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

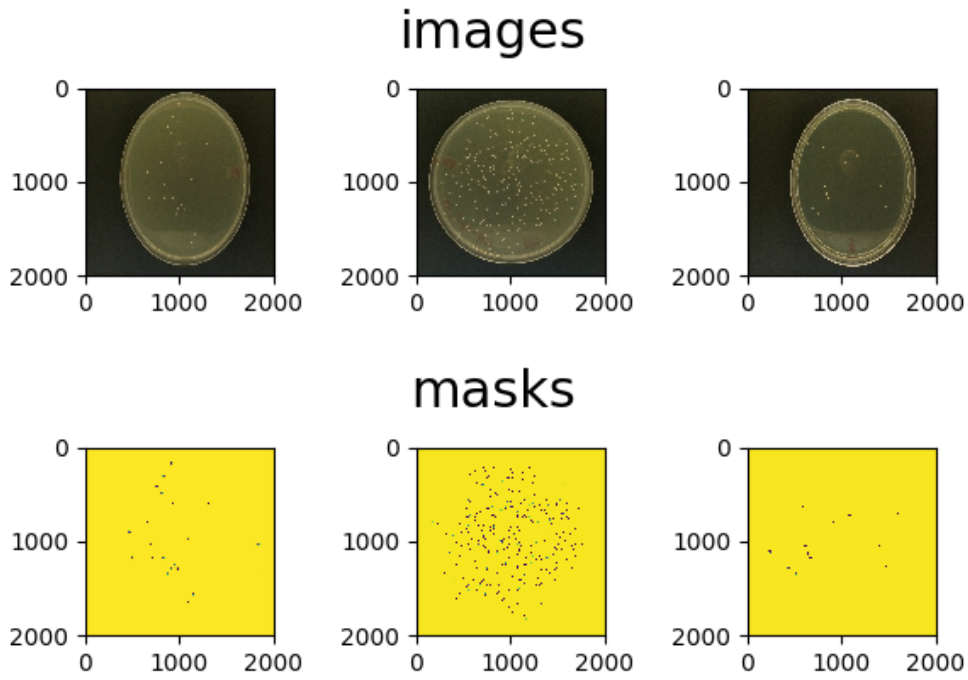
In [2]:
```
# Import our package.
import sys, importlib
sys.path.append("/home/ubuntu/cell_counting")

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

```
/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 [66]:
```
# (if changes are made) Re-import our package.
for module in (dataset, visualization, preprocess, metric, model, neural_net, conv
net1, losses):
    importlib.reload(module)
```

In [3]:
```
# Load the dataset, processing it as a collection of image-mask pairs.
images_masks = dataset.Dataset(1)
images_masks.load_image_mask_pairs("/home/ubuntu/cell_counting/data/easy/raw/image
s",
                                   "/home/ubuntu/cell_counting/data/easy/raw/masks
", (2000, 2000))
```

In [4]:
```
# Plot a batch.
inputs, outputs = images_masks.get_batch(3)
visualization.show_image_grid(inputs, 1, 3, 2, 6, "images")
visualization.show_image_grid(outputs, 1, 3, 2, 6, "masks")
```



In [ ]:
```
# Normalize the images.
#def normalize(batch):
#    inputs, outputs = batch
#    inputs = preprocess.smdm_normalize(inputs, 25, "REFLECT")
#    return (inputs, outputs)
#images_masks.map_batch(normalize)
```

In [ ]:
```
# Plot a batch.
#inputs, outputs = images_masks.get_batch(3)
#visualization.show_image_grid(inputs, 1, 3, 2, 6, "images")
#visualization.show_image_grid(outputs, 1, 3, 2, 6, "masks")
```

In [5]:
```
# Extract patches from the images.
def extract_patches(example):
    input_, output = example
    input_ = preprocess.extract_patches(input_, 25, 100000, 42114)
    output = preprocess.extract_patches(output, 25, 100000, 42114)
    examples = [(input_[i, ...] / 255, 0 if output[i, 25//2 + 1, 25//2 + 1] > 50 e
lse 1) for i in range(input_.shape[0])]
    return examples
images_masks.map(extract_patches)
```

In [6]:
```
# Plot a batch.
inputs, outputs = images_masks.get_batch(8)
visualization.show_image_grid(inputs * 255, 1, 8, 2.5, 16, "images",
                              [("colony" if outputs[i] == 1 else "not colony") for
i in range(outputs.shape[0])])
```

In [7]:
```
# Split the dataset.
train, test = images_masks.split(0.4)
```

In [69]:
```
# Create the net.
import tensorflow as tf
net = convnet1.ConvNet1("saves/17-11-27-PM-03-54", 120)
```

In [70]:
```
# Create some metrics.
train_data = train.get_batch(1000)
test_data = test.get_batch(1000)
loss_fn = losses.mse_loss
metrics = {
    "train_loss": metric.LossMetric(train_data, loss_fn),
    "test_loss": metric.LossMetric(test_data, loss_fn),
    "pred_thpt": metric.PredictionThroughputMetric(test_data)
}
```

In [71]:
```
# Make a function for plotting the metrics.
def plot_metrics():
    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, 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 [72]:
```
# Alternately train and evaluate the net for 30 minutes.
for _ in range(30//3):
    net.train(train, 3*60)
    net.evaluate(metrics)
    plot_metrics()
```

```
ERROR:tensorflow:Model diverged with loss = NaN.
```

```
---------------------------------------------------------------------------
NanLossDuringTrainingError                Traceback (most recent call last)
<ipython-input-72-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)
    965         raise six.reraise(*original_exc_info)
    966       else:
--> 967         raise six.reraise(*original_exc_info)
```

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