In [1]:
```
# Autoreload changed python modules
%load_ext autoreload
```

In [2]:
```
import os, glob
import h5py # for loading .h5 files
import numpy as np
import matplotlib.pyplot as plt
import random
%matplotlib inline
```

In [3]:
```
plt.style.use('seaborn') # pretty matplotlib plots
plt.rcParams["axes.grid"] = False # don't show grid lines on plots by
plt.rcParams['figure.figsize'] = (12, 16) # increase size of subplots
```

In [4]:
```
import cs230_project_utilities as utils
# make sure you have pywt: pip3 install PyWavelets --user
```

Loading the data

In [5]:
```
# Location of directory H5Exports_AnimiX/ (downloaded from Olivier's l
DATASET_DIRECTORY = '/home/ubuntu/cs230/data/H5Exports_AnimiX'
```

In [6]:
```
# Find all the files in our dataset
h5_files = utils.automap.find_dataset_files(DATASET_DIRECTORY)
```

```
Found 134 .h5 files and 134 .txt files.
```

Visualizing the data

In [7]:

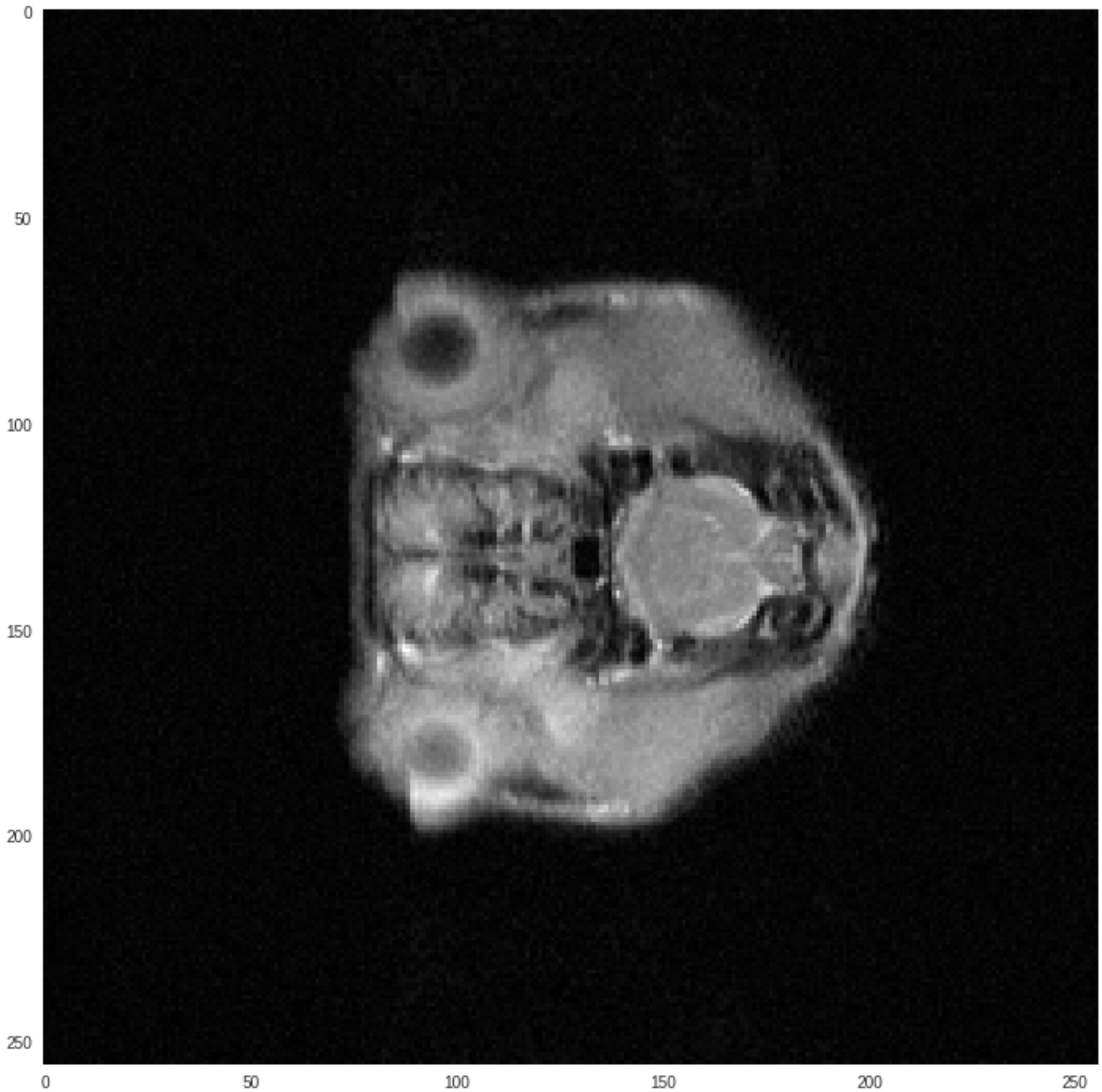
```python
##### Finally, we can see the raw data
sample_filename = list(h5_files.keys())[20]
print('Taking a look at file: {}'.format(sample_filename))
h5 = h5_files[sample_filename]['h5']
h5_data = utils.automap.read_h5_file(h5)
print(h5_data.keys())

images = h5_data['images']
magnitude = h5_data['magnitude']
phase = h5_data['phase']
classification = h5_data['classification']

# (Note: shape of magnitude and phase are different from image)
print(images.shape, magnitude.shape, phase.shape)
print(np.ndarray.flatten(classification))
```

```
Taking a look at file: 943_10158
dict_keys(['classification', 'images', 'magnitude', 'phase'])
(15, 256, 256) (15, 256, 256) (15, 256, 256)
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

In [8]:

```python
sample_index = np.argmax(classification)
```

In [9]:
```
# Uncomment to view example image in dataset
utils.plot.imshowgray(images[sample_index])
```

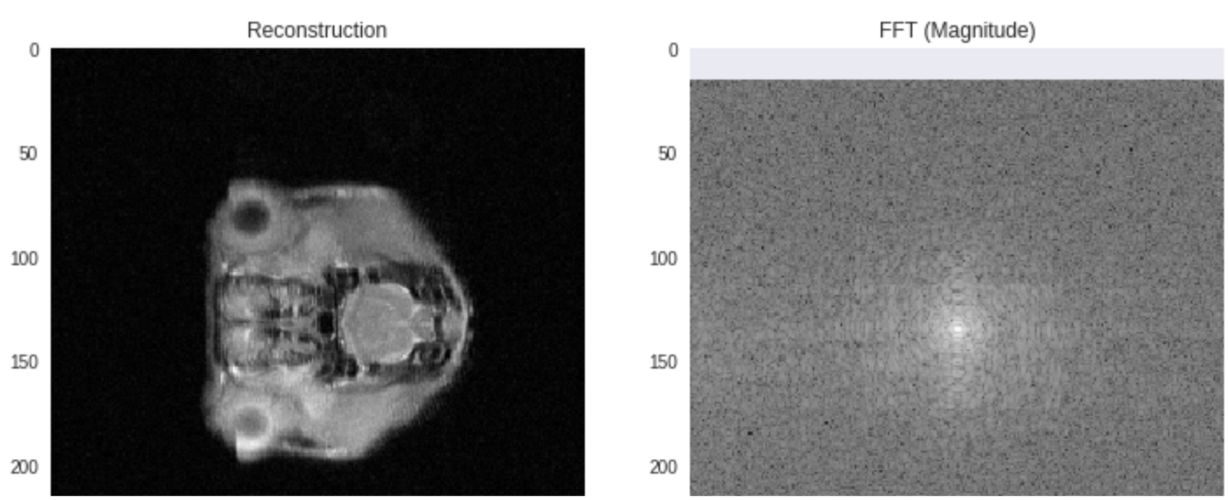


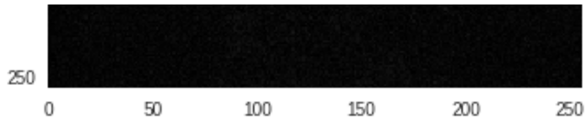
In [10]:
```
# Construct FFT (k-space data) from magnitude and phase
fft = magnitude[sample_index] * np.exp(1j * phase[sample_index])

# Take the inverse FFT
ifft = utils.signal_processing.ifft_2D_centered(fft)

# Note: shape of magnitude and phase are different from image.
# Because of this, the reconstruction shape is different from the
# image shape and so we can't compare the image and reconstruction di
# How will we solve this?

# This check to make sure we are correctly combining magnitude and pha
print('Error in FFT magnitude: {}'.format(utils.signal_processing.mear
print('Error in FFT phase: {}'.format(utils.signal_processing.mean_squ
```

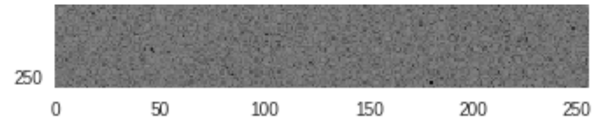
```
Error in FFT magnitude: 2.4687782802584184e-20
Error in FFT phase: 5.288901564141631e-16
```

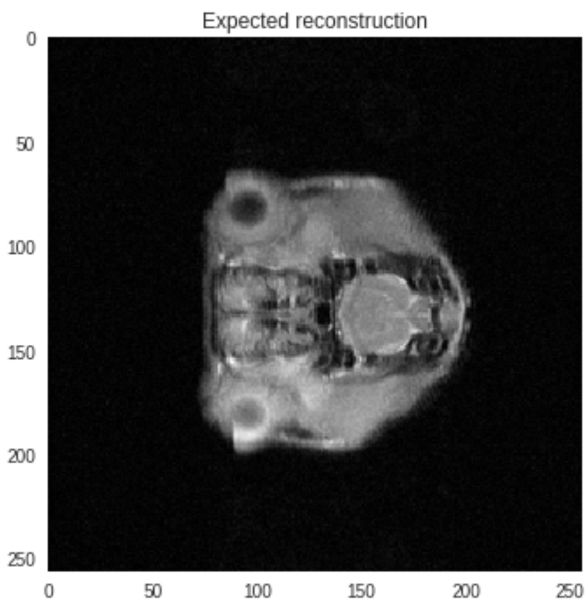
In [11]:
```
''' Uncomment to show plots.'''
plt.subplot(2, 2, 1)
plt.title('Reconstruction')
utils.plot.imshowgray(np.abs(ifft))
#
plt.subplot(2, 2, 2)
plt.title('FFT (Magnitude)')
utils.plot.imshowfft(np.abs(fft))

plt.subplot(2, 2, 3)
plt.title('Expected reconstruction')
image = images[sample_index]
utils.plot.imshowgray(image)

plt.subplot(2, 2, 4)
plt.title('FFT (Phase)')
utils.plot.imshowgray(np.angle(fft))
```

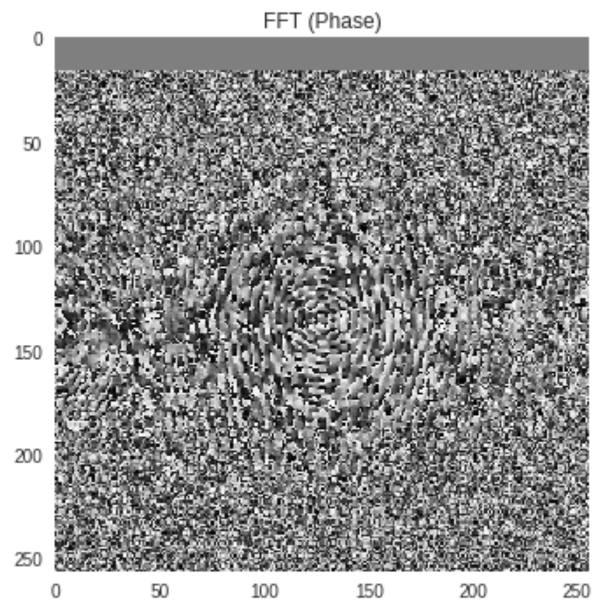
```
/home/ubuntu/cs230/code/MRI_reconstruction_example/cs230_project_util
ities/plot.py:13: RuntimeWarning: divide by zero encountered in log10
  imshowgray(20 * np.log10(np.abs(fft)))
```











Automap Model

In [12]:
```
import tensorflow as tf
```

In [13]:
```
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
```

In [14]:
```
import keras
from keras.layers import Input,Conv2D, Conv2DTranspose, Dense, Reshape
from keras.models import Model
from keras.optimizers import RMSprop
from keras import losses
```

```
Using TensorFlow backend.
```

In [15]:
```
tf.reset_default_graph()
```

In [16]:
```
def load_model():
    n_H, n_W = 256, 256
    X = Input((n_H, n_W, 2))
    conv_downsample1 = Conv2D(16, (4, 4), strides=(2, 2), activation='
    conv_downsample2 = Conv2D(4, (4, 4), strides=(1, 1), activation='t
    conv_downsample3 = Conv2D(2, (4, 4), strides=(2, 2), activation='t
    X1 = Flatten()(conv_downsample3)
    current_H, current_W = (256 // 4, 256 // 4) # after downsampling b
    fc1 = Dense(current_H * current_W * 2, activation = 'tanh')(X1)
    fc2 = Dense(current_H * current_W, activation = 'tanh')(fc1)
    fc3 = Dense(current_H * current_W, activation = 'tanh')(fc2)
    X2 = Reshape((current_H, current_W, 1))(fc3)
    conv1_1 = Conv2D(64, 5, activation='relu', padding='same')(X2)
    conv1_2 = Conv2D(64, 5, activation='relu', padding='same')(conv1_1
    conv1_3a = Conv2DTranspose(64, 9, activation='relu', padding='same
    conv1_3b = Conv2DTranspose(64, 9, strides=2, activation='relu', pa
    conv1_3c = Conv2DTranspose(64, 9, strides=2, activation='relu', pa
    out = Conv2D(1, 1, activation = 'linear',padding='same')(conv1_3c)
    model = Model(inputs=X, outputs=out)
    model.compile(optimizer=keras.optimizers.Adam(lr=1e-4, decay=1e-3)
    return model
```

In [17]:
```
# # Original full model. Trainable params: 26,424,796,993.
# # (The 64 x 64 version has 117,892,929 params. Baseline has 74,843,0
# def load_model():
#     n_H, n_W = 256, 240
#     X = Input((n_H, n_W, 2))
#     X1 = Flatten()(X)
#     fc1 = Dense(n_H * n_W * 2, activation = 'tanh')(X1)
#     fc2 = Dense(n_H * n_W, activation = 'tanh')(fc1)
#     fc3 = Dense(n_H * n_W, activation = 'tanh')(fc2)
#     X2 = Reshape((n_H, n_W, 1))(fc3)
#     conv1_1 = Conv2D(64, 5, activation='relu', padding='same')(X2)
#     conv1_2 = Conv2D(64, 5, activation='relu', padding='same')(conv1
#     conv1_3 = Conv2DTranspose(64, 9, activation='relu', padding='sam
#     out = Conv2D(1, 1, activation = 'linear',padding='same')(conv1_3
#     model = Model(inputs=X, outputs=out)
#     model.compile(optimizer=RMSprop(lr=1e-5), loss='mean_squared_err
#     return model
```

In [18]:
```python
model = load_model()
print(model.summary())
```

```
WARNING:tensorflow:From /home/ubuntu/anaconda3/envs/tensorflow_p36/li
b/python3.6/site-packages/tensorflow/python/framework/op_def_library.
py:263: colocate_with (from tensorflow.python.framework.ops) is depre
cated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         (None, 256, 256, 2)       0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 128, 128, 16)      528       
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 128, 128, 4)       1028      
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 64, 64, 2)         130       
_________________________________________________________________
flatten_1 (Flatten)          (None, 8192)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 8192)              67117056  
_________________________________________________________________
dense_2 (Dense)              (None, 4096)              33558528  
_________________________________________________________________
dense_3 (Dense)              (None, 4096)              16781312  
_________________________________________________________________
reshape_1 (Reshape)          (None, 64, 64, 1)         0         
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 64, 64, 64)        1664      
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 64, 64, 64)        102464    
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 64, 64, 64)        331840    
_________________________________________________________________
conv2d_transpose_2 (Conv2DTr (None, 128, 128, 64)      331840    
_________________________________________________________________
conv2d_transpose_3 (Conv2DTr (None, 256, 256, 64)      331840    
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 256, 256, 1)       65        
=================================================================
Total params: 118,558,295
Trainable params: 118,558,295
Non-trainable params: 0
_________________________________________________________________
None
```

In [19]:
```python
def batch_generator():
    # Generate training batches by reading sequences from disk

    randomize_file_order = True
```

```
    # Find all the files in our dataset
    h5_files = utils.automap.find_dataset_files(DATASET_DIRECTORY)

    batch_size = 16
    n_H, n_W = 256, 256

    X_batch = np.zeros((batch_size, n_H, n_W, 2))
    Y_batch = np.zeros((batch_size, n_H, n_W, 1))

    current_batch_size = 0

    h5_keys = list(h5_files.keys())

    current_h5_key_index = 0

    while True:

        if randomize_file_order:
            h5_key = random.choice(h5_keys)
        else:
            h5_key = h5_keys[current_h5_key_index % len(h5_keys)]
            current_h5_key_index += 1

        h5_file = h5_files[h5_key]

        h5_data = utils.automap.read_h5_file(h5_file['h5'])

        image_sequence = h5_data['images']

        assert len(image_sequence.shape) == 3, 'Input must be have 3 d

        image_sequence = np.expand_dims(image_sequence, axis=-1) # mod

        magnitude_sequence = h5_data['magnitude']
        phase_sequence = h5_data['phase']
        fft_sequence = np.concatenate((np.expand_dims(magnitude_sequen
                                       np.expand_dims(phase_sequence, axis=3)),
                                      axis=3)

        # Contains info on positive/negative samples
        class_sequence = h5_data['classification']

        for i in range(len(fft_sequence)):

            # Only keep good (class == 1?) samples and skip the rest

            try:

                if len(image_sequence) != len(class_sequence):
                    # print("skipping, len(image_sequence) {} != len(c
                    continue

                if class_sequence[i] == 1:
```

```
                    try:
                        X_batch[current_batch_size, ...] = fft_sequenc
                        Y_batch[current_batch_size, ...] = image_seque
                    except Exception as e:
                        print('Error filling arrays of batch: {}'.form

                    current_batch_size += 1

                    if current_batch_size == batch_size:
                        current_batch_size = 0
                        yield X_batch, Y_batch

            except Exception as e:
                print('Unexpected error making batch:', e, class_seque
```

In [20]:

```
# With batch generator:

# Use first batch as validation data for now
X_development, Y_development = next(batch_generator())

sess = tf.Session(config=config)
```

```
Found 134 .h5 files and 134 .txt files.
```

In [21]:

```
# Uncomment to resume training form checkpoint
# model.load_weights('automap_baseline.h5')
```

In [ ]:

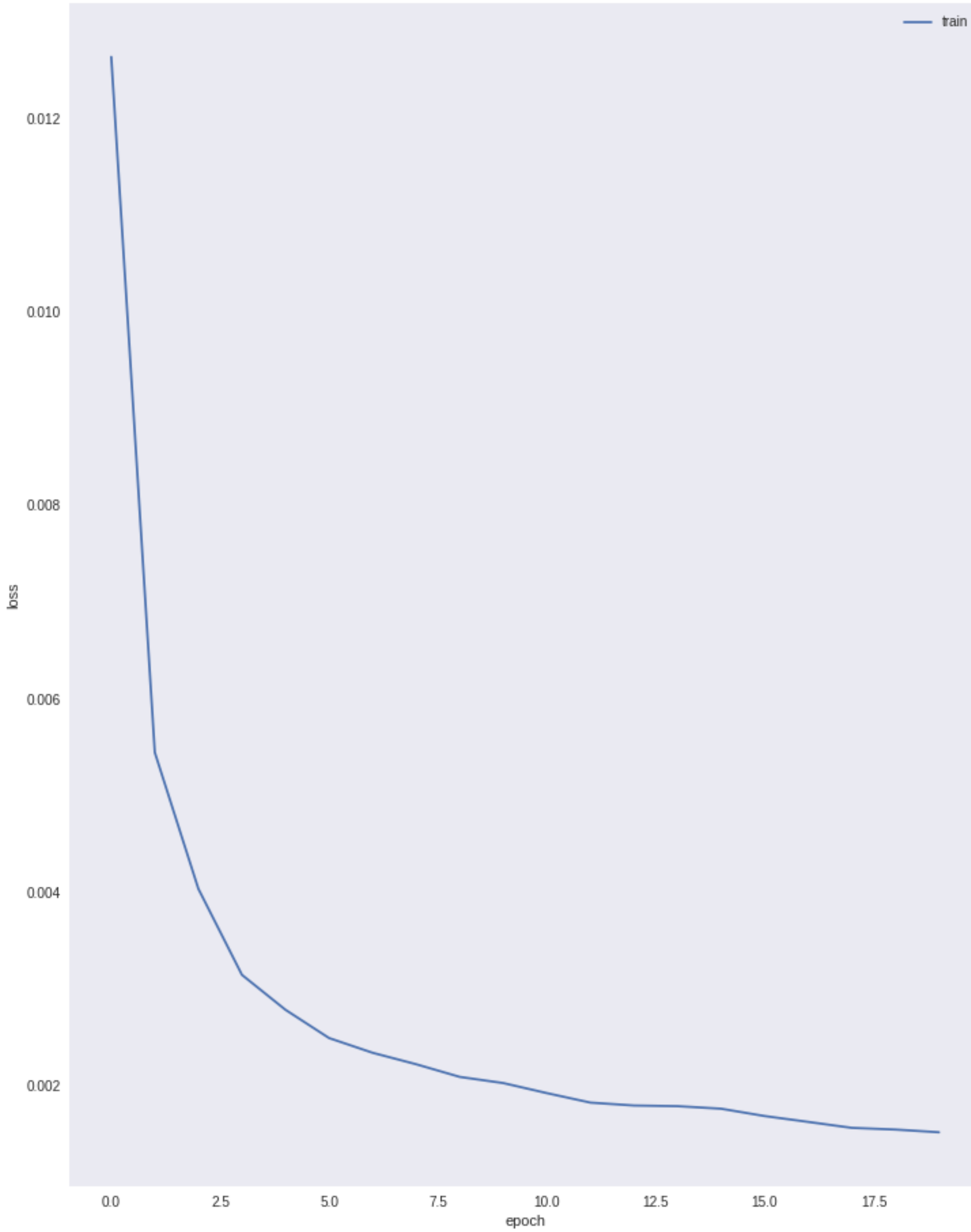
```
# Start training
with sess.as_default():
    fit_model = model.fit_generator(batch_generator(),
                                    validation_data=(X_development, Y_dev
                                    steps_per_epoch=2600 // 16,
                                    epochs=20,
                                    shuffle=True,
                                    verbose=1,
                                    use_multiprocessing=True)

    model.save('automap_keras_good_classes.h5')
```

In [33]:

```
# Visualize predictions
```

In [24]:
```
plt.plot(fit_model.history['loss'])
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train'])
```

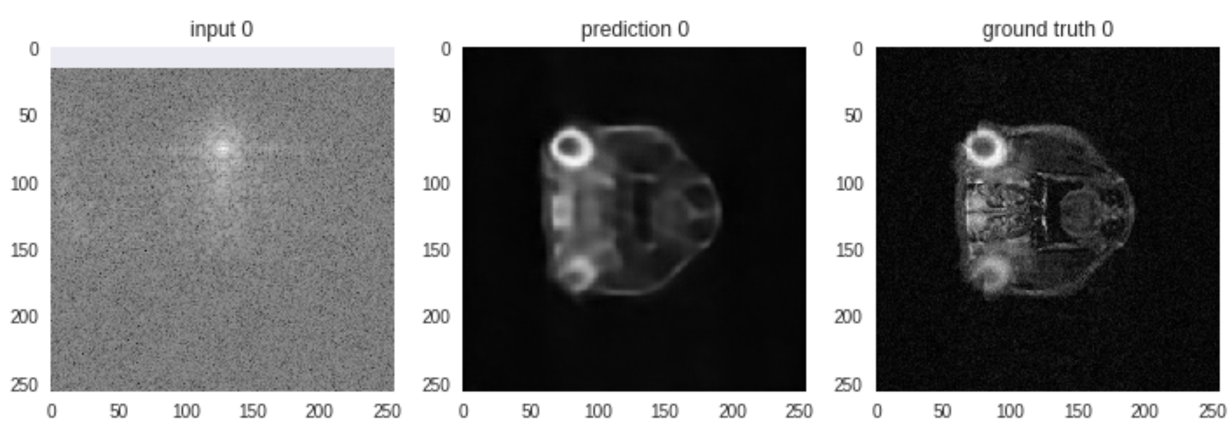
Out[24]: <matplotlib.legend.Legend at 0x7f62f04f95f8>



In [25]:
```
# Run predictions on development set
with sess.as_default():

    prediction_batch = model.predict(X_development)
    prediction_batch = prediction_batch.squeeze()

    ground_truth_batch = Y_development.squeeze()
```

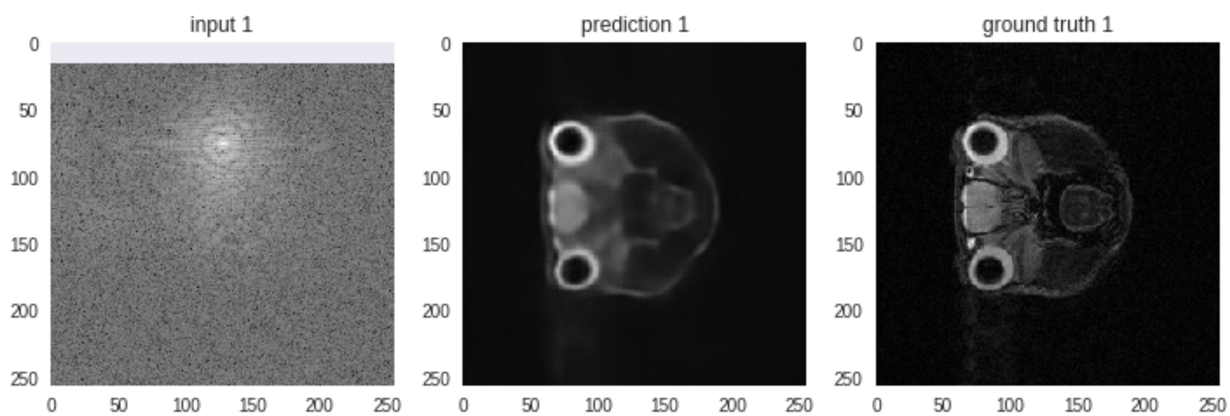
In [26]:
```
plt.rcParams['image.cmap'] = 'gray' # b/w images

# Prediction 0

mag, phase = X_development[0][..., 0],  X_development[0][..., 1]
fft = mag * np.exp(1j * phase)

# Uncomment to see reconstruction:
# plt.title('Reconstruction')
# utils.plot.imshowgray(np.abs(ifft))

plt.subplot(1, 3, 1)
utils.plot.imshowfft(np.abs(fft))
plt.title('input 0')

plt.subplot(1, 3, 2)
plt.imshow(prediction_batch[0])
plt.title('prediction 0')

plt.subplot(1, 3, 3)
plt.imshow(ground_truth_batch[0])
plt.title('ground truth 0');
```

```
/home/ubuntu/cs230/code/MRI_reconstruction_example/cs230_project_util
ities/plot.py:13: RuntimeWarning: divide by zero encountered in log10
  imshowgray(20 * np.log10(np.abs(fft)))
```



In [27]:
```
# Prediction 1

mag, phase = X_development[1][..., 0],  X_development[1][..., 1]
fft = mag * np.exp(1j * phase)

plt.subplot(1, 3, 1)
utils.plot.imshowfft(np.abs(fft))
plt.title('input 1')

plt.subplot(1, 3, 2)
plt.imshow(prediction_batch[1])
plt.title('prediction 1')

plt.subplot(1, 3, 3)
plt.imshow(ground_truth_batch[1])
plt.title('ground truth 1');
```

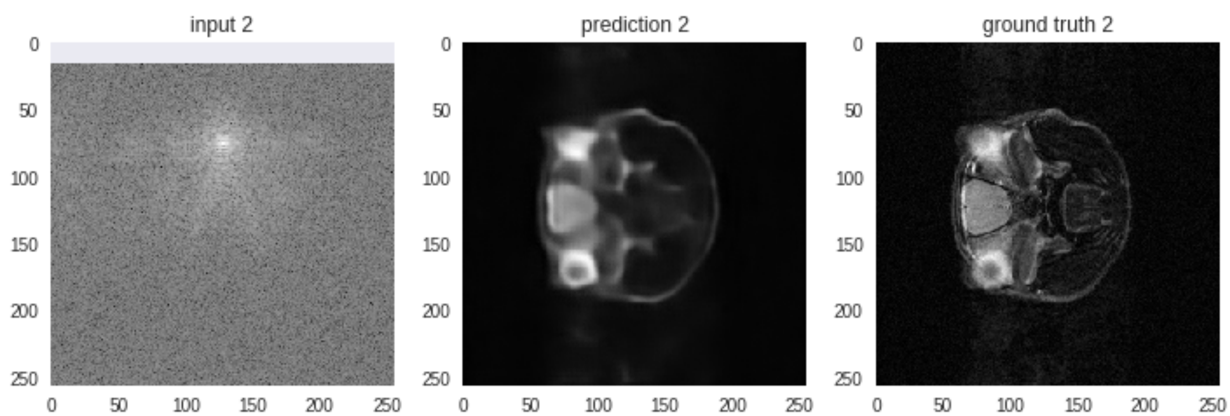
```
/home/ubuntu/cs230/code/MRI_reconstruction_example/cs230_project_util
ities/plot.py:13: RuntimeWarning: divide by zero encountered in log10
  imshowgray(20 * np.log10(np.abs(fft)))
```



In [28]:

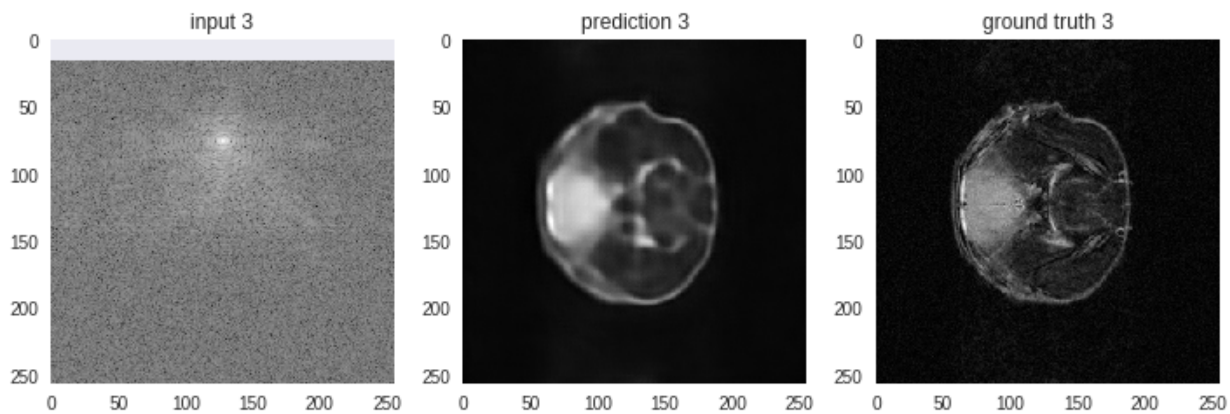
```
# Prediction 2

mag, phase = X_development[2][..., 0],  X_development[2][..., 1]
fft = mag * np.exp(1j * phase)

plt.subplot(1, 3, 1)
utils.plot.imshowfft(np.abs(fft))
plt.title('input 2')

plt.subplot(1, 3, 2)
plt.imshow(prediction_batch[2])
plt.title('prediction 2')

plt.subplot(1, 3, 3)
plt.imshow(ground_truth_batch[2])
plt.title('ground truth 2');
```

```
/home/ubuntu/cs230/code/MRI_reconstruction_example/cs230_project_util
ities/plot.py:13: RuntimeWarning: divide by zero encountered in log10
  imshowgray(20 * np.log10(np.abs(fft)))
```



In [29]:
```
# Prediction 3

mag, phase = X_development[3][..., 0],  X_development[3][..., 1]
fft = mag * np.exp(1j * phase)

plt.subplot(1, 3, 1)
utils.plot.imshowfft(np.abs(fft))
plt.title('input 3')

plt.subplot(1, 3, 2)
plt.imshow(prediction_batch[3])
plt.title('prediction 3')

plt.subplot(1, 3, 3)
plt.imshow(ground_truth_batch[3])
plt.title('ground truth 3');
```

```
/home/ubuntu/cs230/code/MRI_reconstruction_example/cs230_project_util
ities/plot.py:13: RuntimeWarning: divide by zero encountered in log10
  imshowgray(20 * np.log10(np.abs(fft)))
```



In [ ]:

In [ ]:

In [ ]: