In [3]:
```
# Configure matplotlib.
%matplotlib inline
```

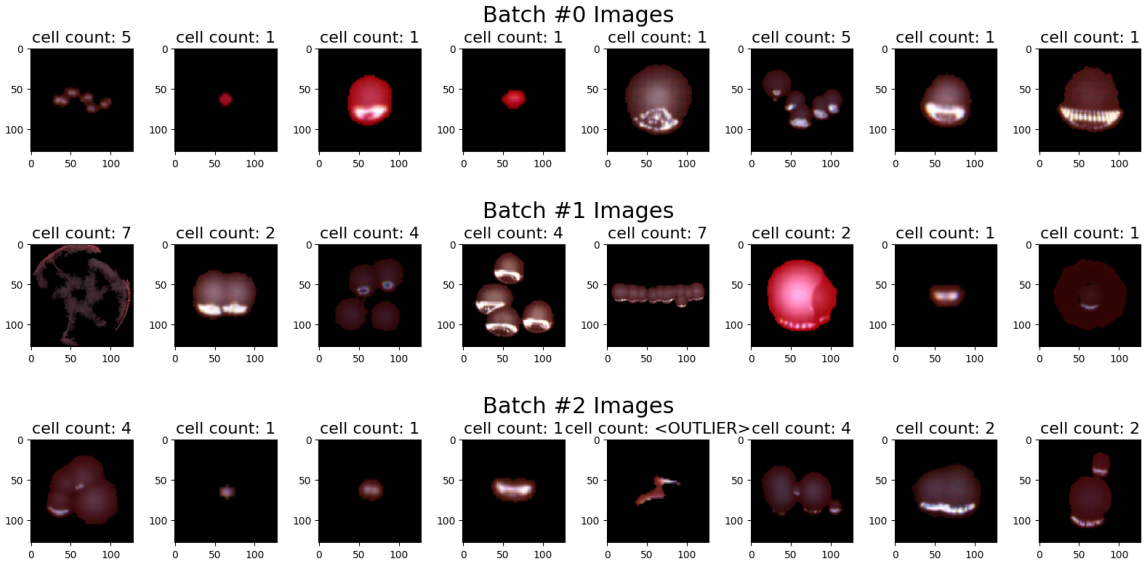
In [5]:
```
# Import our package.
import sys, importlib
sys.path.append("/home/ubuntu/cell_counting")

from src import dataset, visualization, preprocess, metric
from src.model import model
from src.model import neural_net
from src.model.segment_counting.convnet1 import convnet1
```

In [13]:
```
# (if changes are made) Re-import our package.
for module in (dataset, visualization, preprocess, metric, model, neural_net, conv
net1):
    importlib.reload(module)
```

In [6]:
```
# 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(1000)
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 [7]:
```
# 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 [8]:
```
# 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 [9]:
```
# Split the dataset.
train, test = microbia_segments.split(0.1)
```

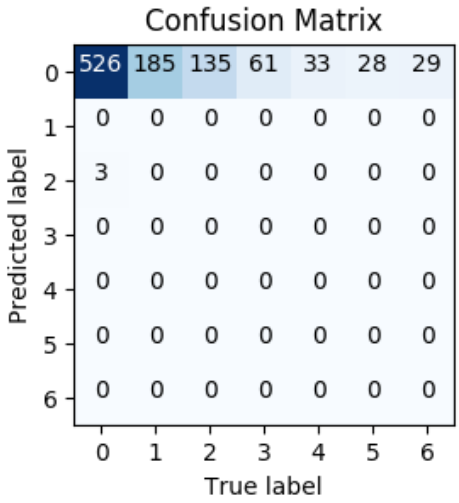
In [14]:
```
# Create the net.
import tensorflow as tf
net = convnet1.ConvNet1("saves/17-12-02-PM-11-27", 120)
```

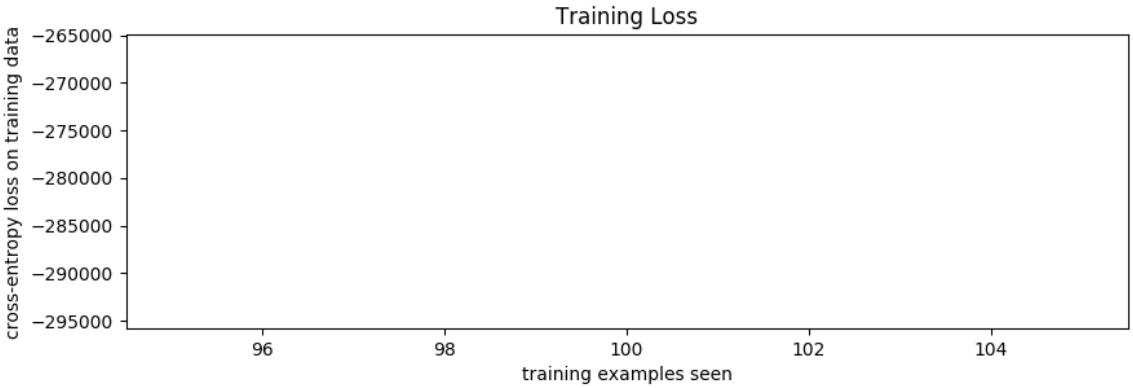
```
INFO:tensorflow:Using config: {'_model_dir': 'saves/17-12-02-PM-11-27', '_tf_ran
dom_seed': None, '_save_summary_steps': 100, '_save_checkpoints_steps': None, '_
save_checkpoints_secs': 120, '_session_config': None, '_keep_checkpoint_max': 2,
'_keep_checkpoint_every_n_hours': 10000, '_log_step_count_steps': 100, '_service
': None, '_cluster_spec': <tensorflow.python.training.server_lib.ClusterSpec obj
ect at 0x7f59fae1fe10>, '_task_type': 'worker', '_task_id': 0, '_master': '', '_
is_chief': True, '_num_ps_replicas': 0, '_num_worker_replicas': 1}
```

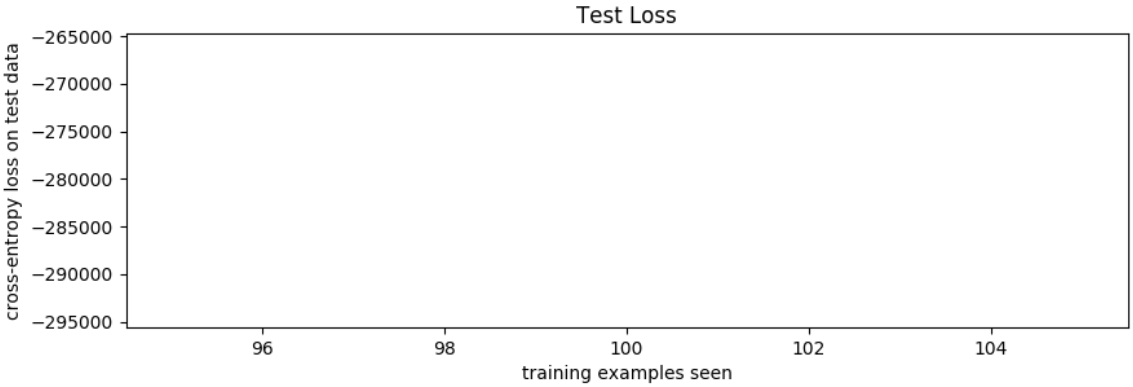
In [15]:
```
# Create some metrics.
train_data = train.get_batch(1000)
test_data = test.get_batch(1000)
def loss_fn(actual, pred):
    with tf.Session() as sess:
        actual = tf.constant(actual)
        pred = tf.constant(pred)
        loss = tf.losses.softmax_cross_entropy(actual, pred, reduction=tf.losses.R
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)
}
```

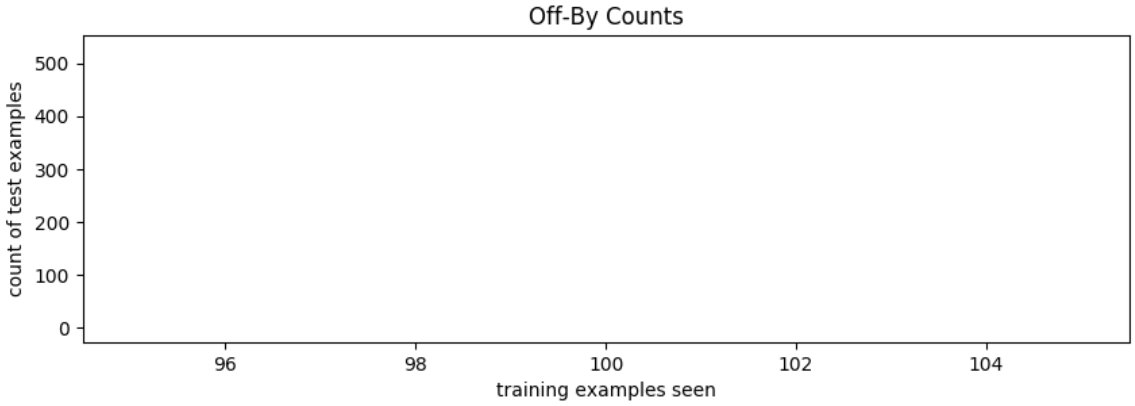
In [16]:
```
# Make a function for plotting the metrics.
def plot_metrics():
    mtx = metrics["conf_mtx"].get_results()[1][-1]
    visualization.plot_confusion_matrix(mtx, "Confusion Matrix", 3, 10)
    xs, ys = metrics["train_loss"].get_results()
    visualization.plot_line(xs, ys, "Training Loss", "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)
```

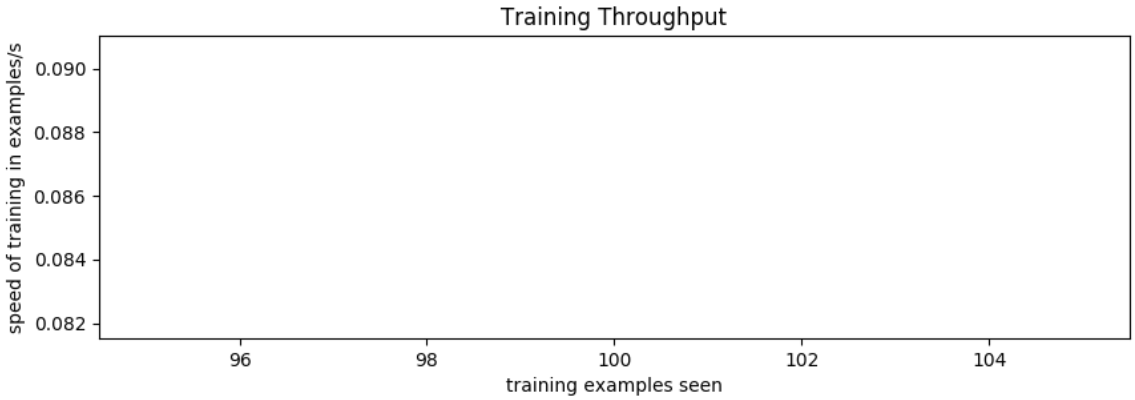
In [17]:
```
# Alternately train and evaluate the net for 30 minutes.
for _ in range(30//3):
    net.train(train, 3*60)
    net.evaluate(metrics)
    plot_metrics()
```

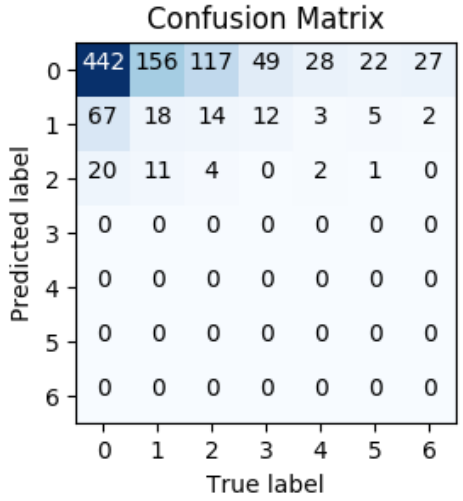

Confusion Matrix
526 185 135 61 33 28 29
0 0 0 0 0 0 0
3 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
Predicted label
True label
0 1 2 3 4 5 6

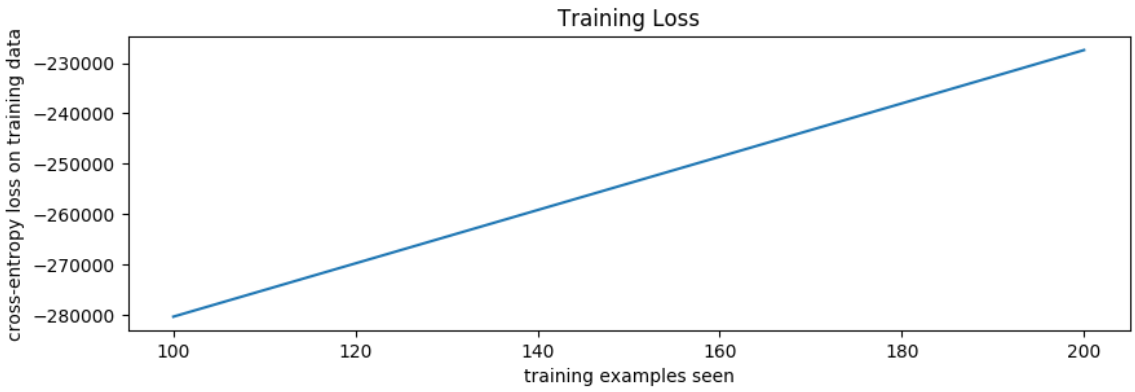

Training Loss
cross-entropy loss on training data
−265000
−270000
−275000
−280000
−285000
−290000
−295000
96 98 100 102 104
training examples seen


Test Loss
cross-entropy loss on test data
−265000
−270000
−275000
−280000
−285000
−290000
−295000
96 98 100 102 104
training examples seen

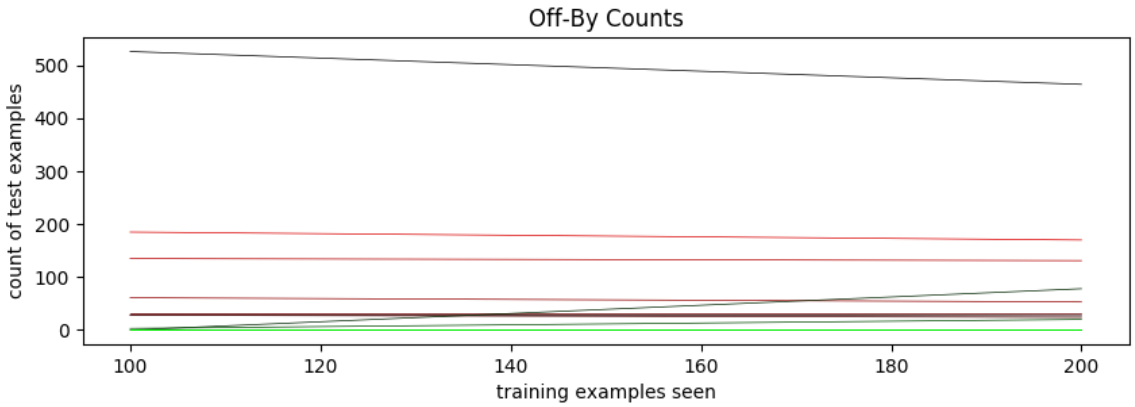

Off-By Counts
count of test examples
500
400
300
200
100
0
96
98
100
102
104
training examples seen

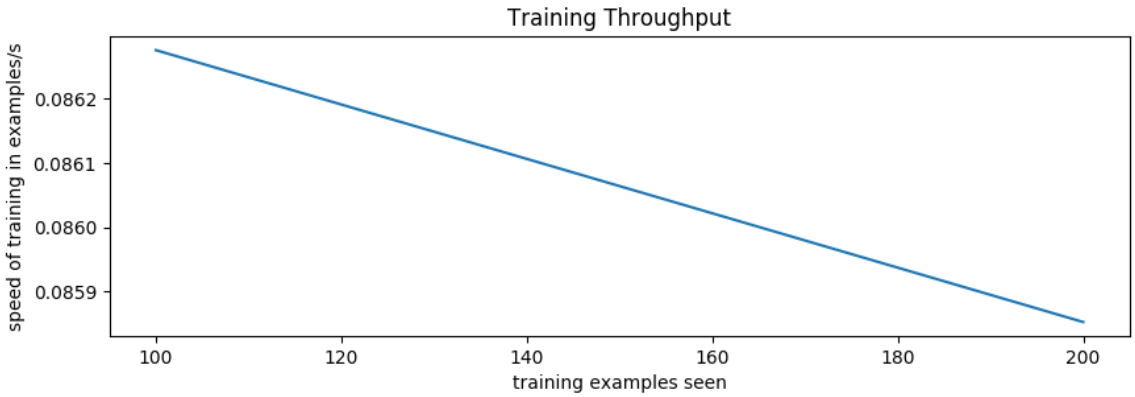

Training Throughput
speed of training in examples/s
0.090
0.088
0.086
0.084
0.082
96
98
100
102
104
training examples seen


Confusion Matrix
Predicted label
True label
442 156 117 49 28 22 27
67 18 14 12 3 5 2
20 11 4 0 2 1 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 1 2 3 4 5 6


Training Loss
cross-entropy loss on training data
−230000
−240000
−250000
−260000
−270000
−280000
100
120
140
160
180
200
training examples seen


Test Loss
cross-entropy loss on test data
−230000
−240000
−250000
−260000
−270000
−280000
100
120
140
160
180
200
training examples seen


Off-By Counts
count of test examples
500
400
300
200
100
0
100
120
140
160
180
200
training examples seen


Training Throughput
speed of training in examples/s
0.0862
0.0861
0.0860
0.0859
100
120
140
160
180
200
training examples seen

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-17-ea4c53e03ea9> in <module>()
      1 # Alternately train and evaluate the net for 30 minutes.
      2 for _ in range(30//3):
----> 3     net.train(train, 3*60)
      4     net.evaluate(metrics)
      5     plot_metrics()

~/cell_counting/src/model/model.py in train(self, dataset, seconds)
     49                                     data_fn = dataset.get_data_fn(self._get_
batch_size(),
     50                                             self._TRAIN_STEPS)
---> 51                                     self._estimator.train(data_fn, steps=sel
f._TRAIN_STEPS)
     52                                     batches += self._TRAIN_STEPS
     53                                     self._global_step += self._TRAIN_STEPS

~/anaconda3/envs/tensorflow_p36/lib/python3.6/site-packages/tensorflow/python/es
timator/estimator.py in train(self, input_fn, hooks, steps, max_steps, saving_li
steners)
    300
    301     saving_listeners = _check_listeners_type(saving_listeners)
--> 302     loss = self._train_model(input_fn, hooks, saving_listeners)
    303     logging.info('Loss for final step: %s.', loss)
    304     return self

~/anaconda3/envs/tensorflow_p36/lib/python3.6/site-packages/tensorflow/python/es
timator/estimator.py in _train_model(self, input_fn, hooks, saving_listeners)
    781         loss = None
    782         while not mon_sess.should_stop():
--> 783           _, loss = mon_sess.run([estimator_spec.train_op, estimator_spe
c.loss])
    784       return loss
    785

~/anaconda3/envs/tensorflow_p36/lib/python3.6/site-packages/tensorflow/python/tr
aining/monitored_session.py in run(self, fetches, feed_dict, options, run_metada
ta)
    519                           feed_dict=feed_dict,
    520                           options=options,
--> 521                           run_metadata=run_metadata)
    522
    523   def should_stop(self):

~/anaconda3/envs/tensorflow_p36/lib/python3.6/site-packages/tensorflow/python/tr
aining/monitored_session.py in run(self, fetches, feed_dict, options, run_metada
ta)
    890                               feed_dict=feed_dict,
    891                               options=options,
--> 892                               run_metadata=run_metadata)
    893       except _PREEMPTION_ERRORS as e:
    894         logging.info('An error was raised. This may be due to a preempti
on in '

~/anaconda3/envs/tensorflow_p36/lib/python3.6/site-packages/tensorflow/python/tr
aining/monitored_session.py in run(self, *args, **kwargs)
    950   def run(self, *args, **kwargs):
    951     try:
--> 952       return self._sess.run(*args, **kwargs)
```

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()
```