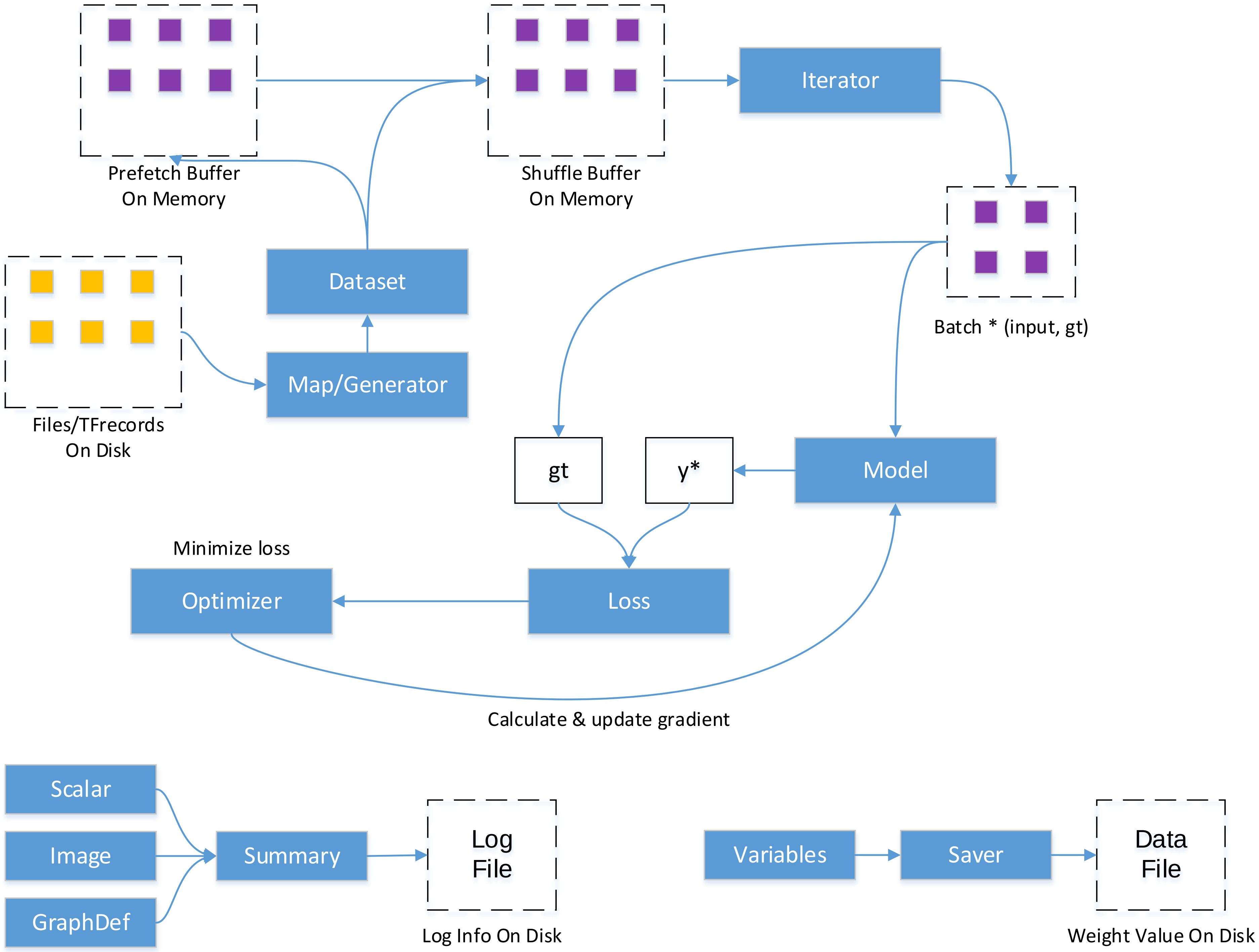
模型建立

Tensorflow

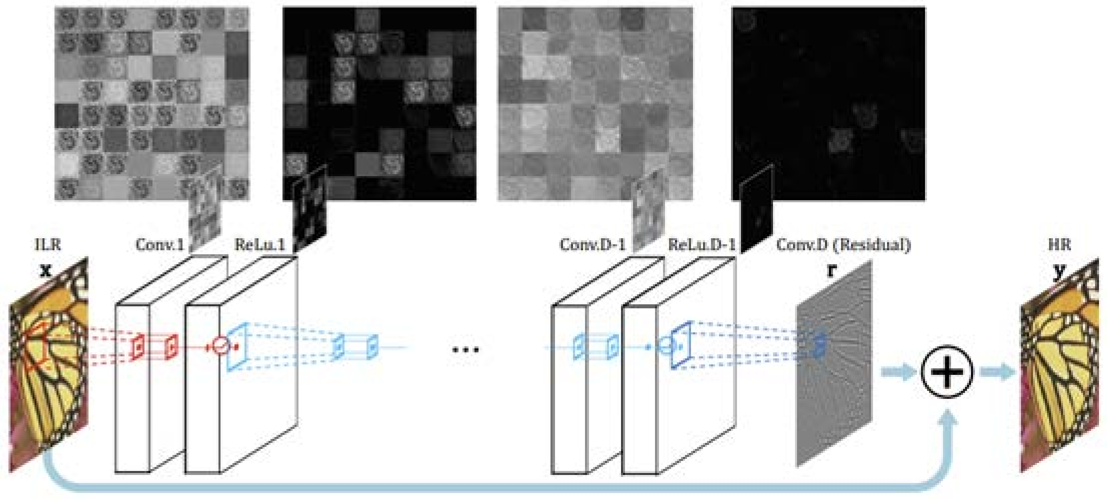
林楚铭

cmlin17@fudan.edu.cn



Prefetch Buffer
On Memory
Shuffle Buffer
On Memory
Iterator
Dataset
Batch * (input, gt)
Map/Generator
Files/TFrecords
On Disk
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

构造VDSR模型



1. 输入图为bicubic图像
2. 20层卷积层，都为3x3卷积，前19层的激活函数为ReLU

tf.layers.conv2d

```
tf.layers.conv2d(
    inputs,
    filters,
    kernel_size,
    strides=(1, 1),
    padding='valid',
    data_format='channels_last',
    dilation_rate=(1, 1),
    activation=None,
    use_bias=True,
    kernel_initializer=None,
    bias_initializer=tf.zeros_initializer(),
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    trainable=True,
    name=None,
    reuse=None
)
```

```
def conv2d(x, num_output, kernel_size=3, stride=1, act=tf.nn.relu, name=None):
    return tf.layers.conv2d(x, num_output, kernel_size, stride, 'same',
                            activation=act, name=name,
                            kernel_regularizer=tf.contrib.layers.l2_regularizer(scale=1e-5))
```

搭建VDSR

```python
class VDSR():
    def __init__(self, scale=4):
        self.scale = scale

    def __call__(self, lr, bic=None):
        with tf.variable_scope('VDSR', reuse=tf.AUTO_REUSE):
            b, h, w, c = tf.unstack(tf.shape(lr))
            scale = self.scale
            layer_num = 20
            if bic == None:
                bic = tf_resize_image(lr, scale)
                bic = tf.reshape(bic, [b, h * scale, w * scale, 1])
            self.bic = bic
            x = bic
            for i in range(layer_num - 1):
                x = conv2d(x, 64)
            x = conv2d(x, 1, act=tf.identity) + bic
            return x
```

tf_resize_image

```python
def tf_resize_image(imgs, scale):
    def resize_image(imgs, scale):
        b = imgs.shape[0]
        c = imgs.shape[-1]
        res = []
        for i in range(b):
            img = imgs[i]
            tar_img = []
            for j in range(c):
                tar_img.append(misc.imresize(img[:, :, j], scale / 1.0, 'bicubic', mode='F'))
            img = np.stack(tar_img, -1)
            res.append(img)

        return np.stack(res)
    return tf.py_func(lambda x: resize_image(x, scale), [imgs], tf.float32)
```

测试模型是否可运行

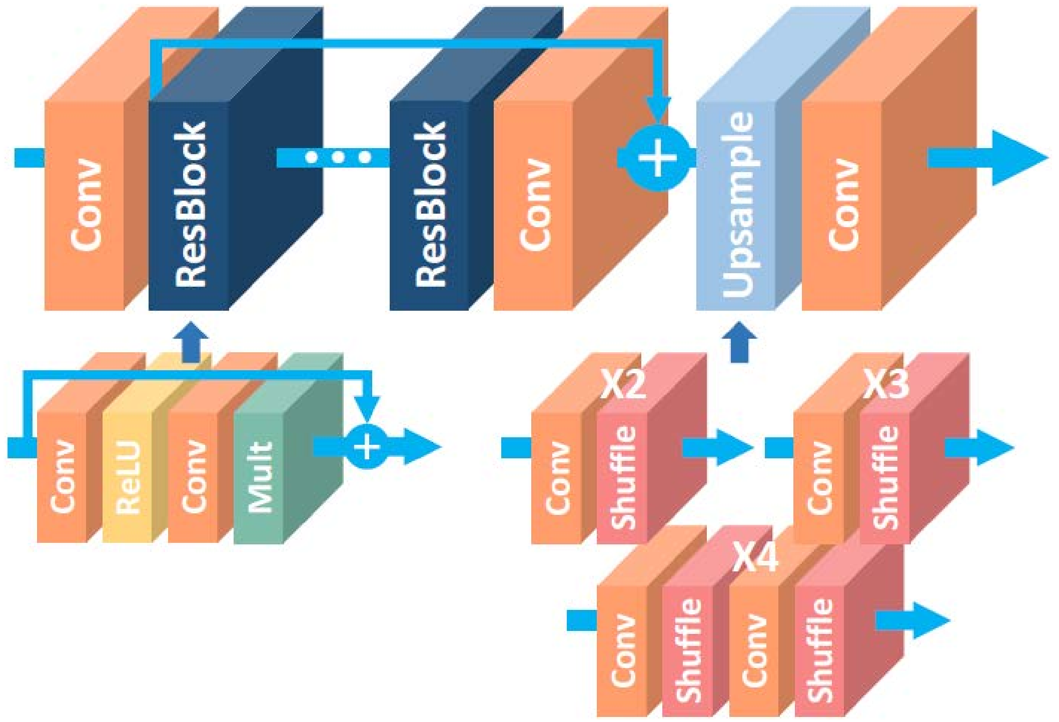
```
if __name__ == '__main__':
    import os

    os.environ['CUDA_VISIBLE_DEVICES'] = '-1'
    lr = tf.zeros([1, 24, 24, 1])
    model = VDSR()
    # model = EDSR()
    res = model(lr)
    sess = tf.Session()
    sess.run(tf.global_variables_initializer())
    out = sess.run(res)
    print(out.shape)

Out: (1, 96, 96, 1)
```

构造EDSR模型

- 构造一个类似EDSR模型
- 用一个反卷积层作为上采样层
- 反卷积层直接输出结果



tf.layers.conv2d_transpose

```
tf.layers.conv2d_transpose(
    inputs,
    filters,
    kernel_size,
    strides=(1, 1),
    padding='valid',
    data_format='channels_last',
    activation=None,
    use_bias=True,
    kernel_initializer=None,
    bias_initializer=tf.zeros_initializer(),
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    trainable=True,
    name=None,
    reuse=None
)
```

```
def deconv2d(x, num_output, kernel_size, stride, act=None, name=None):
    return tf.layers.conv2d_transpose(x, num_output, kernel_size, stride, 'same',
                                      activation=act, name=name,
                                      kernel_regularizer=tf.contrib.layers.l2_regularizer(scale=1e-5))
```

搭建EDSR

```python
class EDSR():
    def __init__(self, scale=4):
        self.scale = scale

    def res2d(self, x, num_output, kernel_size, name, scale=0.1):
        with tf.variable_scope(name):
            x0 = x
            x = conv2d(x, num_output, kernel_size)
            x = conv2d(x, num_output, kernel_size, act=tf.identity)
            x = x0 + x * scale
            return x

    def __call__(self, lr, bic=None):
        with tf.variable_scope('EDSR', reuse=tf.AUTO_REUSE):
            b, h, w, c = tf.unstack(tf.shape(lr))
            scale = self.scale
            if bic == None:
                bic = tf_resize_image(lr, scale)
                bic = tf.reshape(bic, [b, h * scale, w * scale, 1])
            self.bic = bic
            x = conv2d(lr, 256)
            x0 = x
            for i in range(16):
                x = self.res2d(x, 256, 3, 'res2d_%d' % i)
            x = conv2d(x, 256)
            x = x + x0
            x = deconv2d(x, 1, kernel_size=2 * scale, stride=scale)
            return x + bic
```