Setting baseline results:

This document describes how to to get LB=0.82 using a simple "Cnn_Trad_Pool2_Net" from paper [1].
Implementation is in pytorch.

[References]

[1] "Convolutional Neural Networks for Small-footprint Keyword Spotting" - Tara N. Sainath, Carolina Parada, INTERSPEECH 2015

[2] https://research.googleblog.com/2017/08/launching-speech-commands-dataset.html

[3] https://github.com/castorini/honk

1. Network structure

```
35  class Cnn_Trad_Pool2_Net(nn.Module):
36      def __init__(self, in_shape=(1,40,101), num_classes=12 ):
37          super(Cnn_Trad_Pool2_Net, self).__init__()
38          self.num_classes = num_classes
39
40          self.conv1 = nn.Conv2d(1,  64, kernel_size=(20, 8), stride=(1, 1))
41          self.conv2 = nn.Conv2d(64, 64, kernel_size=(10, 4), stride=(1, 1))
42          self.fc = nn.Linear(26624,num_classes)
43
44
45      def forward(self, x):
46
47          x = self.conv1(x)
48          x = F.relu(x,inplace=True)
49          x = F.max_pool2d(x,kernel_size=(2,2),stride=(2,2))
50
51          x = self.conv2(x)
52          x = F.relu(x,inplace=True)
53          x = x.view(x.size(0), -1)
54
55          x = F.dropout(x,p=0.5,training=self.training)
56          x = self.fc(x)
57
58          return x  #logits
```

2. Input (melspectrogram/MFCC)

```
174  def tf_wave_to_mfcc(wave):
175
176      spectrogram = librosa.feature.melspectrogram(wave, sr=AUDIO_SR, n_mels=40, hop_length=160, n_fft=480, fmin=20, fmax=4000)
177      #spectrogram = librosa.power_to_db(spectrogram)
178      idx = [spectrogram > 0]
179      spectrogram[idx] = np.log(spectrogram[idx])
180
181      dct_filters = librosa.filters.dct(n_filters=40, n_input=40)
182      mfcc = [np.matmul(dct_filters, x) for x in np.split(spectrogram, spectrogram.shape[1], axis=1)]
183      mfcc = np.hstack(mfcc)
184      mfcc = mfcc.astype(np.float32)
185
186      return mfcc
```

```
def tf_wave_to_melspectrogram(wave):
    spectrogram = librosa.feature.melspectrogram(wave, sr=AUDIO_SR, n_mels=40, hop_length=160, n_fft=480, fmin=20, fmax=4000)
    spectrogram = librosa.power_to_db(spectrogram)
    spectrogram = spectrogram.astype(np.float32)

    return spectrogram
```

2. Dataset

```python
AUDIO_DIR = '/root/share/project/kaggle/tensorflow/data'
AUDIO_NUM_CLASSES = 12
AUDIO_NAMES =[ 'silence', 'unknown', 'yes', 'no', 'up', 'down', 'left', 'right', 'on', 'off', 'stop', 'go']
assert(AUDIO_NUM_CLASSES==len(AUDIO_NAMES))

AUDIO_SR     = 16000 #sampling rate
AUDIO_LENGTH = 16000
sd.default.samplerate = AUDIO_SR

AUDIO_SILENCE = \
    librosa.core.load( AUDIO_DIR + '/silence.wav', sr=AUDIO_SR)[0]

AUDIO_NOISES=[]
for file in ['dude_miaowing.wav',  'pink_noise.wav',   'white_noise.wav', 'doing_the_dishes.wav',
             'exercise_bike.wav',  'running_tap.wav']:
    audio_file = AUDIO_DIR + '/train/audio/_background_noise_/' + file
    wave = librosa.core.load(audio_file, sr=AUDIO_SR)[0]
    AUDIO_NOISES.append(wave)
```

this is a "zero array" audio file

split is a list of file paths,

train_train_51088

```
train/audio/tree/17c94b23_nohash_0.wav
train/audio/tree/4f2ab70c_nohash_0.wav
train/audio/tree/c3538de1_nohash_1.wav
train/audio/tree/fb9d6d23_nohash_2.wav
train/audio/tree/9385508d_nohash_0.wav
train/audio/tree/d0426d63_nohash_0.wav
train/audio/tree/b665723d_nohash_0.wav
train/audio/tree/26e9ae6b_nohash_0.wav
train/audio/tree/61e50f62_nohash_0.wav
train/audio/tree/afb9e62e_nohash_0.wav
train/audio/tree/ccfd721c_nohash_2.wav
train/audio/tree/b59fa113_nohash_0.wav
train/audio/tree/a0f93943_nohash_0.wav
train/audio/tree/8012c69d_nohash_0.wav
train/audio/tree/ae04cdbe_nohash_1.wav
train/audio/tree/102192fd_nohash_1.wav
train/audio/tree/a1c63f25_nohash_0.wav
train/audio/tree/079dfce3_nohash_0.wav
train/audio/tree/b3bdded5_nohash_0.wav
```

```python
# #data iterator ---------------------------------------------------------------
class AudioDataset(Dataset):

    def __init__(self, split, transform=None, mode='train'):
        super(AudioDataset, self).__init__()
        start = timer()

        self.split = split
        self.transform = transform
        self.mode = mode

        #label
        label_to_name = dict(zip(list(range(AUDIO_NUM_CLASSES)), AUDIO_NAMES))
        name_to_label = dict(zip( AUDIO_NAMES,list(range(AUDIO_NUM_CLASSES))))

        #read split
        lines = read_list_from_file(AUDIO_DIR + '/split/' + split, comment='#')
        num_classes = len(AUDIO_NAMES)
        ids=[]
        labels=[]
        index_by_class=[]
        for line in lines:
            ids.append(line)

        if mode in ['train']:
            for line in lines:
                name = line.split('/')[2]
                if name not in AUDIO_NAMES:
                    #continue
                    name='unknown'
                labels.append(name_to_label[name])

            for i in range(num_classes):
                index =np.where(np.array(labels)==i)[0]
                index_by_class.append(list(index))

#divide sample index by class

        #save
        self.index_by_class = index_by_class
        self.ids    = ids
        self.labels = labels

        #print
        print('\ttime = %0.2f min'%((timer() - start) / 60))
        print('\tnum_ids = %d'%(len(self.ids)))
        print('\tnum_classes = %d'%(num_classes))
        if mode in ['train']:
            for i in range(num_classes):
                print('\t\t%2d   %16s = %5d (%0.3f)'%(i,AUDIO_NAMES[i],len(index_by_class[i]),len(index_by_class[i])/len(sel
        print('')


    def __getitem__(self, index):

        label = None

        if index==-1: #silence
            wave = AUDIO_SILENCE

            if self.mode in ['train']:
                label = 0
        else:
            audio_file = AUDIO_DIR + '/' + self.ids[index]
            wave = librosa.core.load(audio_file, sr=AUDIO_SR)[0]

            if self.mode in ['train']:
                label = self.labels[index]


        if self.transform is not None:
            return self.transform(wave, label, index)
        else:
            return wave, label, index


    def __len__(self):
        return len(self.ids)
```

sampler for given silence and unknow samples

```
class TFRandomSampler(Sampler):
    def __init__(self, data, silence_probability=0.1, unknown_probability=0.1):
        self.data = data
        self.silence_probability = silence_probability
        self.unknown_probability = unknown_probability
        self.known_probability   = 1-silence_probability-unknown_probability

        known_num = 0
        for i in range(2,AUDIO_NUM_CLASSES):
            known_num += len(data.index_by_class[i])

        self.known_num   = known_num
        self.silence_num = int((self.known_num/self.known_probability)*self.silence_probability)
        self.unknown_num = int((self.known_num/self.known_probability)*self.unknown_probability)
        self.length = self.silence_num + self.unknown_num + self.known_num

    def __iter__(self):
        data = self.data
        l  = []
        if self.silence_num>0:#silence
            #empty (index is -1)
            silence_list = ([-1]+ data.index_by_class[0])*math.ceil(self.silence_num/(1+len(data.index_by_class[0])))
            random.shuffle(silence_list)
            silence_list = silence_list[:self.silence_num]
            l += silence_list

        if self.unknown_num>0:#unknown
            unknown_list = data.index_by_class[1]*math.ceil(self.unknown_num/len(data.index_by_class[1]))
            random.shuffle(unknown_list)
            unknown_list = unknown_list[:self.unknown_num]
            l +=  unknown_list

        for i in range(2,AUDIO_NUM_CLASSES):
            l +=  data.index_by_class[i]

        assert(len(l)==self.length)
        random.shuffle(l)
        return iter(l)
```

epoch epoches uses different silence and unknow samples

augmentation

```python
def tf_random_add_noise_transform(wave, noise_limit=0.2, u=0.5):

    if random.random() < u:
        num_noises = len(AUDIO_NOISES)
        noise = AUDIO_NOISES[np.random.choice(num_noises)]

        wave_length  = len(wave)
        noise_length = len(noise)
        t = np.random.randint(0, noise_length - wave_length - 1)
        noise = noise[t:t + wave_length]

        alpha = np.random.random() * noise_limit
        wave  = np.clip(alpha * noise + wave, -1, 1)

    return wave


def tf_random_time_shift_transform(wave, shift_limit=0.2, u=0.5):
    if random.random() < u:
        wave_length  = len(wave)
        shift_limit = shift_limit*wave_length
        shift = np.random.randint(-shift_limit, shift_limit)
        t0 = -min(0, shift)
        t1 =  max(0, shift)
        wave = np.pad(wave, (t0, t1), 'constant')
        wave = wave[:-t0] if t0 else wave[t1:]

    return wave
```

```python
def tf_random_pad_transform(wave, length=AUDIO_LENGTH):

    if len(wave)<AUDIO_LENGTH:
        L = abs(len(wave)-AUDIO_LENGTH)
        start = np.random.choice(L)
        wave  = np.pad(wave, (start, L-start), 'constant')

    elif len(wave)>AUDIO_LENGTH:
        L = abs(len(wave)-AUDIO_LENGTH)
        start = np.random.choice(L)
        wave  = wave[start: start+AUDIO_LENGTH]

    return wave


def tf_fix_pad_transform(wave, length=AUDIO_LENGTH):
    # wave = np.pad(wave, (0, max(0, AUDIO_LENGTH - len(wave))), 'constant')
    # return wave

    if len(wave)<AUDIO_LENGTH:
        L = abs(len(wave)-AUDIO_LENGTH)
        start = L//2
        wave  = np.pad(wave, (start, L-start), 'constant')

    elif len(wave)>AUDIO_LENGTH:
        L = abs(len(wave)-AUDIO_LENGTH)
        start = L//2
        wave  = wave[start: start+AUDIO_LENGTH]

    return wave
```

3. Training (SGD)

```
from net.model.cnn_trad_pool2_net import load_pretrain_file, Cnn_Trad_Pool2_Net as Net
```

```
#--------------------------------------------------------------
def run_train():

    out_dir  = '/root/share/project/kaggle/tensorflow/results/cnn_trad_pool2_net-12' # s_x
    initial_checkpoint = \
        None #'/root/share/project/kaggle/tensorflow/results/cnn_trad_pool2_net-02/checkpo
        #

    pretrain_file = None

    ## setup  ----------------------------
    os.makedirs(out_dir +'/checkpoint', exist_ok=True)
    os.makedirs(out_dir +'/backup', exist_ok=True)
    backup_project_as_zip(PROJECT_PATH, out_dir +'/backup/code.train.%s.zip'%IDENTIFIER)
```

```
    ## dataset ----------------------------------------
    log.write('** dataset setting **\n')
    train_dataset = AudioDataset(
                               #'train_trainvalid_57886', mode='train',
                               'train_train_51088',  mode='train',
                               transform = train_augment)
    train_loader  = DataLoader(
                        train_dataset,
                        #sampler = TFRandomSampler(train_dataset,0.1,0.1),
                        sampler = TFRandomSampler(train_dataset,0.1,0.1),
                        batch_size  = batch_size,
                        drop_last   = True,
                        num_workers = 6,
                        pin_memory  = True,
                        collate_fn  = collate)

    valid_dataset = AudioDataset(
                               #'train_valid_6798',  mode='train',
                               'train_test_6835',  mode='train',
                                transform = valid_augment)
    valid_loader  = DataLoader(
                        valid_dataset,
                        sampler     = TFSequentialSampler(valid_dataset,0.1,0.1),
                        batch_size  = batch_size,
                        drop_last   = False,
                        num_workers = 6,
                        pin_memory  = True,
                        collate_fn  = collate)
```

random sampling with given silence and unknown probabilities

fix sampling with given silence and unknown probabilities

train/valid augmentation

```
def train_augment(wave,label,index):
    wave = tf_random_time_shift_transform(wave, shift_limit=0.2, u=0.5)
    wave = tf_random_add_noise_transform (wave, noise_limit=0.2, u=0.5)
    wave = tf_random_pad_transform(wave)

    #tensor = tf_wave_to_melspectrogram(wave)[np.newaxis,:]
    tensor = tf_wave_to_mfcc(wave)[np.newaxis,:]
    tensor = torch.from_numpy(tensor)
    return tensor,label,index


def valid_augment(wave,label,index):
    wave = tf_fix_pad_transform(wave)

    #tensor = tf_wave_to_melspectrogram(wave)[np.newaxis,:]
    tensor = tf_wave_to_mfcc(wave)[np.newaxis,:]
    tensor = torch.from_numpy(tensor)
    return tensor,label,index
```

```
## net ---------------------------- -
log.write('** net setting **\n')
net = Net(in_shape = (1,40,101), num_classes=AUDIO_NUM_CLASSES).cuda()
```

```
## optimiser ----------------------------------
iter_accum  = 1
batch_size  = 128

num_iters   = 1000  *1000
iter_smooth = 20
iter_log    = 500
iter_valid  = 500
iter_save   = [0, num_iters-1]\
              + list(range(0,num_iters,1000))#1*1000

#LR = StepLR([ (0, 0.01),  (200, 0.001),  (300, -1)])
LR = None #StepLR([ (0, 0.01),])
optimizer = optim.SGD(filter(lambda p: p.requires_grad, net.parameters()),
                      lr=0.005/iter_accum, momentum=0.9, weight_decay=0.0001)
```

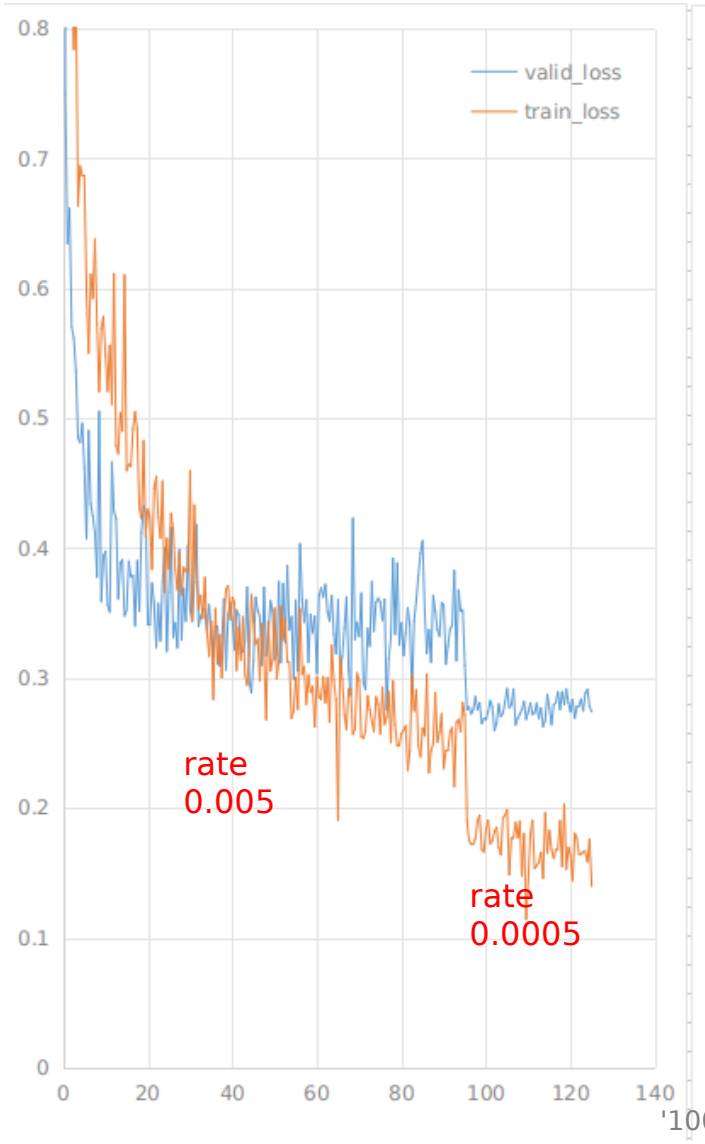
optimizer

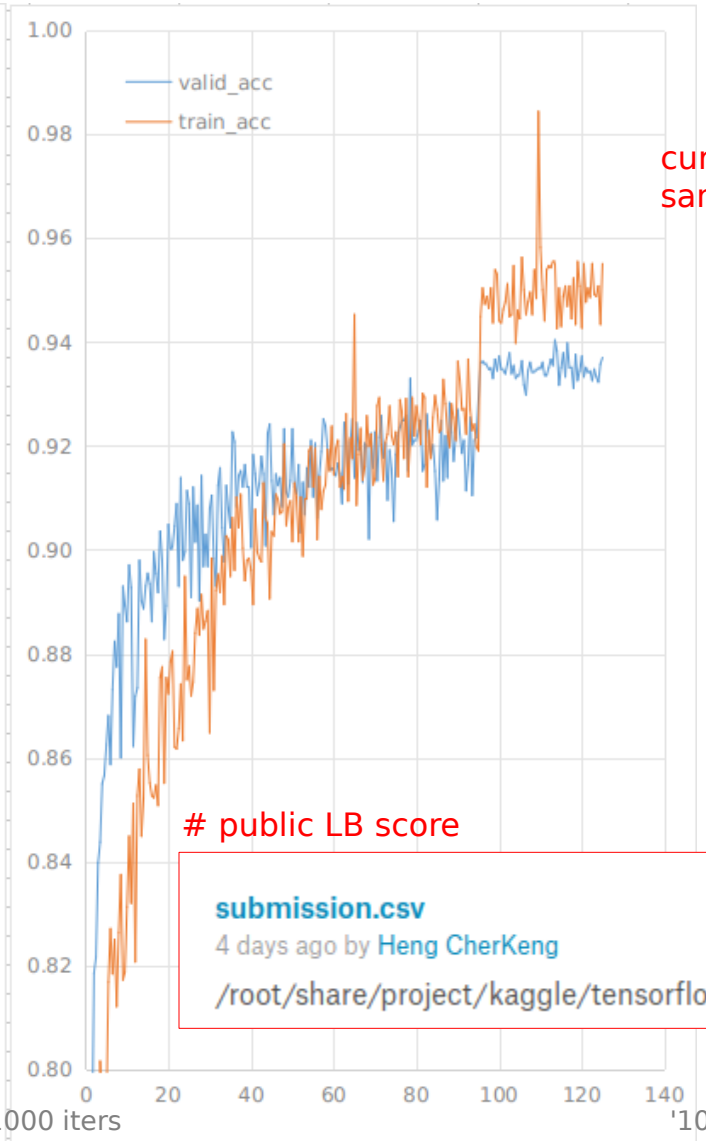
```
while  i<num_iters:  # loop over the dataset multiple times
    sum_train_loss = 0.0
    sum_train_acc  = 0.0
    sum = 0

    net.train()
    optimizer.zero_grad()
    for tensors, labels, indices in train_loader:
        i = j/iter_accum + start_iter
        epoch = (i-start_iter)*batch_size*iter_accum/len(tr
```

```
    # learning rate schduler -------------
    if LR is not None:
        lr = LR.get_rate(i)
        if lr<0 : break
        adjust_learning_rate(optimizer, lr/iter_accum)
    rate = get_learning_rate(optimizer)[0]*iter_accum


    # one iteration update  -------------
    tensors = Variable(tensors).cuda()
    labels  = Variable(labels).cuda()
    logits  = data_parallel(net, tensors)
    probs   = F.softmax(logits,dim=1)
    loss    = F.cross_entropy(logits, labels)
    acc     = top_accuracy(probs, labels, top_k=(1,))

    # accumulated update
    loss.backward()
    if j%iter_accum == 0:
        #torch.nn.utils.clip_grad_norm(net.parameters(), 1)
        optimizer.step()
        optimizer.zero_grad()
```

4. Example Results

/root/share/project/kaggle/tensorflow/results/cnn_trad_pool2_net-02a





curve are "spiky" becuase the silence and unknown samples are changed per epoch

public LB score

submission.csv 0.82

4 days ago by Heng CherKeng

/root/share/project/kaggle/tensorflow/results/cnn_trad_pool2_net-02a/submit

example log file

```
<class 'net.model.cnn_trad_pool2_net.Cnn_Trad_Pool2_Net'>


** dataset setting **
    train_dataset.split = train_train_51088
    valid_dataset.split = train_test_6835
    len(train_dataset)  = 51088
    len(valid_dataset)  = 6835
    len(train_loader)   = 181
    len(valid_loader)   = 26
    batch_size  = 128
    iter_accum  = 1
    batch_size*iter_accum  = 128


** start training here! **
 optimizer=<torch.optim.sgd.SGD object at 0x7f58b40ef518>
 momentum=0.900000
 LR=None

 waves_per_epoch = 51088

   rate   iter_k   epoch  num_m| valid_loss/acc | train_loss/acc | batch_loss/acc |  time
-------------------------------------------------------------------------------------------
0.0000    0.0 k   0.00   0.0 | 2.5259  0.0939 | 0.0000  0.0000 | 0.0000  0.0000 |  0 hr 00 min
0.0050    0.5 k   1.25   0.1 | 0.7491  0.7621 | 1.1497  0.6445 | 1.0478  0.6797 |  0 hr 00 min
0.0050    1.0 k   2.51   0.1 | 0.6345  0.7983 | 0.9155  0.7180 | 0.9168  0.7422 |  0 hr 01 min
0.0050    1.5 k   3.76   0.2 | 0.6615  0.7795 | 0.8521  0.7520 | 1.0927  0.6641 |  0 hr 01 min
0.0050    2.0 k   5.01   0.3 | 0.5703  0.8185 | 0.8371  0.7336 | 0.8761  0.7266 |  0 hr 02 min
0.0050    2.5 k   6.26   0.3 | 0.5618  0.8216 | 0.7841  0.7629 | 0.6984  0.7969 |  0 hr 03 min
0.0050    3.0 k   7.52   0.4 | 0.5366  0.8397 | 0.8350  0.7348 | 0.8584  0.7422 |  0 hr 03 min
0.0050    3.5 k   8.77   0.4 | 0.4847  0.8438 | 0.6636  0.8016 | 0.6819  0.7969 |  0 hr 04 min
0.0050    4.0 k  10.02   0.5 | 0.4815  0.8550 | 0.6942  0.7949 | 0.5046  0.7891 |  0 hr 04 min
0.0050    4.5 k  11.27   0.6 | 0.4961  0.8566 | 0.6867  0.7816 | 0.8565  0.7578 |  0 hr 05 min
0.0050    5.0 k  12.53   0.6 | 0.4621  0.8625 | 0.6866  0.7910 | 0.7025  0.7500 |  0 hr 06 min
0.0050    5.5 k  13.78   0.7 | 0.4074  0.8681 | 0.5988  0.8168 | 0.9045  0.7422 |  0 hr 06 min
0.0050    6.0 k  15.03   0.8 | 0.4907  0.8587 | 0.5502  0.8271 | 0.5604  0.8203 |  0 hr 07 min
0.0050    6.5 k  16.29   0.8 | 0.4343  0.8731 | 0.6109  0.8184 | 0.5596  0.8281 |  0 hr 07 min
0.0050    7.0 k  17.54   0.9 | 0.4247  0.8824 | 0.5922  0.8250 | 0.5160  0.8438 |  0 hr 08 min
0.0050    7.5 k  18.79   1.0 | 0.4113  0.8775 | 0.6378  0.8121 | 0.5519  0.8359 |  0 hr 09 min
0.0050    8.0 k  20.04   1.0 | 0.3777  0.8877 | 0.5723  0.8263 | 0.6090  0.8125 |  0 hr 09 min
0.0050    8.5 k  21.30   1.1 | 0.5053  0.8600 | 0.5206  0.8375 | 0.6299  0.7891 |  0 hr 10 min
0.0050    9.0 k  22.55   1.2 | 0.3589  0.8930 | 0.5691  0.8172 | 0.3635  0.8594 |  0 hr 11 min
0.0050    9.5 k  23.80   1.2 | 0.3939  0.8893 | 0.5783  0.8187 | 0.5067  0.8359 |  0 hr 11 min
0.0050   10.0 k  25.05   1.3 | 0.3976  0.8862 | 0.5482  0.8313 | 0.6009  0.8203 |  0 hr 12 min
0.0050   10.5 k  26.31   1.3 | 0.3564  0.8971 | 0.5208  0.8449 | 0.7246  0.8594 |  0 hr 12 min
0.0050   11.0 k  27.56   1.4 | 0.3510  0.8927 | 0.5562  0.8320 | 0.4671  0.8516 |  0 hr 13 min
0.0050   11.5 k  28.81   1.5 | 0.4662  0.8622 | 0.5107  0.8512 | 0.4238  0.8594 |  0 hr 14 min
```