2.5
−3.0
−3.5
800
1000
1200
1400
1600
1800
2000
training examples seen

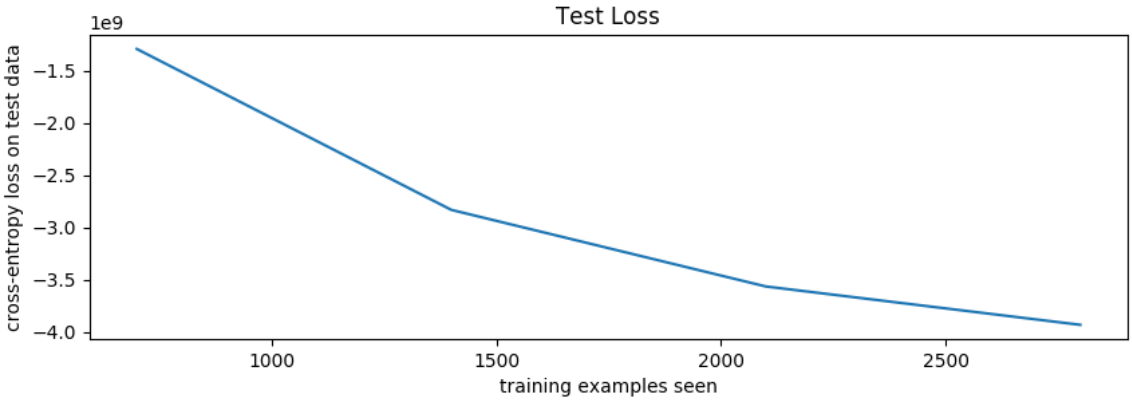

Test Loss
1e9
cross-entropy loss on test data
−1.5
−2.0
−2.5
−3.0
−3.5
800
1000
1200
1400
1600
1800
2000
training examples seen

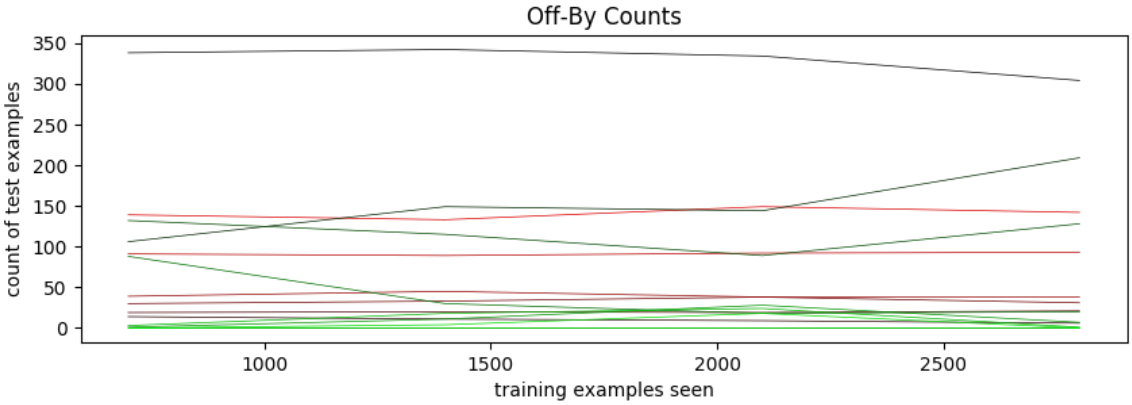

Off-By Counts
count of test examples
350
300
250
200
150
100
50
0
800
1000
1200
1400
1600
1800
2000
training examples seen


Training Throughput
speed of training in examples/s
1.100
1.095
1.090
1.085
800
1000
1200
1400
1600
1800
2000
training examples seen

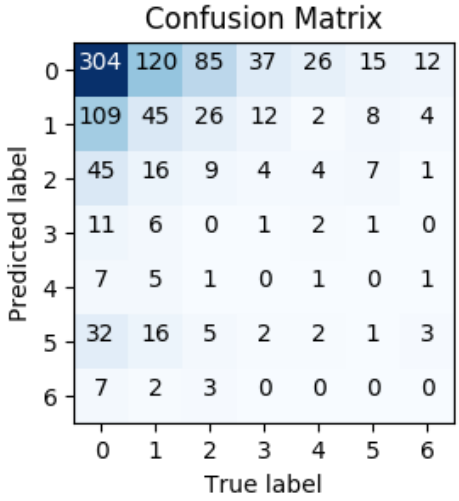

Confusion Matrix
Predicted label
True label
0 1 2 3 4 5 6
210 87 63 23 16 12 6
165 67 39 15 6 9 9
116 41 24 15 13 9 6
18 12 2 3 1 2 0
6 1 0 0 0 0 0
0 1 1 0 1 0 0
0 1 0 0 0 0 0

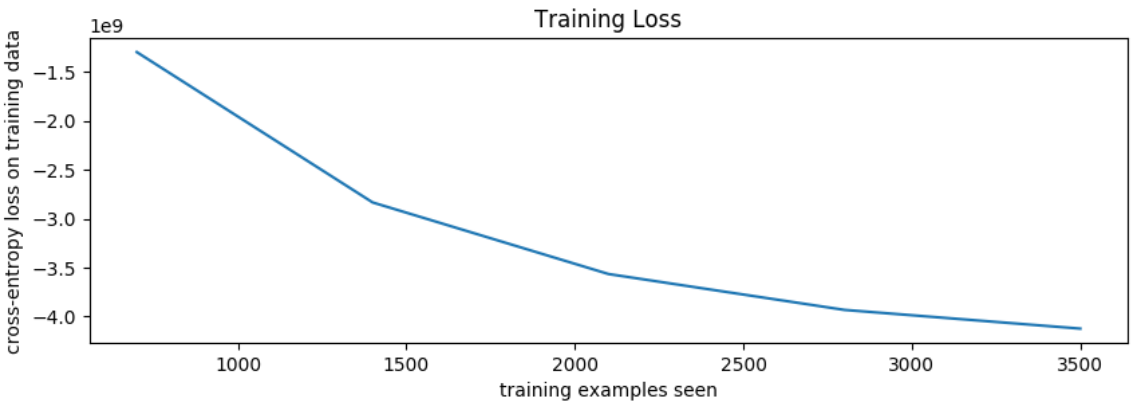

Training Loss
1e9
cross-entropy loss on training data
−1.5
−2.0
−2.5
−3.0
−3.5
−4.0
1000
1500
2000
2500
training examples seen

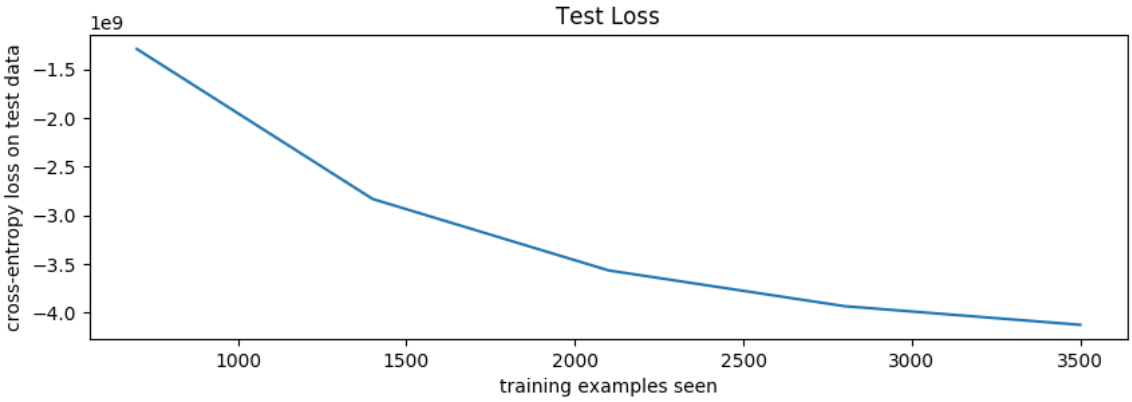

Test Loss
1e9
cross-entropy loss on test data
−1.5
−2.0
−2.5
−3.0
−3.5
−4.0
1000
1500
2000
2500
training examples seen

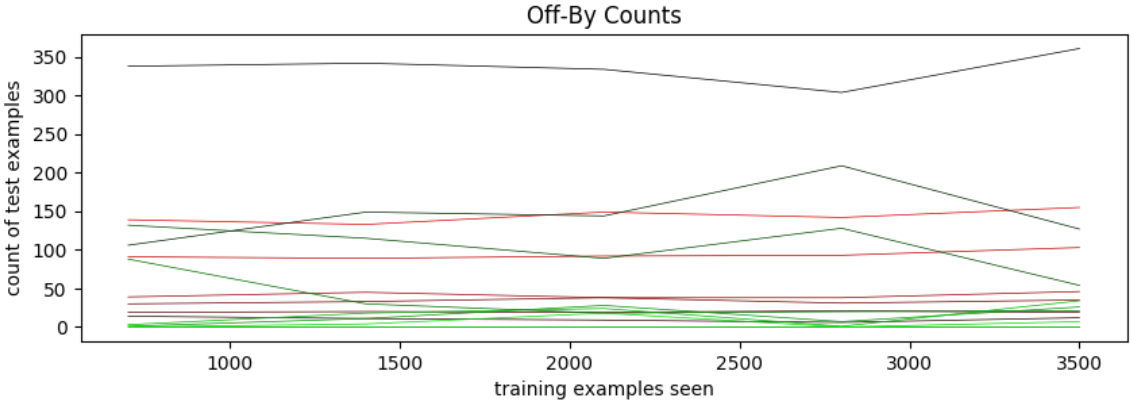

Off-By Counts
count of test examples
training examples seen
0
50
100
150
200
250
300
350
1000
1500
2000
2500


Training Throughput
speed of training in examples/s
training examples seen
1.080
1.085
1.090
1.095
1.100
1000
1500
2000
2500

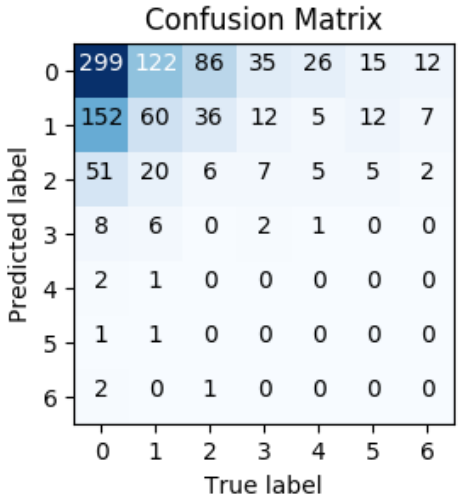

Confusion Matrix
Predicted label
True label
304 120 85 37 26 15 12
109 45 26 12 2 8 4
45 16 9 4 4 7 1
11 6 0 1 2 1 0
7 5 1 0 1 0 1
32 16 5 2 2 1 3
7 2 3 0 0 0 0
0 1 2 3 4 5 6

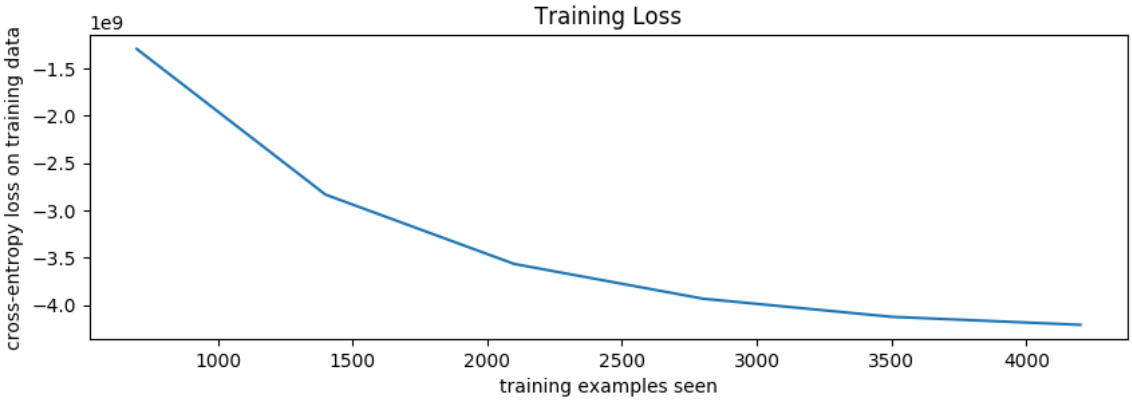

Training Loss
1e9
cross-entropy loss on training data
−1.5
−2.0
−2.5
−3.0
−3.5
−4.0
1000
1500
2000
2500
3000
3500
training examples seen


Test Loss
1e9
cross-entropy loss on test data
−1.5
−2.0
−2.5
−3.0
−3.5
−4.0
1000
1500
2000
2500
3000
3500
training examples seen

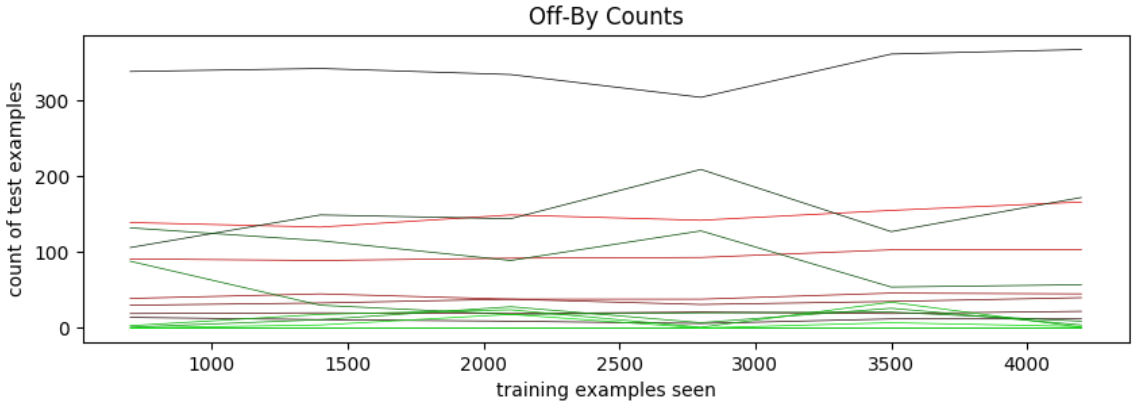

Off-By Counts
count of test examples
350
300
250
200
150
100
50
0
1000
1500
2000
2500
3000
3500
training examples seen


Training Throughput
speed of training in examples/s
1.105
1.100
1.095
1.090
1.085
1.080
1000
1500
2000
2500
3000
3500
training examples seen

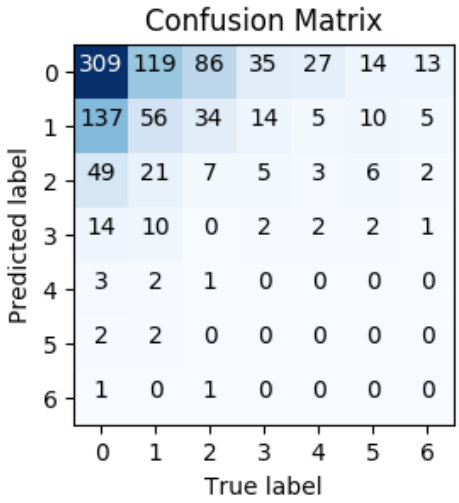

Confusion Matrix
Predicted label
True label
299 122 86 35 26 15 12
152 60 36 12 5 12 7
51 20 6 7 5 5 2
8 6 0 2 1 0 0
2 1 0 0 0 0 0
1 1 0 0 0 0 0
2 0 1 0 0 0 0
0 1 2 3 4 5 6


Training Loss
1e9
cross-entropy loss on training data
−1.5
−2.0
−2.5
−3.0
−3.5
−4.0
1000
1500
2000
2500
3000
3500
4000
training examples seen


Test Loss
1e9
cross-entropy loss on test data
−1.5
−2.0
−2.5
−3.0
−3.5
−4.0
1000
1500
2000
2500
3000
3500
4000
training examples seen

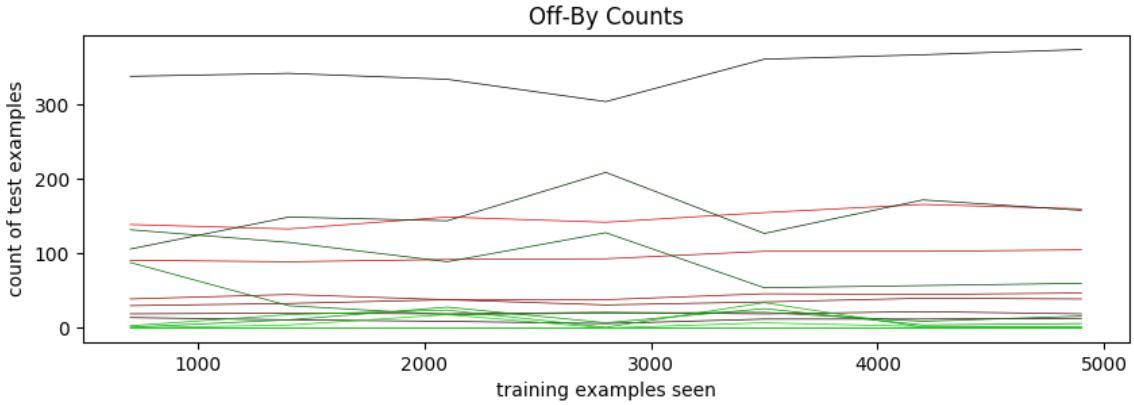

Off-By Counts
count of test examples
300
200
100
0
1000
1500
2000
2500
3000
3500
4000
training examples seen

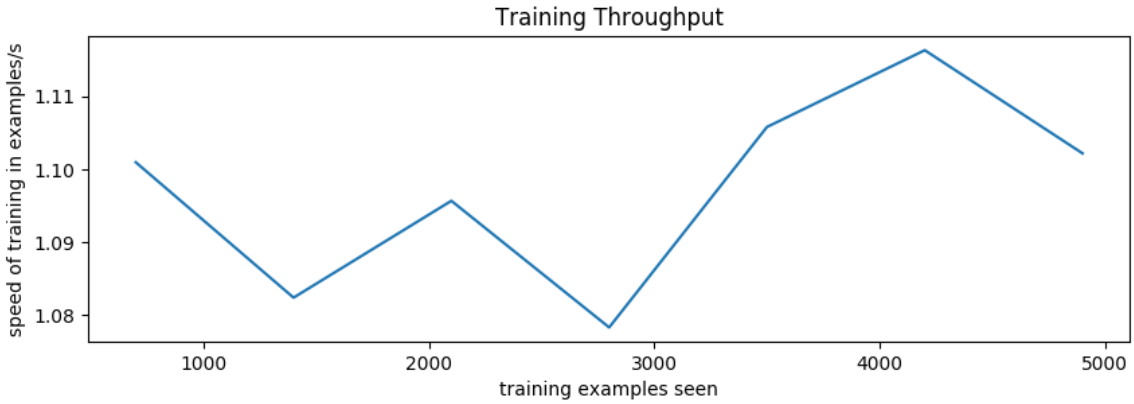

Training Throughput
speed of training in examples/s
1.11
1.10
1.09
1.08
1000
1500
2000
2500
3000
3500
4000
training examples seen

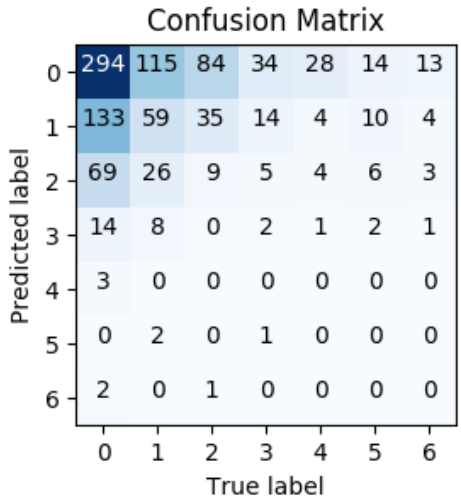

Confusion Matrix
Predicted label
True label
309 119 86 35 27 14 13
137 56 34 14 5 10 5
49 21 7 5 3 6 2
14 10 0 2 2 2 1
3 2 1 0 0 0 0
2 2 0 0 0 0 0
1 0 1 0 0 0 0
0 1 2 3 4 5 6

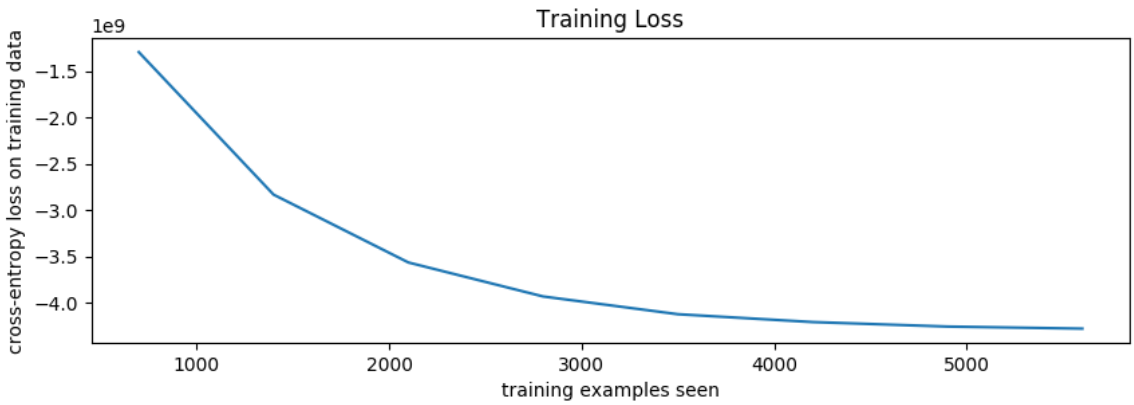

Training Loss
1e9
cross-entropy loss on training data
training examples seen
−1.5
−2.0
−2.5
−3.0
−3.5
−4.0
1000
2000
3000
4000
5000


Test Loss
1e9
cross-entropy loss on test data
training examples seen
−1.5
−2.0
−2.5
−3.0
−3.5
−4.0
1000
2000
3000
4000
5000

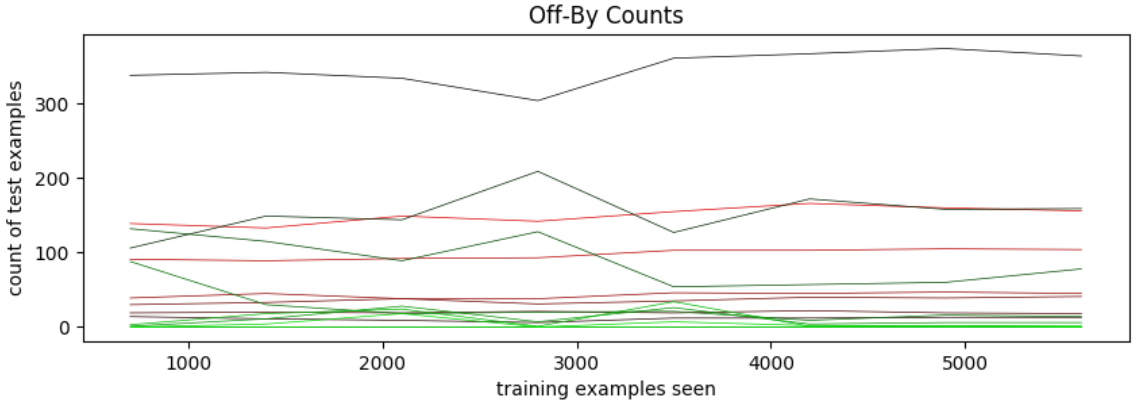

Off-By Counts
count of test examples
0
100
200
300
1000
2000
3000
4000
5000
training examples seen


Training Throughput
speed of training in examples/s
1.08
1.09
1.10
1.11
1000
2000
3000
4000
5000
training examples seen


Confusion Matrix
Predicted label
True label
294 115 84 34 28 14 13
133 59 35 14 4 10 4
69 26 9 5 4 6 3
14 8 0 2 1 2 1
3 0 0 0 0 0 0
0 2 0 1 0 0 0
2 0 1 0 0 0 0
0 1 2 3 4 5 6


Training Loss
1e9
−1.5
−2.0
−2.5
−3.0
−3.5
−4.0
cross-entropy loss on training data
1000
2000
3000
4000
5000
training examples seen


Test Loss
1e9
−1.5
−2.0
−2.5
−3.0
−3.5
−4.0
cross-entropy loss on test data
1000
2000
3000
4000
5000
training examples seen


Off-By Counts
300
200
100
0
count of test examples
1000
2000
3000
4000
5000
training examples seen

In [ ]:
```
# Alternately train and evaluate the net for 30 minutes.
for _ in range(30//3):
    net.train(train, 3*60)
    net.evaluate(metrics)
    plot_metrics()
```

In [ ]:
```
# Close the dataset.
microbia_segments.close()
```