FCN.py performance analysis

- Code breakdown

  - FCN.py workload is being slowed due to scipy code in training loop

```
def train_nn(sess, epochs, batch_size, get_batches_fn, train_op, cross_entropy_l
    keep_prob_value = 0.5

    learning_rate_value = 0.001

    for epoch in range(epochs):

        epoch_t0 = time.time()

        total_loss = 0
        for X_batch, gt_batch i  get_batches_fn(ba tch_size):

            loss, _ = sess.run([cross_entropy_loss, train_op],
            feed_dict={input_image: X_batch, correct_label: gt_batch,
            keep_prob: keep_prob_value, learning_rate:learning_rate_value})

            total_loss += loss;
        epoch_t1 = time.time()
```

```
def gen_batch_function(data_folder, image_shape):
    def get_batches_fn(batch_size):
        image_paths = glob(os.path.join(data_folder, 'image_2'
        label_paths = {
            re.sub(r'_(lane|road)_', '_', os.path.basename(pat
            for path in glob(os.path.join(data_folder, 'gt_ima

        background_color = np.array([255, 0, 0])

        random.shuffle(image_paths)

        for batch_i in range(0, len(image_paths), batch_size):
            images = []
            gt_images = []

            for image_file in image_paths[batch_i:batch_i+batc
                gt_imag _file = label_paths[os.path.basename(i
                image =  scipy.misc.imresize(scipy.misc.imread(

                gt_imag  = scipy.misc.imresize(scipy.misc.imre

                gt_bg = np.all(gt_image == background_color, a
                gt_bg = gt_bg.reshape(*gt_bg.shape, 1)

                gt_image = np.concatenate((gt_bg, np.invert(gt

                images.append(image)
                gt_images.append(gt_image)

            yield np.array(images), np.array(gt_images)
    return get_batches_fn
```

- Code modification for better performance
  - Read in data into preprocessed numpy array

```
t0 = time.time()
image_target = helper_dv.read_images_dv(training_dir, IMAGE_SHAPE)
print("##########################")
get_batches_fn = helper_dv.gen_batch_function_dv(image_target)
```

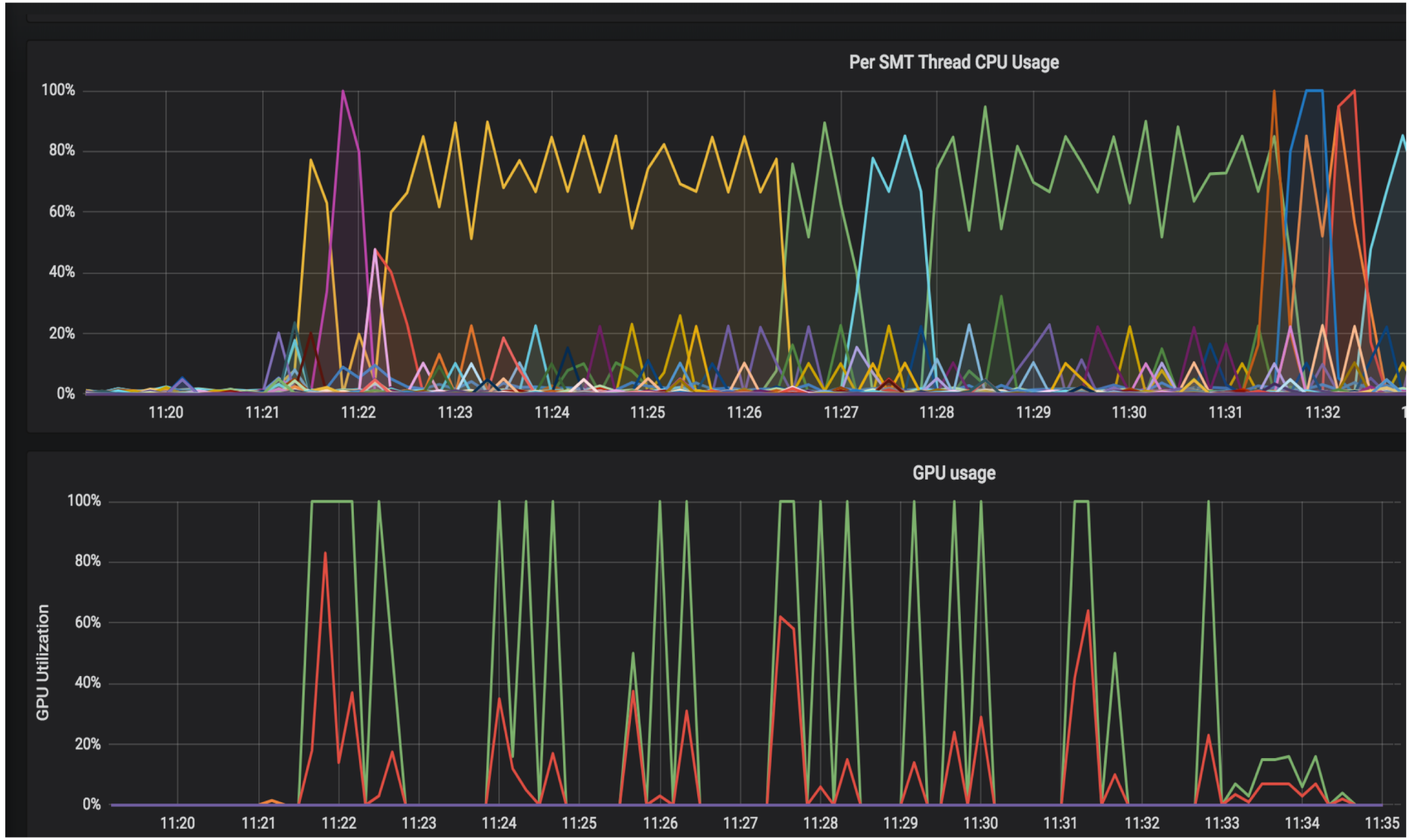
  - Get_batches_fn now just fetches data from numpy array, no preprocessing

```
for epoch in range(epochs):

    print("Timing epoch {}".format(epoch))
    epoch_t0 = time.time()
    total_loss = 0
    for X_batch, gt_batch in get_batches_fn(batch_size):

        loss, _ = sess.run([cross_entropy_loss, train_op],
        feed_dict={input_image: X_batch, correct_label: gt_batch,
        keep_prob: keep_prob_value, learning_rate:learning_rate_value}

        total_loss += loss;
    epoch_t1 = time.time()

    print("EPOCH {} ... {}".format(epoch + 1, epoch_t1 - epoch_t0))
    print("Loss = {:.3f}".format(total_loss))
    print()
```

```
def gen_batch_function_dv(image_target):
    def get_batches_fn_dv(batch_size):
        random.shuffle(image_target)
        #pdb.set_trace()

        for batch_i in range(0, len(image_target), batch_size):
            images = []
            gt_images = []

            for img_target_tuple in image_target[batch_i:batch_i+batch_size]:
                images.append(img_target_tuple[0])
                gt_images.append(img_target_tuple[1])

            yield np.array(images), np.array(gt_images)
    return get_batches_fn_dv
```

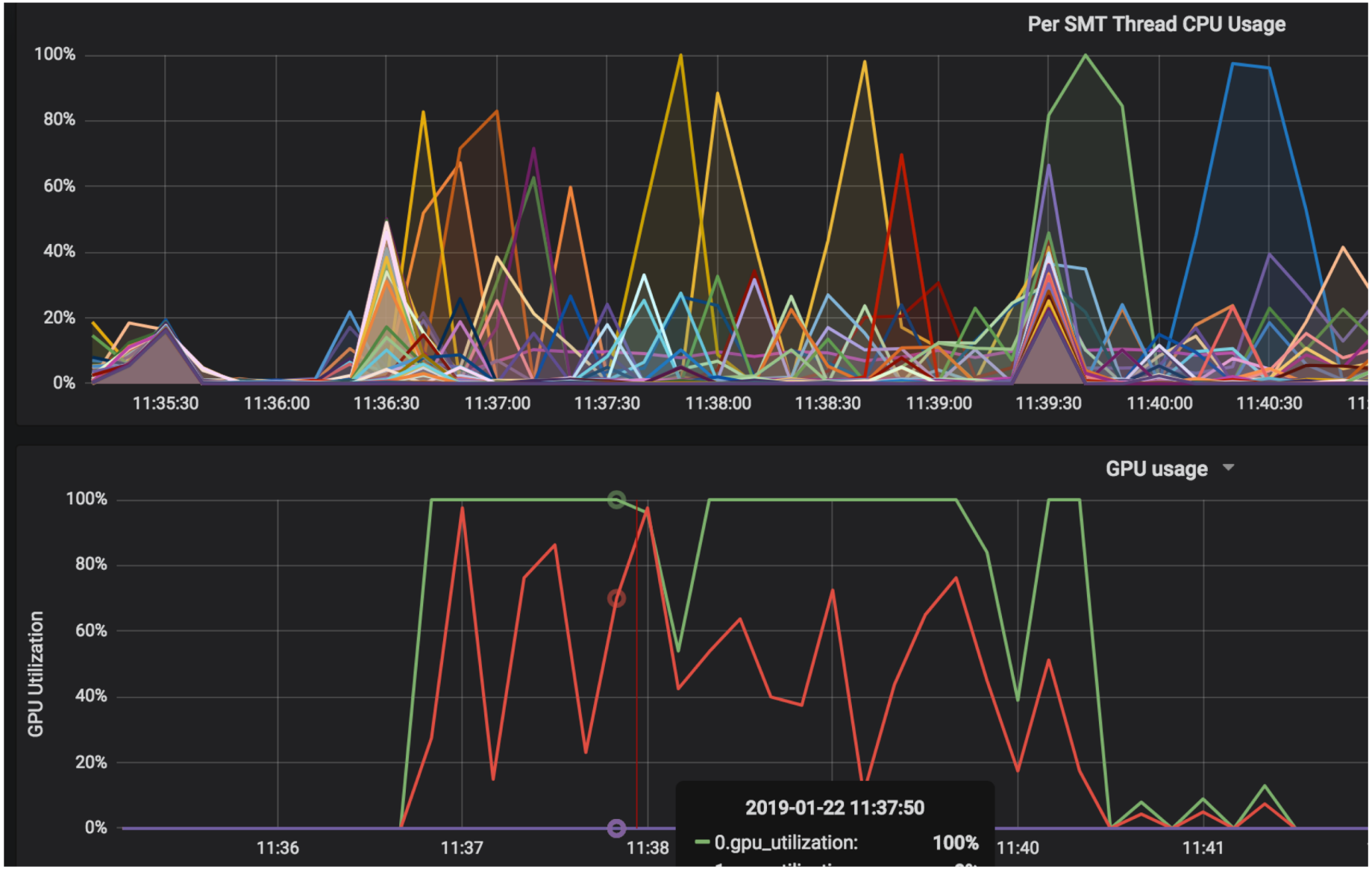
This results in 330s runtime vs 800s original implementation

Cpu / GPU trace (original FCN.py)



CPU preprocessing is not allowing GPU to be fully utilized

Cpu / GPU trace (optimized FCN_opt.py)



GPU now utilized better