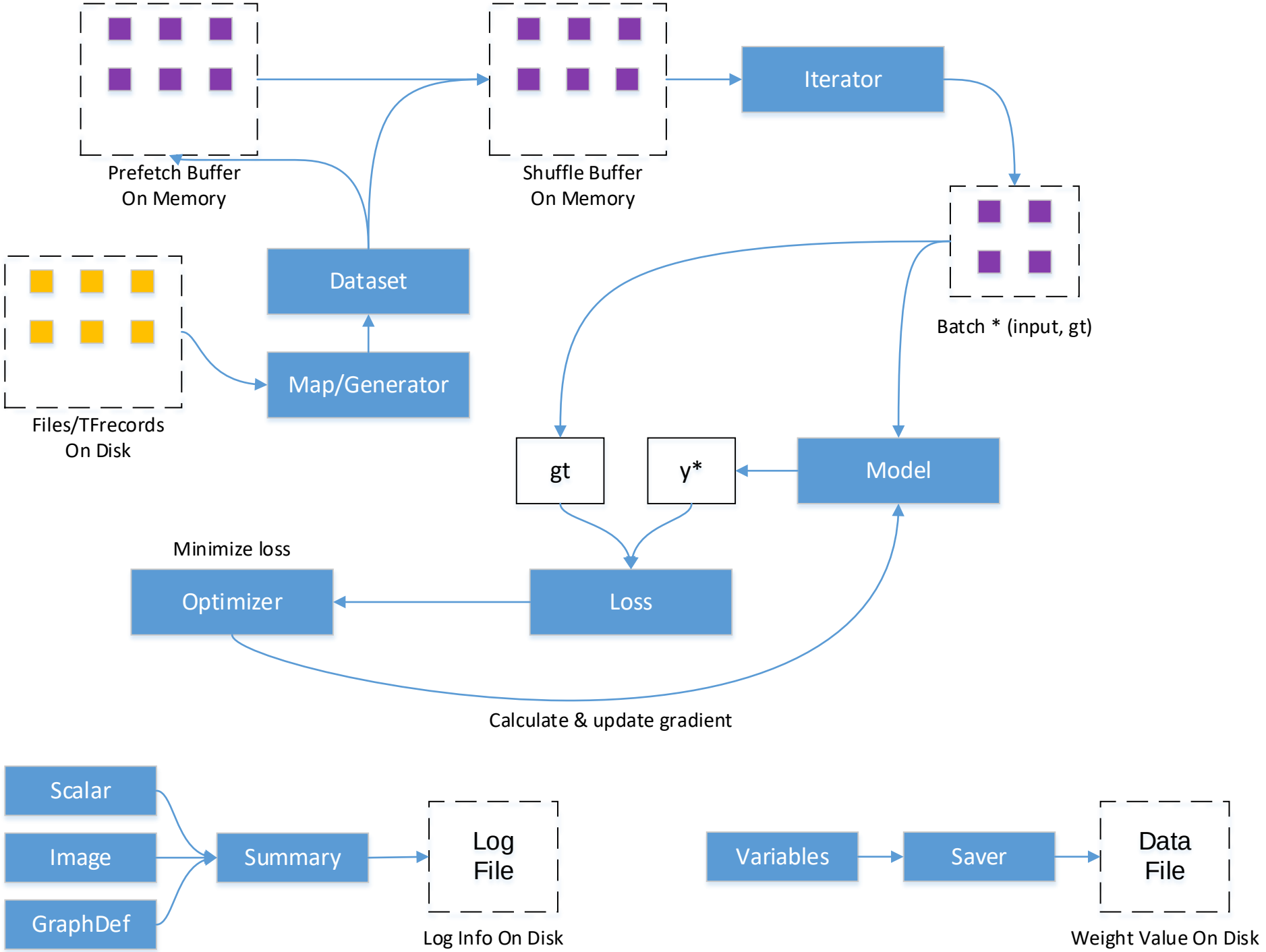
模型训练与测试

Tensorflow

林楚铭

cmlin17@fudan.edu.cn


Prefetch Buffer
On Memory
Shuffle Buffer
On Memory
Iterator
Dataset
Files/TFrecords
On Disk
Map/Generator
Batch * (input, gt)
gt
y*
Model
Minimize loss
Optimizer
Loss
Calculate & update gradient
Scalar
Image
GraphDef
Summary
Log
File
Log Info On Disk
Variables
Saver
Data
File
Weight Value On Disk

简介

- 结合之前的数据读取与模型建立模块，进行模型的训练
- 使用Adam优化器进行训练
- 训练中可以定时保存训练结果
- 训练过程可视化
- 将训练好的模型在测试集上测试

基础参数与准备

```
MODEL = 'VDSR'  # 'VDSR' or 'EDSR'
TRAIN_DIR = 'output/{}/model'.format(MODEL)
LOG_DIR = 'output/{}/log'.format(MODEL)
BATCH_SIZE = 64
SHUFFLE_NUM = 20000
PREFETCH_NUM = 10000
MAX_TRAIN_STEP = 50000
LR_BOUNDS = [45000]
LR_VALS = [1e-4, 1e-5]
SAVE_PER_STEP = 2000
TRAIN_PNG_PATH = 'DIV2K/DIV2K_train_HR'
TRAIN_TFRECORD_PATH = 'DIV2K/tfrecords'
DATA_LOADER_MODE = 'RAW'  # 'TFRECORD' or 'RAW'
DEVICE_MODE = 'GPU'  # 'CPU' or 'GPU'
DEVICE_GPU_ID = '0'

if not os.path.exists(TRAIN_DIR):
    os.makedirs(TRAIN_DIR)
if not os.path.exists(LOG_DIR):
    os.makedirs(LOG_DIR)

if DEVICE_MODE == 'CPU':
    os.environ['CUDA_VISIBLE_DEVICES'] = '-1'
else:
    os.environ['CUDA_VISIBLE_DEVICES'] = DEVICE_GPU_ID
```

配置数据与模型

```
if MODEL == 'VDSR':
    model = VDSR()
else:
    model = EDSR()

data_loader = DataLoader(data_dir=TRAIN_PNG_PATH,
                         batch_size=BATCH_SIZE,
                         shuffle_num=SHUFFLE_NUM,
                         prefetch_num=PREFETCH_NUM)

if DATA_LOADER_MODE == 'TFRECORD':
    if len(os.listdir(TRAIN_TFRECORD_PATH)) == 0:
        data_loader.gen_tfrecords(TRAIN_TFRECORD_PATH)
    lrs, bics, gts = data_loader.read_tfrecords(TRAIN_TFRECORD_PATH)
else:
    lrs, bics, gts = data_loader.read_pngs()

res = model(lrs, bics)
```

配置训练操作

```
with tf.name_scope('train'):
    global_step = tf.Variable(0, trainable=False, name='global_step')
    mse_loss = tf.reduce_sum(tf.square(res - gts)) / BATCH_SIZE
    reg_loss = tf.losses.get_regularization_loss()
    loss = mse_loss + reg_loss

    learning_rate = tf.train.piecewise_constant(global_step, LR_BOUNDS, LR_VALS)

    optimizer = tf.train.AdamOptimizer(learning_rate)
    train_op = optimizer.minimize(loss, global_step)
```

配置训练可视化操作

```
with tf.name_scope('summaries'):
    tf.summary.scalar('learning_rate', learning_rate)
    tf.summary.scalar('mse_loss', mse_loss)
    tf.summary.scalar('reg_loss', reg_loss)
    tf.summary.scalar('loss', loss)

    tf.summary.image('lr', lrs, 1)
    tf.summary.image('bic', model.bic, 1)
    tf.summary.image('out', tf.clip_by_value(res, 0, 1), 1)
    tf.summary.image('gt', gts, 1)

    summary_op = tf.summary.merge_all()
```

Session启动训练

```python
saver = tf.train.Saver(max_to_keep=500)
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
sess = tf.Session(config=config)
sess.run(tf.global_variables_initializer())

restore_session_from_checkpoint(sess, saver)

start_time = datetime.datetime.now()
writer = tf.summary.FileWriter(LOG_DIR, sess.graph)

while True:
    _, loss_value, step = sess.run([train_op, loss, global_step])
    if step % 20 == 0:
        end_time = datetime.datetime.now()
        print('[{}] Step:{}, loss:{}'.format(
            end_time - start_time, step, loss_value
        ))
        summary_value = sess.run(summary_op)
        writer.add_summary(summary_value, step)
        start_time = end_time
    if step % SAVE_PER_STEP == 0:
        saver.save(sess, os.path.join(TRAIN_DIR, 'checkpoint.ckpt'), global_step=step)
    if step >= MAX_TRAIN_STEP:
        print('Done train.')
        break
```

```python
def restore_session_from_checkpoint(sess, saver):
    checkpoint = tf.train.latest_checkpoint(TRAIN_DIR)
    if checkpoint:
        saver.restore(sess, checkpoint)
        return True
    else:
        return False
```

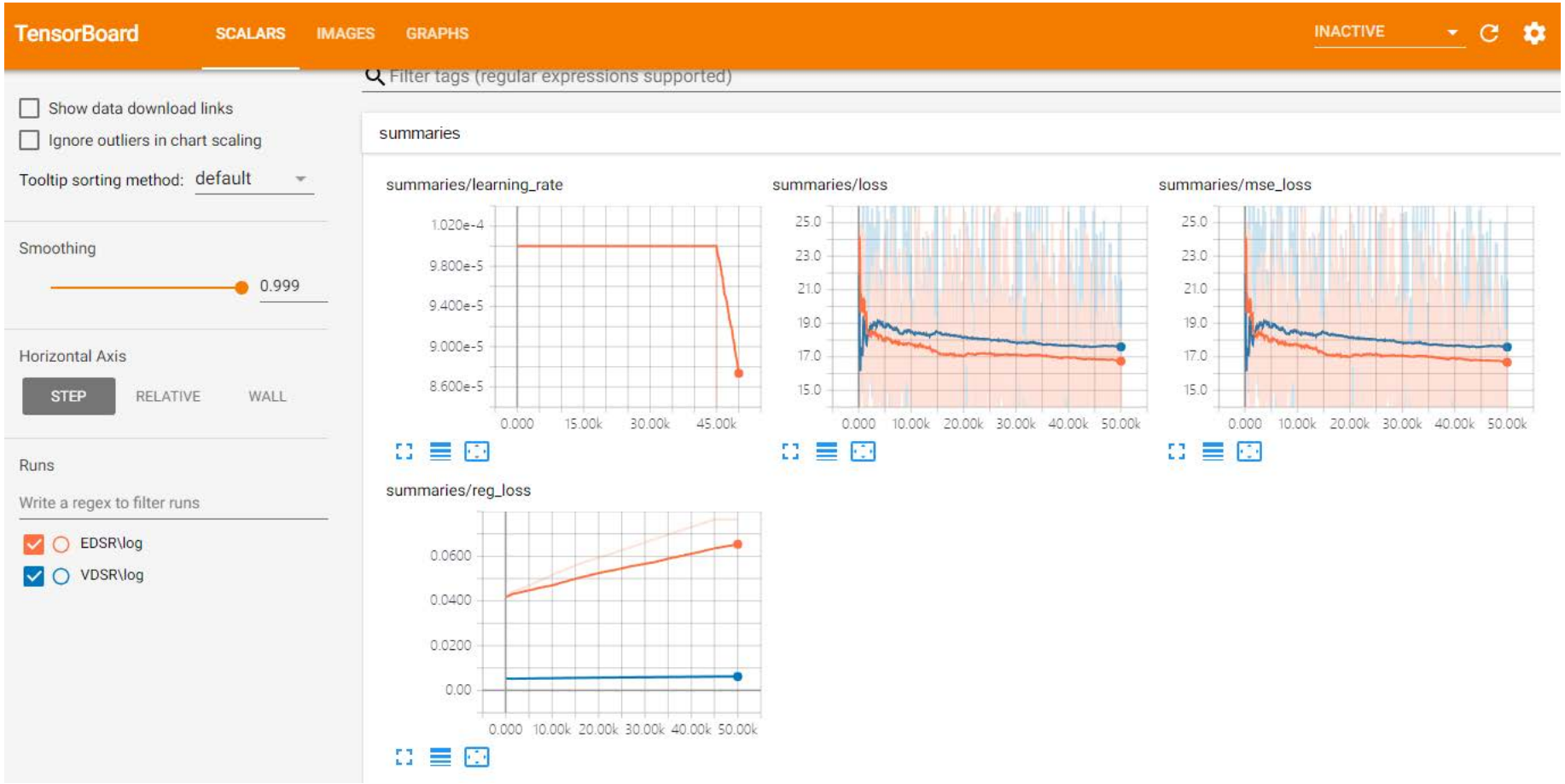
名称	修改日期	类型	大小
checkpoint	2018/7/8 4:00	文件	2 KB
checkpoint.ckpt-2000.data-00000-of-...	2018/7/7 23:18	DATA-00000-OF...	7,804 KB
checkpoint.ckpt-2000.index	2018/7/7 23:18	INDEX 文件	5 KB
checkpoint.ckpt-2000.meta	2018/7/7 23:18	META 文件	392 KB
checkpoint.ckpt-4000.data-00000-of-...	2018/7/7 23:29	DATA-00000-OF...	7,804 KB
checkpoint.ckpt-4000.index	2018/7/7 23:29	INDEX 文件	5 KB
checkpoint.ckpt-4000.meta	2018/7/7 23:29	META 文件	392 KB
checkpoint.ckpt-6000.data-00000-of-...	2018/7/7 23:41	DATA-00000-OF...	7,804 KB
checkpoint.ckpt-6000.index	2018/7/7 23:41	INDEX 文件	5 KB
checkpoint.ckpt-6000.meta	2018/7/7 23:41	META 文件	392 KB
checkpoint.ckpt-8000.data-00000-of-...	2018/7/7 23:53	DATA-00000-OF...	7,804 KB
checkpoint.ckpt-8000.index	2018/7/7 23:53	INDEX 文件	5 KB
checkpoint.ckpt-8000.meta	2018/7/7 23:53	META 文件	392 KB
checkpoint.ckpt-10000.data-00000-of...	2018/7/8 0:04	DATA-00000-OF...	7,804 KB
checkpoint.ckpt-10000.index	2018/7/8 0:04	INDEX 文件	5 KB
checkpoint.ckpt-10000.meta	2018/7/8 0:04	META 文件	392 KB
checkpoint.ckpt-12000.data-00000-of...	2018/7/8 0:16	DATA-00000-OF...	7,804 KB
checkpoint.ckpt-12000.index	2018/7/8 0:16	INDEX 文件	5 KB
checkpoint.ckpt-12000.meta	2018/7/8 0:16	META 文件	392 KB
checkpoint.ckpt-14000.data-00000-of...	2018/7/8 0:28	DATA-00000-OF...	7,804 KB
checkpoint.ckpt-14000.index	2018/7/8 0:28	INDEX 文件	5 KB
checkpoint.ckpt-14000.meta	2018/7/8 0:28	META 文件	392 KB
checkpoint.ckpt-16000.data-00000-of...	2018/7/8 0:39	DATA-00000-OF...	7,804 KB
checkpoint.ckpt-16000.index	2018/7/8 0:39	INDEX 文件	5 KB
checkpoint.ckpt-16000.meta	2018/7/8 0:39	META 文件	392 KB
checkpoint.ckpt-18000.data-00000-of...	2018/7/8 0:51	DATA-00000-OF...	7,804 KB
checkpoint.ckpt-18000.index	2018/7/8 0:51	INDEX 文件	5 KB

训练可视化

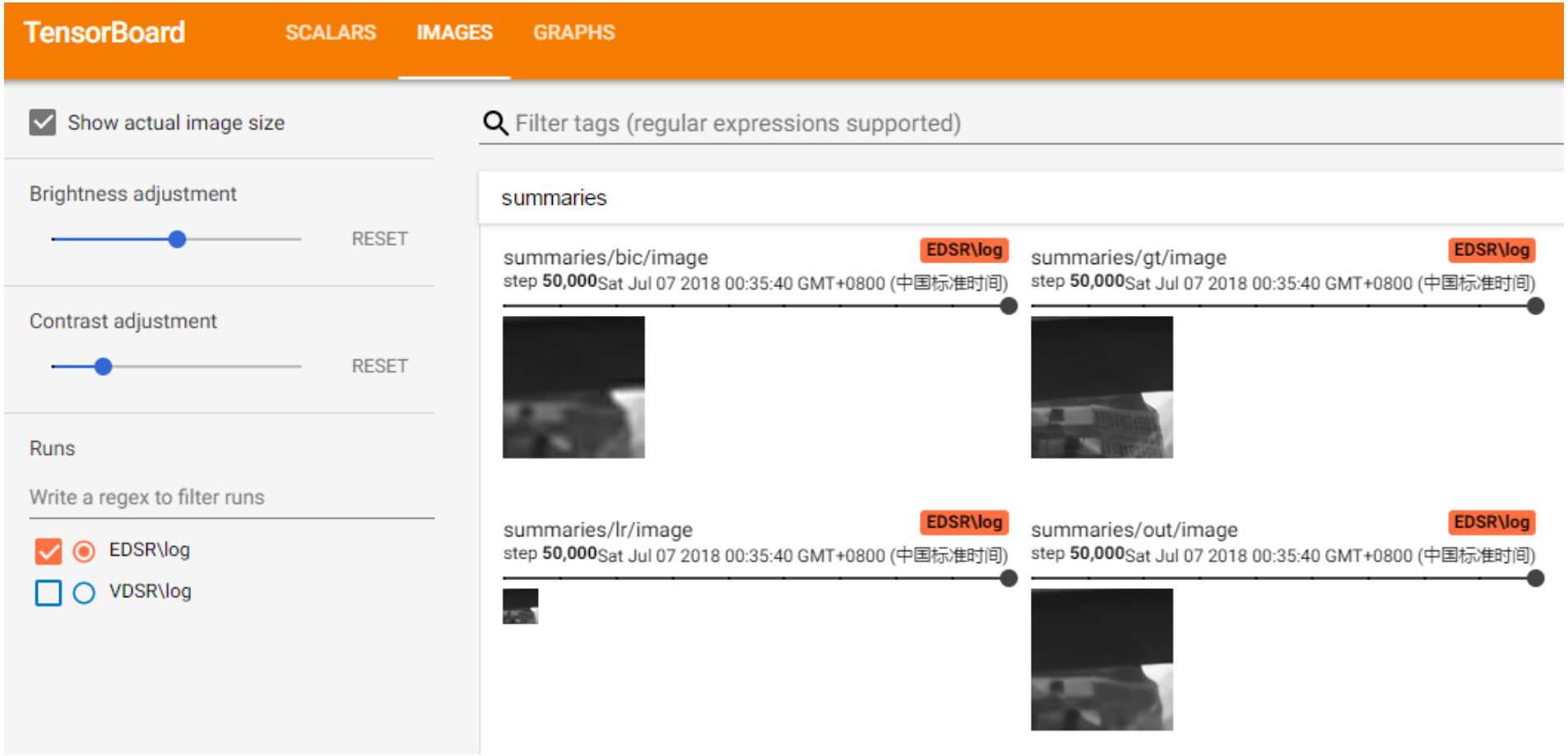
- 当前目录下执行 tensorboard --logdir=output
- 执行后会启动，查找该目录下所有的log
- 浏览器访问指定地址即可查看训练信息

```
2018-07-14 20:02:00.485602: I T:\src\github\tensorflow\tensorflow\core\p
2018-07-14 20:02:01.050991: I T:\src\github\tensorflow\tensorflow\core\c
name: GeForce GTX 1080 Ti major: 6 minor: 1 memoryClockRate(GHz): 1.582
pciBusID: 0000:01:00.0
totalMemory: 11.00GiB freeMemory: 9.08GiB
2018-07-14 20:02:01.059242: I T:\src\github\tensorflow\tensorflow\core\c
2018-07-14 20:02:03.008679: I T:\src\github\tensorflow\tensorflow\core\c
2018-07-14 20:02:03.013493: I T:\src\github\tensorflow\tensorflow\core\c
2018-07-14 20:02:03.016630: I T:\src\github\tensorflow\tensorflow\core\c
TensorBoard 1.8.0 at http://DESKTOP-86QFJ4A:6006 (Press CTRL+C to quit)
```

训练可视化


TensorBoard
SCALARS
IMAGES
GRAPHS
INACTIVE
Filter tags (regular expressions supported)
Show data download links
Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing
0.999
Horizontal Axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
EDSR\log
VDSR\log
summaries
summaries/learning_rate
1.020e-4
9.800e-5
9.400e-5
9.000e-5
8.600e-5
0.000
15.00k
30.00k
45.00k
summaries/loss
25.0
23.0
21.0
19.0
17.0
15.0
0.000
10.00k
20.00k
30.00k
40.00k
50.00k
summaries/mse_loss
25.0
23.0
21.0
19.0
17.0
15.0
0.000
10.00k
20.00k
30.00k
40.00k
50.00k
summaries/reg_loss
0.0600
0.0400
0.0200
0.00
0.000
10.00k
20.00k
30.00k
40.00k
50.00k

训练可视化


TensorBoard
SCALARS
IMAGES
GRAPHS
Show actual image size
Filter tags (regular expressions supported)
Brightness adjustment
RESET
Contrast adjustment
RESET
Runs
Write a regex to filter runs
EDSR\log
VDSR\log
summaries
summaries/bic/image
EDSR\log
step 50,000 Sat Jul 07 2018 00:35:40 GMT+0800 (中国标准时间)
summaries/gt/image
EDSR\log
step 50,000 Sat Jul 07 2018 00:35:40 GMT+0800 (中国标准时间)
summaries/lr/image
EDSR\log
step 50,000 Sat Jul 07 2018 00:35:40 GMT+0800 (中国标准时间)
summaries/out/image
EDSR\log
step 50,000 Sat Jul 07 2018 00:35:40 GMT+0800 (中国标准时间)

模型测试

```
TEST_DIR = 'test_data/Set5'
MODEL = 'VDSR'  # 'VDSR' or 'EDSR' or 'BICUBIC'
MODEL_CKPT_PATH = 'model/{}/checkpoint.ckpt-50000'.format(MODEL)
OUTPUT_DIR = 'result/calendar/{}'.format(MODEL)
DEVICE_MODE = 'GPU'  # 'CPU' or 'GPU'
DEVICE_GPU_ID = '0'
SCALE = 4

if DEVICE_MODE == 'CPU':
    os.environ['CUDA_VISIBLE_DEVICES'] = '-1'
else:
    os.environ['CUDA_VISIBLE_DEVICES'] = DEVICE_GPU_ID

if not os.path.exists(OUTPUT_DIR):
    os.makedirs(OUTPUT_DIR)

if MODEL == 'VDSR':
    model = VDSR()
elif MODEL == 'EDSR':
    model = EDSR()
else:
    model = BICUBIC()

lr = tf.placeholder(tf.float32, [None, None, None, 1])
res = model(lr)
```

```python
if not MODEL == 'BICUBIC':
    saver = tf.train.Saver()
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
sess = tf.Session(config=config)
sess.run(tf.global_variables_initializer())
if not MODEL == 'BICUBIC':
    saver.restore(sess, MODEL_CKPT_PATH)

fs = glob.glob(os.path.join(TEST_DIR, '*.png'))
psnrs = []
for f in fs:
    img = misc.imread(f)
    lr_img = misc.imresize(img, 1.0 / SCALE, 'bicubic')
    lr_y = utils.rgb2ycbcr(lr_img)[:, :, :1]
    lr_y = np.expand_dims(lr_y, 0).astype(np.float32) / 255.0

    res_y = sess.run(res, feed_dict={lr: lr_y})

    res_y = np.clip(res_y, 0, 1)[0] * 255.0
    bic_img = misc.imresize(lr_img, SCALE / 1.0, 'bicubic')

    bic_ycbcr = utils.rgb2ycbcr(bic_img)
    bic_ycbcr[:, :, :1] = res_y
    res_img = utils.img_to_uint8(utils.ycbcr2rgb(bic_ycbcr))
    img_name = f.split(os.sep)[-1]
    misc.imsave(os.path.join(OUTPUT_DIR, img_name), res_img)

    gt_y = utils.rgb2ycbcr(img)[:, :, :1]
    psnr = utils.psnr(res_y[SCALE:-SCALE, SCALE:-SCALE], gt_y[SCALE:-SCALE, SCALE:-SCALE])
    psnrs.append(psnr)
    print(img_name, 'PSNR:', psnr)

print('AVG PSNR:', np.mean(psnrs))
```

需要思考的一些问题

- 1. 如何确保训练的过程中是几乎完全利用了GPU？如果没有完全利用，有哪些步骤影响了其训练计算过程？
- 2. 如何将自己的模型与他人的模型进行混合利用，如通过预训练好的VGG16网络进行一部分特征计算？
- 3. 如何将不同的网络参数以不同的学习率进行训练，或者部分网络参数不参与训练？
- 4. 如何将模型的部分参数用预训练好的参数进行初始化？
- 5. 为了增大batch size，有时候使用多张卡进行并行训练，具体怎么做？