UCLOUD

TensorFlow 101

Application In UCloud

---

http://hacker.duanshishi.com
https://github.com/burness

段石石@UCloud

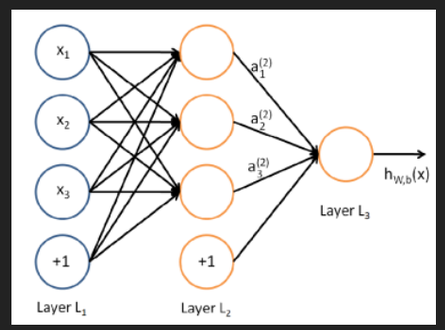
Bigger Data

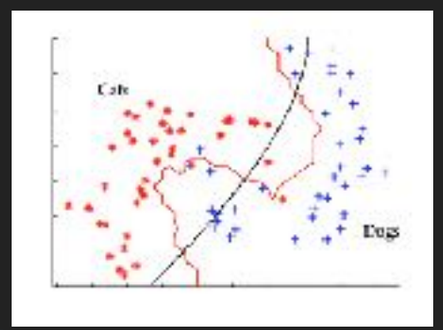
High Power

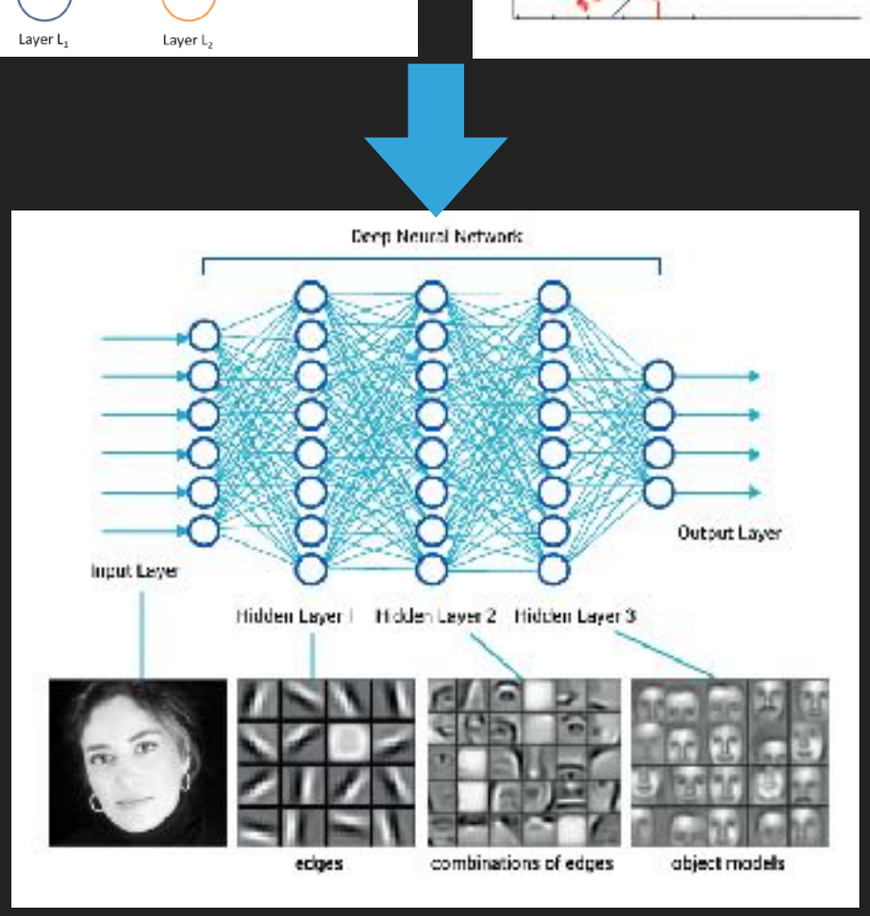
Better Model

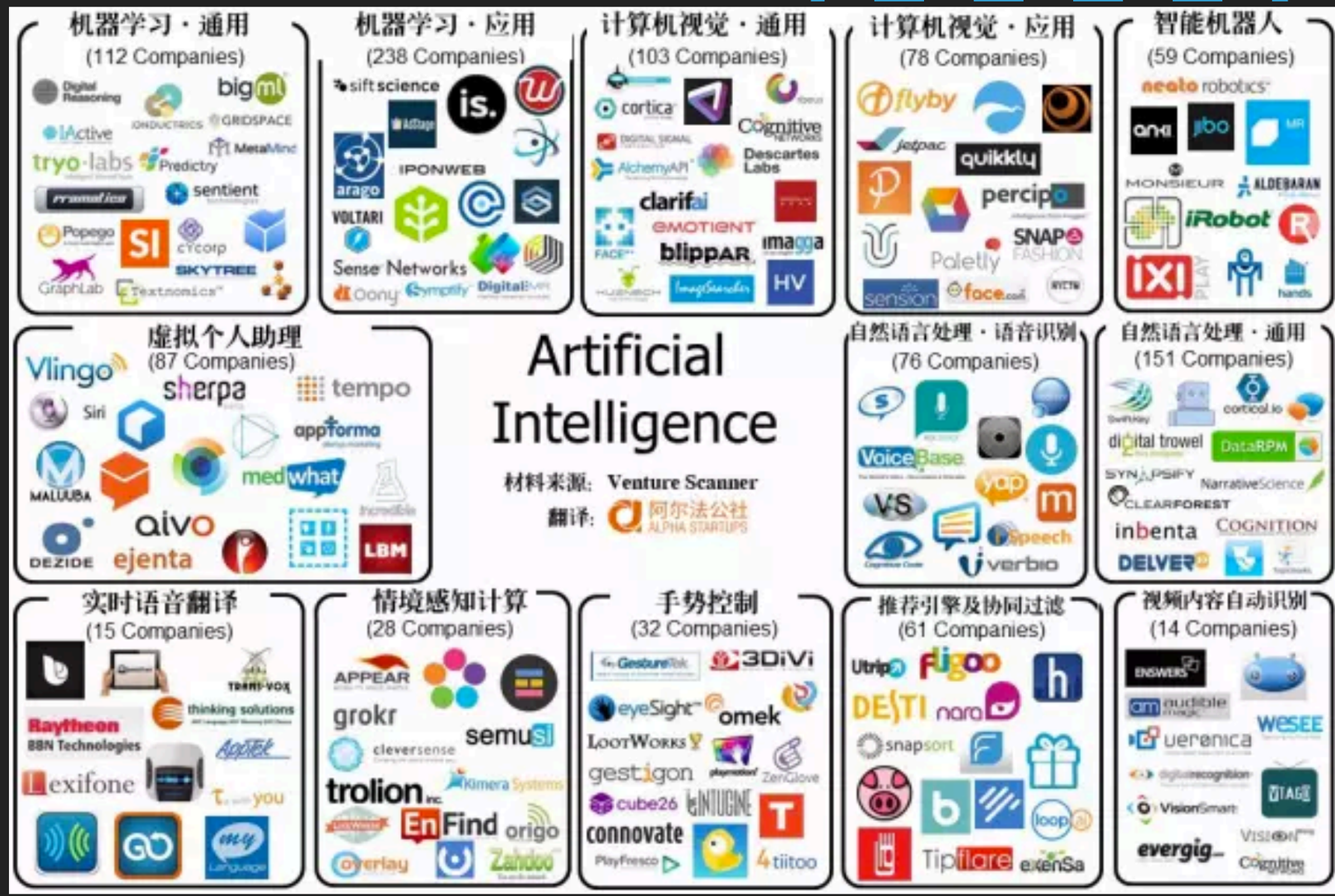
Better Performance

Shallow Learning → Deep Learning

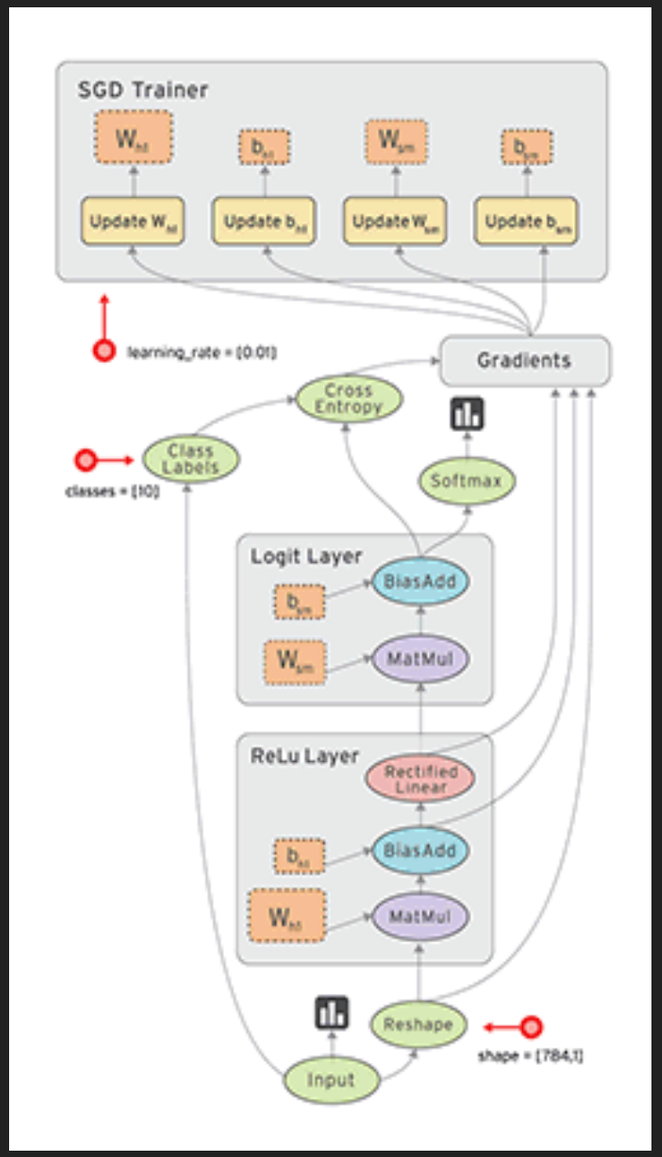


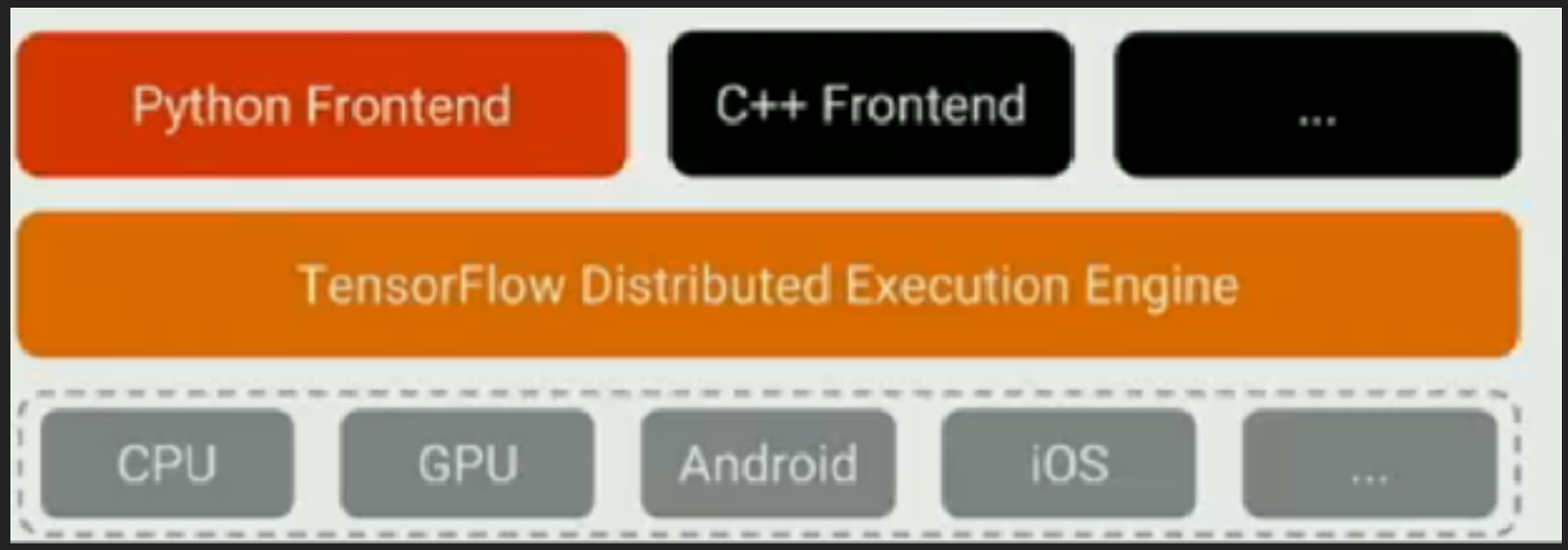






TensorFlow-Introduction





TensorFlow 1.0 Is Coming

- XLA (experimental): initial release of XLA, a domain-specific compiler for TensorFlow graphs, that targets CPUs and GPUs.
- TensorFlow Debugger (tfdbg): command-line interface and API.
- New python 3 docker images added.
- Made pip packages pypi compliant. TensorFlow can now be installed by pip install tensorflow command.
- Several python API calls have been changed to resemble NumPy more closely.
- New (experimental) Java API.
- Android: new person detection + tracking demo implementing "Scalable Object Detection using Deep Neural Networks" (w additional YOLO object detector support)
- Android: new camera-based image stylization demo based on "A Learned Representation For Artistic Style"

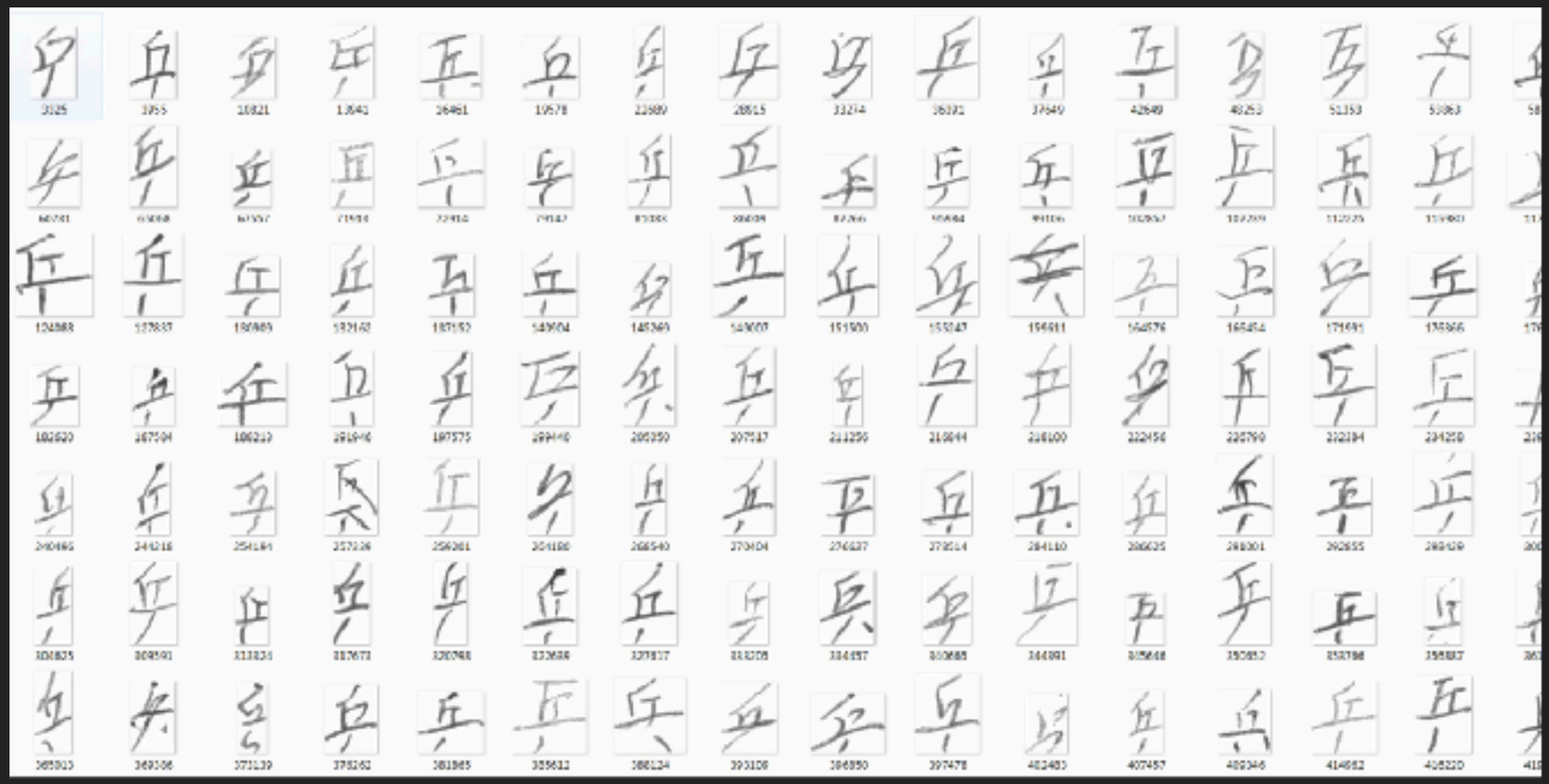
tf_upgrade.py --infile foo.py --outfile foo-upgraded.py

https://www.tensorflow.org/install/migration

Agenda

- A Toy But Complete Example In TensorFlow (70%)
  - Read Image Data In TensorFlow
  - Preprocess Your Image Data
  - Build Network
  - Train, Validate, Inference
  - Http Service With Flask
- TensorFlow In UCloud Ai Team (30%)
  - Distributed Train Platform
  - Inference Platform

A Toy Example-Not Mnist



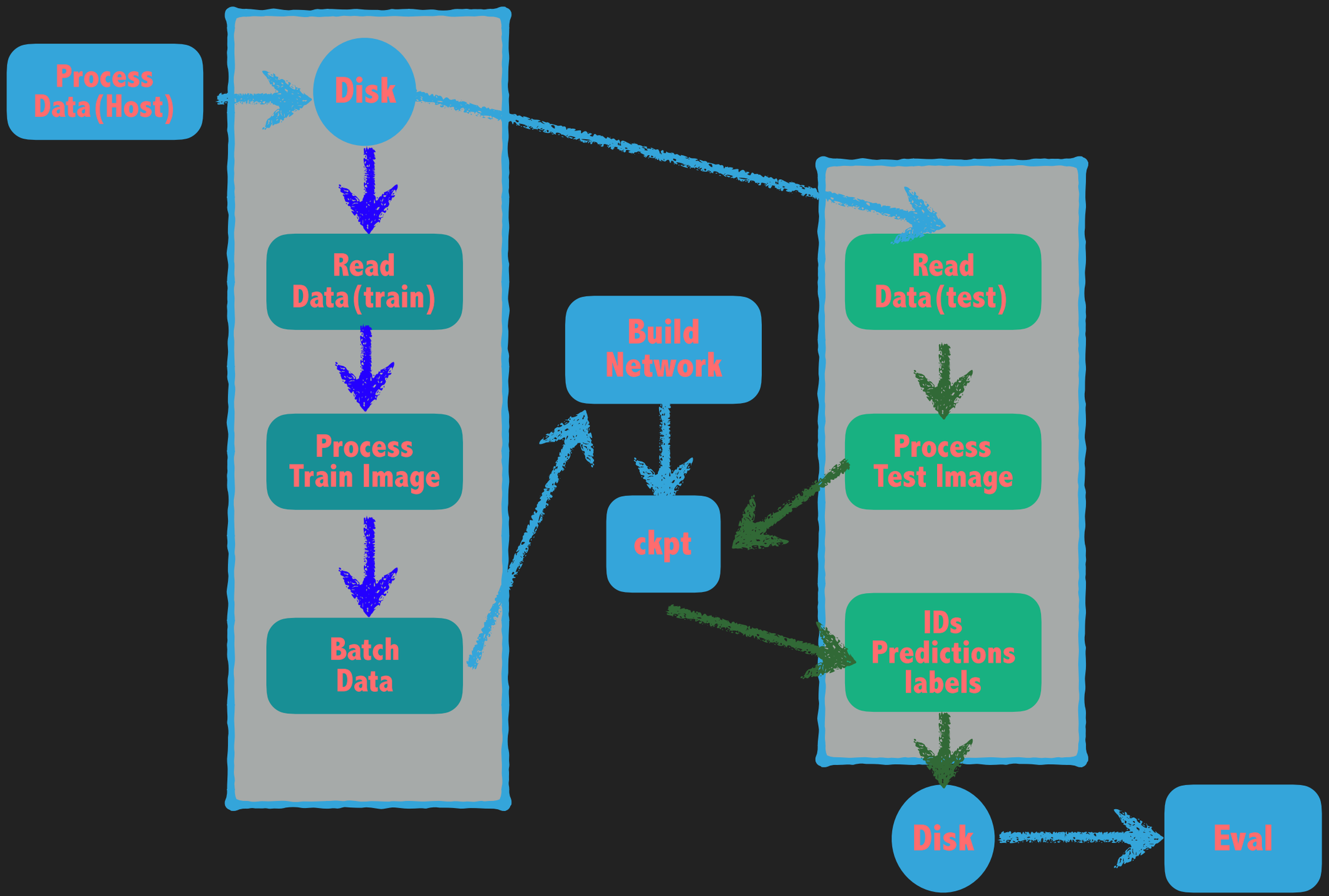
兵

data_url: http://www.nlpr.ia.ac.cn/CN/folder/folder8.shtml

wget http://www.nlpr.ia.ac.cn/databases/download/feature_data/HWDB1.1trn_gnt.zip

wget http://www.nlpr.ia.ac.cn/databases/download/feature_data/HWDB1.1tst_gnt.zip

WorkFlow


Process Data (Host)
Disk
Read Data (train)
Process Train Image
Batch Data
Build Network
ckpt
Read Data (test)
Process Test Image
IDs Predictions labels
Disk
Eval

gnt—>images/labels

```
def one_file(f):
    header_size = 10
    while True:
        header = np.fromfile(f, dtype='uint8', count=header_size)
        if not header.size: break
        sample_size = header[0] + (header[1]<<8) + (header[2]<<16) + (header[3]<<24)
        tagcode = header[5] + (header[4]<<8)
        width = header[6] + (header[7]<<8)
        height = header[8] + (header[9]<<8)
        if header_size + width*height != sample_size:
            break
        image = np.fromfile(f, dtype='uint8', count=width*height).reshape((height, width))
        yield image, tagcode
```

Read Image Data I—Reading Data In TensorFlow

- Feeding: build feed_dict when running each step
- Reading from files: build an input pipeline reads the data from files at the beginning of a TensorFlow graph
- Preloaded data: new a constant or variable in the TensorFlow graph holds all the data(Usually small data).

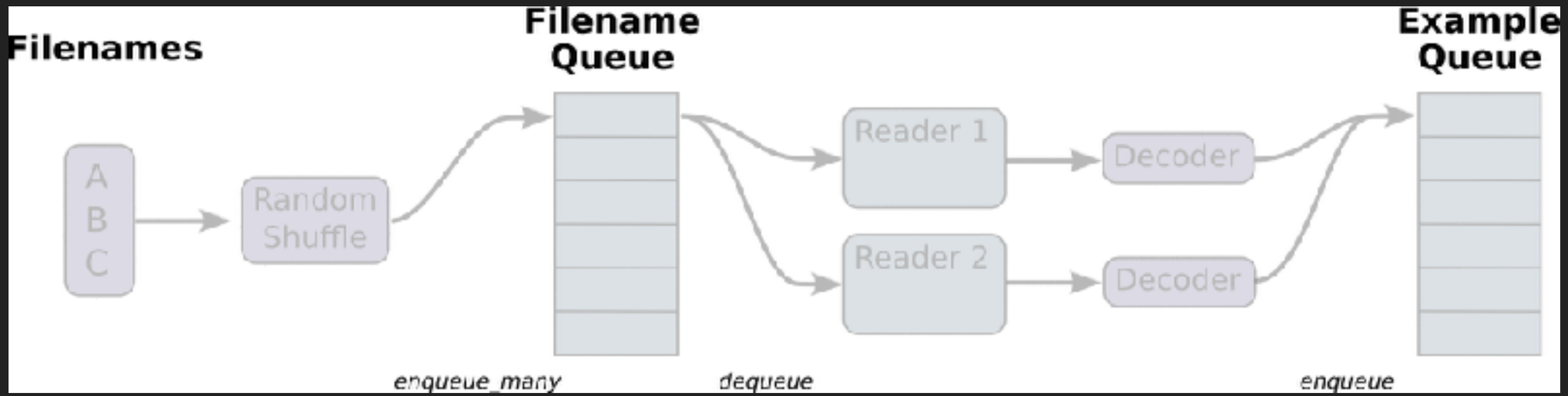
```
with tf.Session():
    input = tf.placeholder(tf.float32)
    classifier = ...
    print(classifier.eval(feed_dict={input: my_python_preprocessing_fn()}))
```

```
training_data = ...
training_labels = ...
with tf.Session():
    input_data = tf.constant(training_data)
    input_labels = tf.constant(training_labels)
    ...
```

```
training_data = ...
training_labels = ...
with tf.Session() as sess:
  data_initializer = tf.placeholder(dtype=training_data.dtype,
                                    shape=training_data.shape)
  label_initializer = tf.placeholder(dtype=training_labels.dtype,
                                     shape=training_labels.shape)
  input_data = tf.Variable(data_initializer, trainable=False, collections=[])
  input_labels = tf.Variable(label_initializer, trainable=False, collections=[])
  ...
  sess.run(input_data.initializer,
           feed_dict={data_initializer: training_data})
  sess.run(input_labels.initializer,
           feed_dict={label_initializer: training_labels})
```

- Get the list of filenames/labels
- New the filenames queue(shuffling/epoch)
- Apply reader with your file format
- Decoder for a record read by the reader
- Preprocess your data/label(data augmentation in TensorFlow)
- Shuffle and batch with the queue

Read Image Data II—Files Queue



If you have many small files, covert it to TFRecords first

https://github.com/burness/tensorflow-101/tree/master/covert_to_tfrecord

Read Image Data III—Image Decode

tf.image.decode_png
tf.image.decode_jpeg

[height, width, channels]

tf.image.decode_gif

[num_frames, height, width, 3]

tf.image.decode_jpeg(contents, channels=None, ratio=None, fancy_upscaling=None, try_recover_truncated=None, acceptable_fraction=None, dct_method=None, name=None)
tf.image.decode_png(contents, channels=None, dtype=None, name=None)
tf.image.decode_gif(contents, name=None)
tf.image.decode_image(contents, channels=None, name=None) (r1.0)

Read Image Data IV—Data Augmentation In TensorFlow

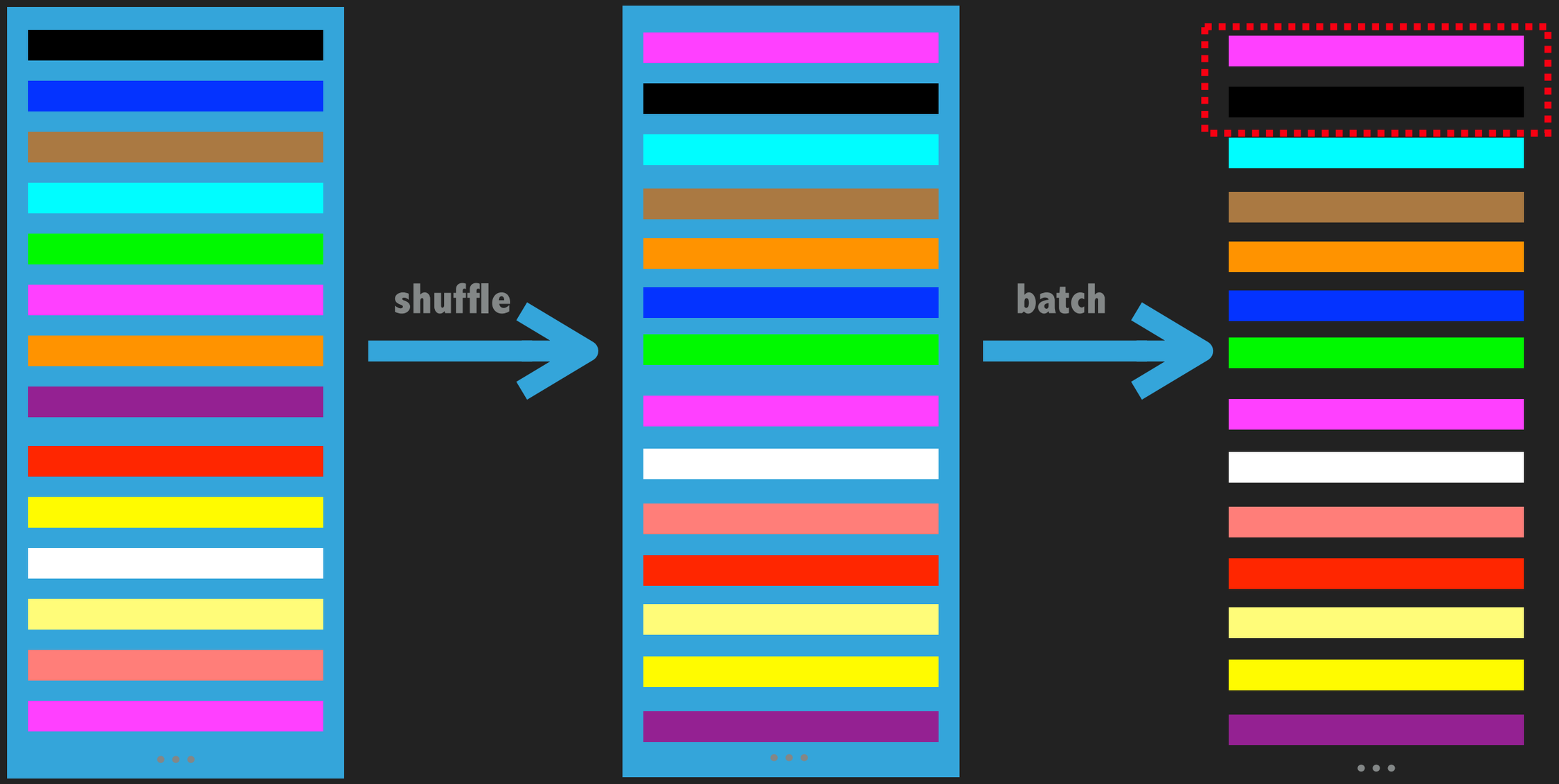
Flipping Rotating and Transposing

- `tf.image.flip_up_down(image)`
- `tf.image.random_flip_up_down(image, seed=None)`
- `tf.image.flip_left_right(image)`
- `tf.image.random_flip_left_right(image, seed=None)`
- `tf.image.transpose_image(image)`
- `tf.image.rot90(image, k=1, name=None)`

Image Adjustments

- `tf.image.adjust_brightness(image, delta)`
- `tf.image.random_brightness(image, max_delta, seed=None)`
- `tf.image.adjust_contrast(images, contrast_factor)`
- `tf.image.random_contrast(image, lower, upper, seed=None)`
- `tf.image.adjust_hue(image, delta, name=None)`
- `tf.image.random_hue(image, max_delta, seed=None)`
- `tf.image.adjust_gamma(image, gamma=1, gain=1)`
- `tf.image.adjust_saturation(image, saturation_factor, name=None)`
- `tf.image.random_saturation(image, lower, upper, seed=None)`
- `tf.image.per_image_standardization(image)`

Read Image Data V—Shuffle And Batch



Read Image Data VI—Put It Together

```python
class DataIterator:
    def __init__(self, data_dir):
        # Set FLAGS.charset_size to a small value if available computation power is limited.
        truncate_path = data_dir + ('%05d' % FLAGS.charset_size)
        print(truncate_path)
        self.image_names = []
        for root, sub_folder, file_list in os.walk(data_dir):
            if root < truncate_path:
                self.image_names += [os.path.join(root, file_path) for file_path in file_list]
        random.shuffle(self.image_names)
        self.labels = [int(file_name[len(data_dir):].split(os.sep)[0]) for file_name in self.image_names]

    @property
    def size(self):
        return len(self.labels)

    @staticmethod
    def data_augmentation(images):
        if FLAGS.random_flip_up_down:
            images = tf.image.random_flip_up_down(images)
        if FLAGS.random_brightness:
            images = tf.image.random_brightness(images, max_delta=0.3)
        if FLAGS.random_contrast:
            images = tf.image.random_contrast(images, 0.8, 1.2)
        return images

    def input_pipeline(self, batch_size, num_epochs=None, aug=False):
        images_tensor = tf.convert_to_tensor(self.image_names, dtype=tf.string)
        labels_tensor = tf.convert_to_tensor(self.labels, dtype=tf.int64)
        input_queue = tf.train.slice_input_producer([images_tensor, labels_tensor], num_epochs=num_epochs)

        labels = input_queue[1]
        images_content = tf.read_file(input_queue[0])
        images = tf.image.convert_image_dtype(tf.image.decode_png(images_content, channels=1), tf.float32)
        if aug:
            images = self.data_augmentation(images)
        new_size = tf.constant([FLAGS.image_size, FLAGS.image_size], dtype=tf.int32)
        images = tf.image.resize_images(images, new_size)
        image_batch, label_batch = tf.train.shuffle_batch([images, labels], batch_size=batch_size, capacity=50000,
                                                          min_after_dequeue=10000)
        return image_batch, label_batch
```

Build Network—Slim

```
input = ...
with tf.name_scope('conv1_1') as scope:
  kernel = tf.Variable(tf.truncated_normal([3, 3, 64, 128], dtype=tf.float32,
                                           stddev=1e-1), name='weights')
  conv = tf.nn.conv2d(input, kernel, [1, 1, 1, 1], padding='SAME')
  biases = tf.Variable(tf.constant(0.0, shape=[128], dtype=tf.float32),
                       trainable=True, name='biases')
  bias = tf.nn.bias_add(conv, biases)
  conv1 = tf.nn.relu(bias, name=scope)
```

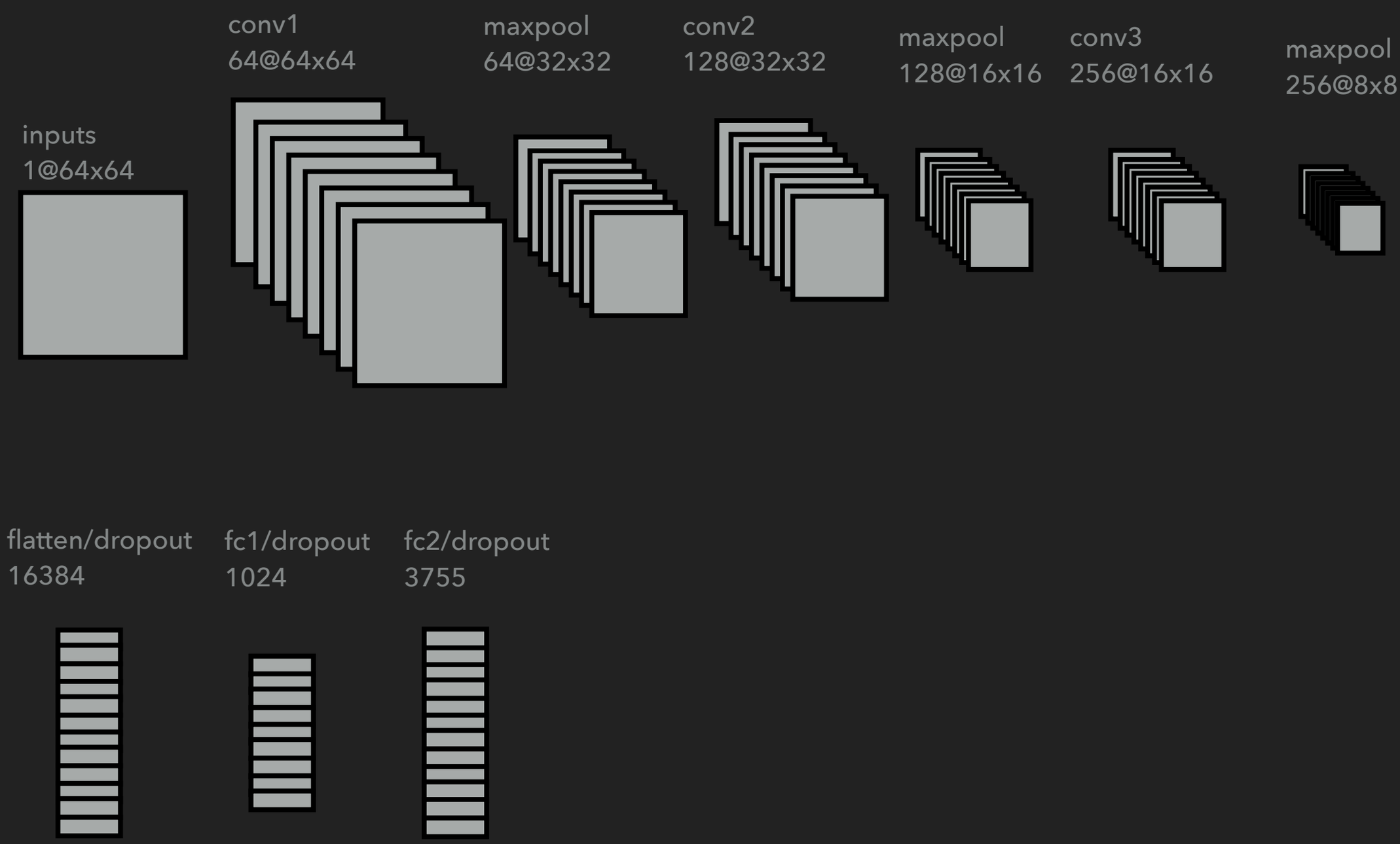
```
input = ...
net = slim.conv2d(input, 128, [3, 3], scope='conv1_1')
```

https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/slim

https://github.com/tflearn/tflearn

https://github.com/fchollet/keras

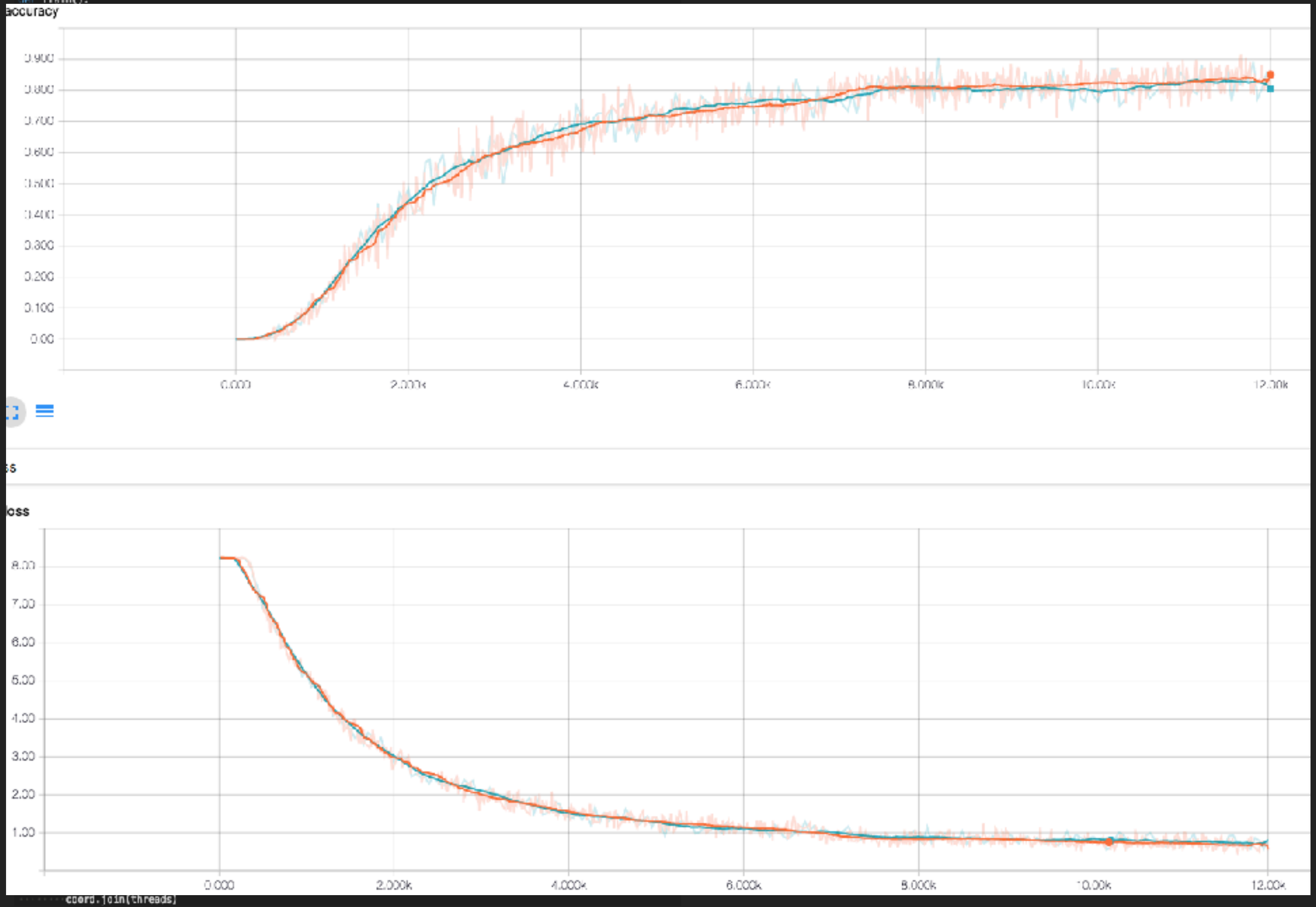
Build Network—Build Graph



Build Network—Build Graph

```python
def build_graph(top_k):
    # with tf.device('/cpu:0'):
    keep_prob = tf.placeholder(dtype=tf.float32, shape=[], name='keep_prob')
    images = tf.placeholder(dtype=tf.float32, shape=[None, 64, 64, 1], name='image_batch')
    labels = tf.placeholder(dtype=tf.int64, shape=[None], name='label_batch')

    conv_1 = slim.conv2d(images, 64, [3, 3], 1, padding='SAME', scope='conv1')
    max_pool_1 = slim.max_pool2d(conv_1, [2, 2], [2, 2], padding='SAME')
    conv_2 = slim.conv2d(max_pool_1, 128, [3, 3], padding='SAME', scope='conv2')
    max_pool_2 = slim.max_pool2d(conv_2, [2, 2], [2, 2], padding='SAME')
    conv_3 = slim.conv2d(max_pool_2, 256, [3, 3], padding='SAME', scope='conv3')
    max_pool_3 = slim.max_pool2d(conv_3, [2, 2], [2, 2], padding='SAME')

    flatten = slim.flatten(max_pool_3)
    fc1 = slim.fully_connected(slim.dropout(flatten, keep_prob), 1024, activation_fn=tf.nn.tanh, scope='fc1')
    logits = slim.fully_connected(slim.dropout(fc1, keep_prob), FLAGS.charset_size, activation_fn=None, scope='fc2')
        # logits = slim.fully_connected(flatten, FLAGS.charset_size, activation_fn=None, reuse=reuse, scope='fc')
    loss = tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logits(logits, labels))
    accuracy = tf.reduce_mean(tf.cast(tf.equal(tf.argmax(logits, 1), labels), tf.float32))

    global_step = tf.get_variable("step", [], initializer=tf.constant_initializer(0.0), trainable=False)
    rate = tf.train.exponential_decay(2e-4, global_step, decay_steps=2000, decay_rate=0.97, staircase=True)
    train_op = tf.train.AdamOptimizer(learning_rate=rate).minimize(loss, global_step=global_step)
    probabilities = tf.nn.softmax(logits)

    tf.summary.scalar('loss', loss)
    tf.summary.scalar('accuracy', accuracy)
    merged_summary_op = tf.summary.merge_all()
    predicted_val_top_k, predicted_index_top_k = tf.nn.top_k(probabilities, k=top_k)
    accuracy_in_top_k = tf.reduce_mean(tf.cast(tf.nn.in_top_k(probabilities, labels, top_k), tf.float32))

    return {'images': images,
            'labels': labels,
            'keep_prob': keep_prob,
            'top_k': top_k,
            'global_step': global_step,
            'train_op': train_op,
            'loss': loss,
            'accuracy': accuracy,
            'accuracy_top_k': accuracy_in_top_k,
            'merged_summary_op': merged_summary_op,
            'predicted_distribution': probabilities,
            'predicted_index_top_k': predicted_index_top_k,
            'predicted_val_top_k': predicted_val_top_k}
```

Train, Validate, Inference



Train, Validate, Inference

```python
def validation():
    print('validation')
    test_feeder = DataIterator(data_dir='../data/test/')

    final_predict_val = []
    final_predict_index = []
    groundtruth = []

    with tf.Session() as sess:
        test_images, test_labels = test_feeder.input_pipeline(batch_size=FLAGS.batch_size, num_epochs=1)
        graph = build_graph(3)

        sess.run(tf.global_variables_initializer())
        sess.run(tf.local_variables_initializer())  # initialize test_feeder's inside state

        coord = tf.train.Coordinator()
        threads = tf.train.start_queue_runners(sess=sess, coord=coord)

        saver = tf.train.Saver()
        ckpt = tf.train.latest_checkpoint(FLAGS.checkpoint_dir)
        if ckpt:
            saver.restore(sess, ckpt)
            print("restore from the checkpoint {0}".format(ckpt))

        logger.info(':::Start validation:::')
        try:
            i = 0
            acc_top_1, acc_top_k = 0.0, 0.0
            while not coord.should_stop():
                i += 1
                start_time = time.time()
                test_images_batch, test_labels_batch = sess.run([test_images, test_labels])
                feed_dict = {graph['images']: test_images_batch,
                             graph['labels']: test_labels_batch,
                             graph['keep_prob']: 1.0}
                batch_labels, probs, indices, acc_1, acc_k = sess.run([graph['labels'],
                                                                       graph['predicted_val_top_k'],
                                                                       graph['predicted_index_top_k'],
                                                                       graph['accuracy'],
                                                                       graph['accuracy_top_k']], feed_dict=feed_dict)
                final_predict_val += probs.tolist()
                final_predict_index += indices.tolist()
                groundtruth += batch_labels.tolist()
                acc_top_1 += acc_1
                acc_top_k += acc_k
                end_time = time.time()
                logger.info("the batch {0} takes {1} seconds, accuracy = {2}(top_1) {3}(top_k)"
                            .format(i, end_time - start_time, acc_1, acc_k))

        except tf.errors.OutOfRangeError:
            logger.info('----------------Validation Finished----------------')
            acc_top_1 = acc_top_1 * FLAGS.batch_size / test_feeder.size
            acc_top_k = acc_top_k * FLAGS.batch_size / test_feeder.size
            logger.info('top 1 accuracy {0} top k accuracy {1}'.format(acc_top_1, acc_top_k))
        finally:
            coord.request_stop()
        coord.join(threads)
    return {'prob': final_predict_val, 'indices': final_predict_index, 'groundtruth': groundtruth}
```

```
the batch 1740 takes 0.050522804260 seconds, accuracy = 0.8671875(top_1) 0.953125(top_k)
the batch 1741 takes 0.0505909919739 seconds, accuracy = 0.84375(top_1) 0.9296875(top_k)
the batch 1742 takes 0.0502688884735 seconds, accuracy = 0.8359375(top_1) 0.9453125(top_k)
the batch 1743 takes 0.0501899719238 seconds, accuracy = 0.8046875(top_1) 0.8984375(top_k)
the batch 1744 takes 0.0520930290222 seconds, accuracy = 0.828125(top_1) 0.953125(top_k)
the batch 1745 takes 0.0517480373779 seconds, accuracy = 0.8203125(top_1) 0.921875(top_k)
the batch 1746 takes 0.0515489578247 seconds, accuracy = 0.8203125(top_1) 0.9375(top_k)
the batch 1747 takes 0.0507018566132 seconds, accuracy = 0.8671875(top_1) 0.9453125(top_k)
the batch 1748 takes 0.0513739585876 seconds, accuracy = 0.8671875(top_1) 0.9375(top_k)
the batch 1749 takes 0.0513660907745 seconds, accuracy = 0.8515625(top_1) 0.9453125(top_k)
----------------Validation Finished----------------
top 1 accuracy 0.829636012161 top k accuracy 0.928849819859
Write result into result.dict
Write file ends
```

Train, Validate, Inference

```python
def inference(image):
    print('inference')
    temp_image = Image.open(image).convert('L')
    temp_image = temp_image.resize((FLAGS.image_size, FLAGS.image_size), Image.ANTIALIAS)
    temp_image = np.asarray(temp_image) / 255.0
    temp_image = temp_image.reshape([-1, 64, 64, 1])
    with tf.Session() as sess:
        logger.info('========start inference============')
        # images = tf.placeholder(dtype=tf.float32, shape=[None, 64, 64, 1])
        # Pass a shadow label 0. This label will not affect the computation graph.
        graph = build_graph(top_k=3)
        saver = tf.train.Saver()
        ckpt = tf.train.latest_checkpoint(FLAGS.checkpoint_dir)
        if ckpt:
            saver.restore(sess, ckpt)
        predict_val, predict_index = sess.run([graph['predicted_val_top_k'], graph['predicted_index_top_k']],
                                              feed_dict={graph['images']: temp_image, graph['keep_prob']: 1.0})
    return predict_val, predict_index
```

```
I tensorflow/core/common_runtime/gpu/gpu_device.cc:906] DMA: 0
I tensorflow/core/common_runtime/gpu/gpu_device.cc:916] 0:   Y
I tensorflow/core/common_runtime/gpu/gpu_device.cc:975] Creating TensorFlow device (/gpu:0) -> (device: 0, name: 
========start inference==========
the result info label 190 predict index [[ 190  538 2658]] predict_val [[ 0.77886552  0.14424528  0.01731586]]
```

Http Service With Flask

```python
class myTfModel(object):
    def __init__(self, checkpoint_file):
        self.checkpoint_file = checkpoint_file
        self.output = {}
        self.load_model()

    def load_model(self):
        sess = tf.Session()
        input_tensor = tf.placeholder(tf.float32, [None, 299, 299, 3])
        arg_scope = inception_v3_arg_scope()
        with slim.arg_scope(arg_scope):
            logits, end_points = inception_v3(
                input_tensor, is_training=False, num_classes=1001)
            saver = tf.train.Saver()
        # params_file = tf.train.latest_checkpoint(self.model_dir)
        saver.restore(sess, self.checkpoint_file)
        self.output['sess'] = sess
        self.output['input_tensor'] = input_tensor
        self.output['logits'] = logits
        self.output['end_points'] = end_points
        # return sess, input_tensor, logits, end_points

    def execute(self, data, **kwargs):
        sess = self.output['sess']
        input_tensor = self.output['input_tensor']
        logits = self.output['logits']
        end_points = self.output['end_points']
        # ims = []
        # for i in range(kwargs['batch_size']):
        im = Image.open(data).resize((299, 299))
        im = np.array(im) / 255.0
        im = im.reshape(-1, 299, 299, 3)
        start = time.time()
        predict_values, logit_values = sess.run(
            [end_points['Predictions'], logits], feed_dict={input_tensor: im})
        return predict_values
```

Inference result: patas, hussar monkey, Erythrocebus patas macaque baboon probascis monkey, Nasalis larvatus chimpanzee, chimp, Pan troglodytes

Here is Inception-V3 not the above model

https://github.com/burness/tensorflow-101/blob/master/finetuning/flask/flask_inference_1000.py

Maybe A Bug

Just focus on the block after `if step % FLAGS.eval_steps == 1:` It is all right (the step shouldn't add one) to run on CPU(add line `with tf.device("/cpu:0")` after `with tf.Session() as sess:`), But as show the below image, it will add one when we use GPU:

On CPU:

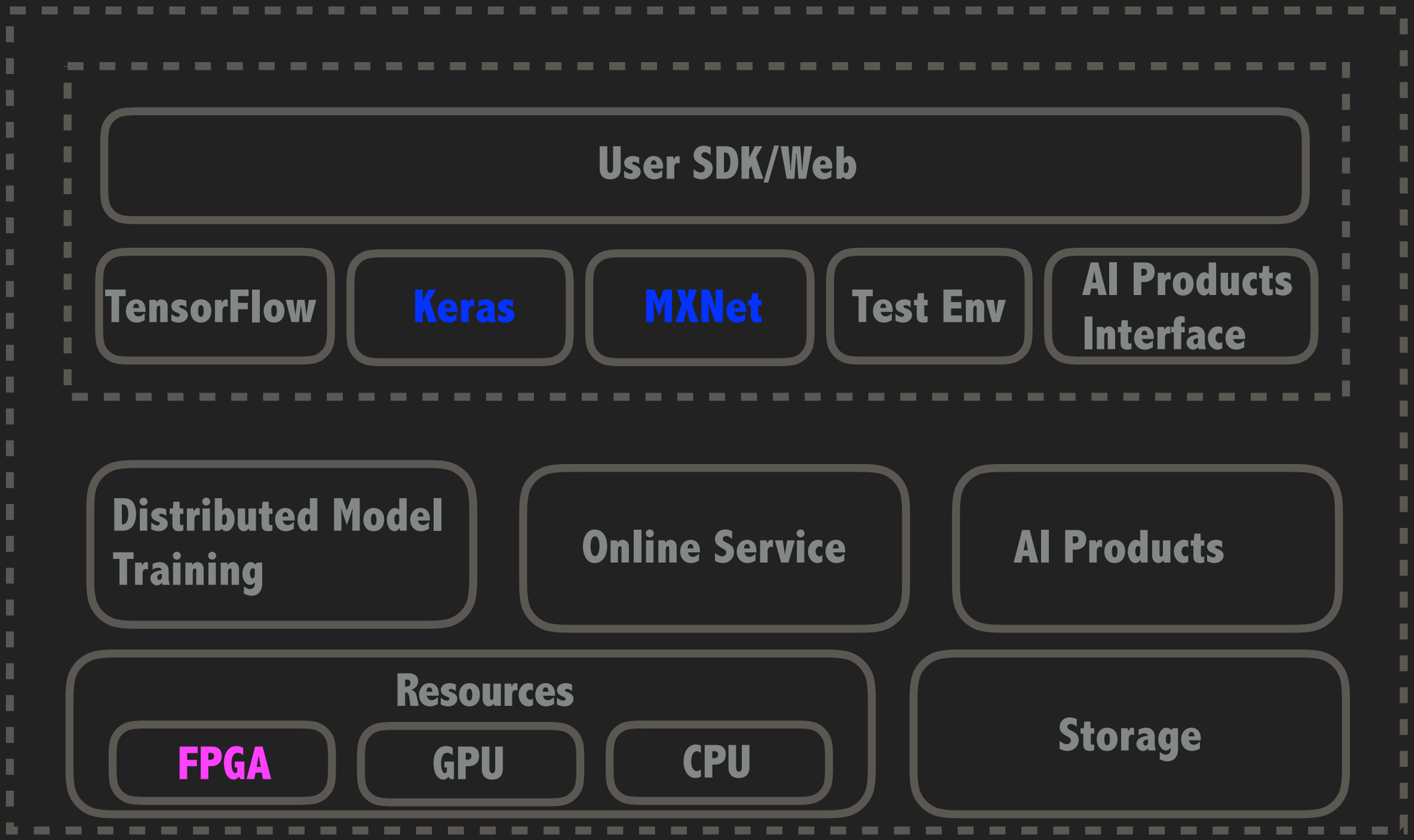
```
the step 50.0 takes 2.3421339988708496 loss 4.622500896453857
the step 51.0 takes 2.292131185531616 loss 4.648638112640381
==============Eval a batch=======================
the step 51.0 test accuracy: 0.03125
==============Eval a batch=======================
eval step again: step = 51.0
eval step again: step = 51.0
eval step again: step = 51.0
eval step again: step = 51.0
eval step again: step = 51.0
eval step again: step = 51.0
eval step again: step = 51.0
eval step again: step = 51.0
eval step again: step = 51.0
eval step again: step = 51.0
Save the ckpt of 51.0
the step 52.0 takes 2.302131414413452 loss 4.608097076416016
the step 53.0 takes 2.4231386184692383 loss 4.5948896408081055
```

On GPU:

```
the step 46.0 takes 0.308898925781 loss 4.68617630005
the step 47.0 takes 0.307048797607 loss 4.66950702667
the step 48.0 takes 0.293892860413 loss 4.64998865128
the step 49.0 takes 0.306770086288 loss 4.59738922119
the step 50.0 takes 0.306005954742 loss 4.60396003723
the step 51.0 takes 0.298496007919 loss 4.58915996552
==============Eval a batch=======================
the step 52.0 test accuracy: 0.015625
==============Eval a batch=======================
eval step again: step = 52.0
eval step again: step = 52.0
eval step again: step = 52.0
eval step again: step = 52.0
eval step again: step = 52.0
eval step again: step = 52.0
eval step again: step = 52.0
eval step again: step = 52.0
```

Tensorflow Applications In UCloud

Overview


User SDK/Web
TensorFlow
Keras
MXNet
Test Env
AI Products Interface
Distributed Model Training
Online Service
AI Products
Resources
FPGA
GPU
CPU
Storage

Distributed Deep Learning Train Platform

- TensorFlow On Kubernetes
- TensorFlow On Spark
- TensorFlow In UCloud Eco.

UCLOUD

TensorFlow on Kubernetes

- 集群管理
  - 通过Kubernetes的DNS机制设置服务器地址
  - 通过replica controller控制失败重启
  - Kubernetes提供监控、调度等功能

- 生命周期管理
  - 目前不会自动结束
  - 难以区分正常结束还是异常退出
  - 手动管理

- 存储解决方案
  - 使用nfs、ceph等分布式存储
  - HDFS

from 郑泽宇 才云科技 caicloud.io

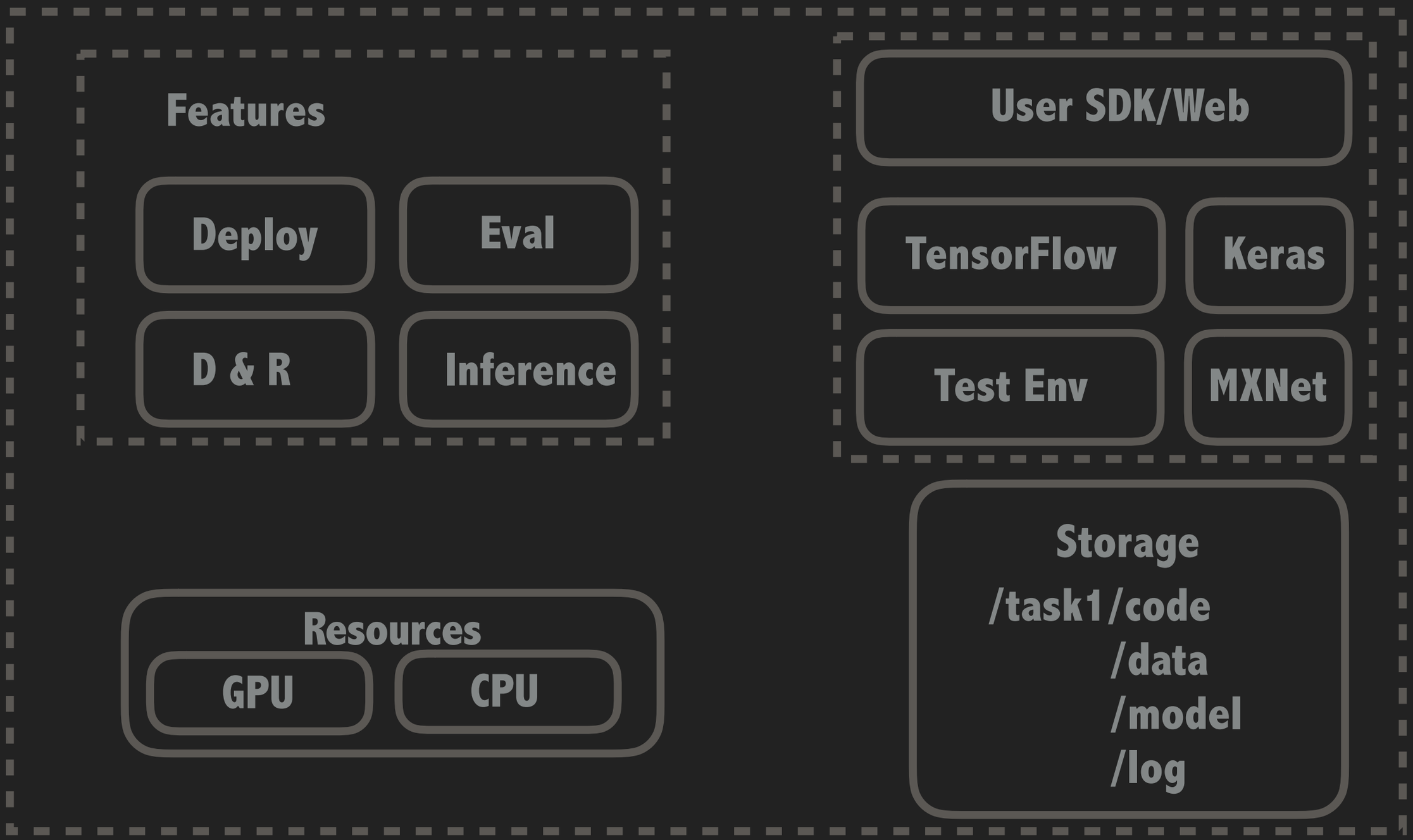
TensorFlow On Spark

TF-Slim Inception

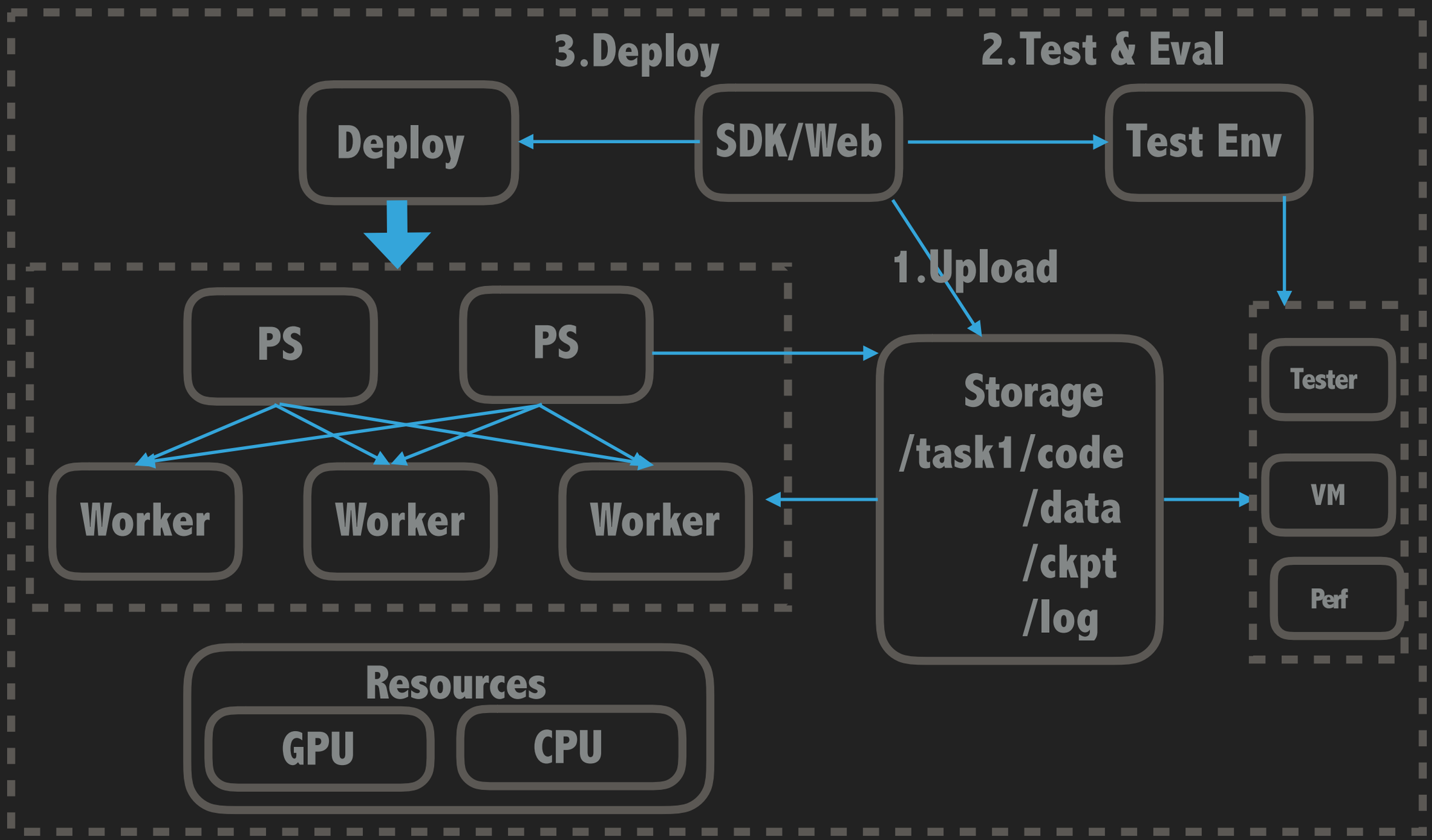
```
# set environment variables (if not already done)
export PYTHON_ROOT=~/Python
export PYSPARK_PYTHON=${PYTHON_ROOT}/bin/python
export SPARK_YARN_USER_ENV="PYSPARK_PYTHON=Python/bin/python"
export PATH=${PYTHON_ROOT}/bin/:$PATH
export QUEUE=gpu
# export DATASET_DIR=<HDFS path to your downloaded files>

# for CPU mode:
# export QUEUE=default
# --conf spark.executorEnv.LD_LIBRARY_PATH="$JAVA_HOME/jre/lib/amd64/server" \
# remove --driver-library-path

# hadoop fs -rm -r slim_train
export NUM_GPU=1
export MEMORY=$((NUM_GPU * 27))
${SPARK_HOME}/bin/spark-submit --master yarn --deploy-mode cluster \
--queue ${QUEUE} \
--num-executors 3 \
--executor-memory ${MEMORY}G \
--py-files tensorflow/tfspark.zip,slim.zip \
--conf spark.dynamicAllocation.enabled=false \
--conf spark.yarn.maxAppAttempts=1 \
--conf spark.ui.view.acls=* \
--archives hdfs:///user/${USER}/Python.zip#Python \
--conf spark.executorEnv.LD_LIBRARY_PATH="/usr/local/cuda-7.5/lib64:$JAVA_HOME/jre/lib/amd64/server" \
--driver-library-path="/usr/local/cuda-7.5/lib64" \
tensorflow/examples/slim/train_image_classifier.py \
--dataset_dir $DATASET_DIR \
--train_dir hdfs://default/user/${USER}/slim_train \
--dataset_name imagenet \
--dataset_split_name train \
--model_name inception_v3 \
--max_number_of_steps 1000 \
--num_gpus ${NUM_GPU} \
--batch_size 32 \
--num_ps_tasks 1
```

https://github.com/yahoo/TensorFlowOnSpark

Distributed Train In UCloud I



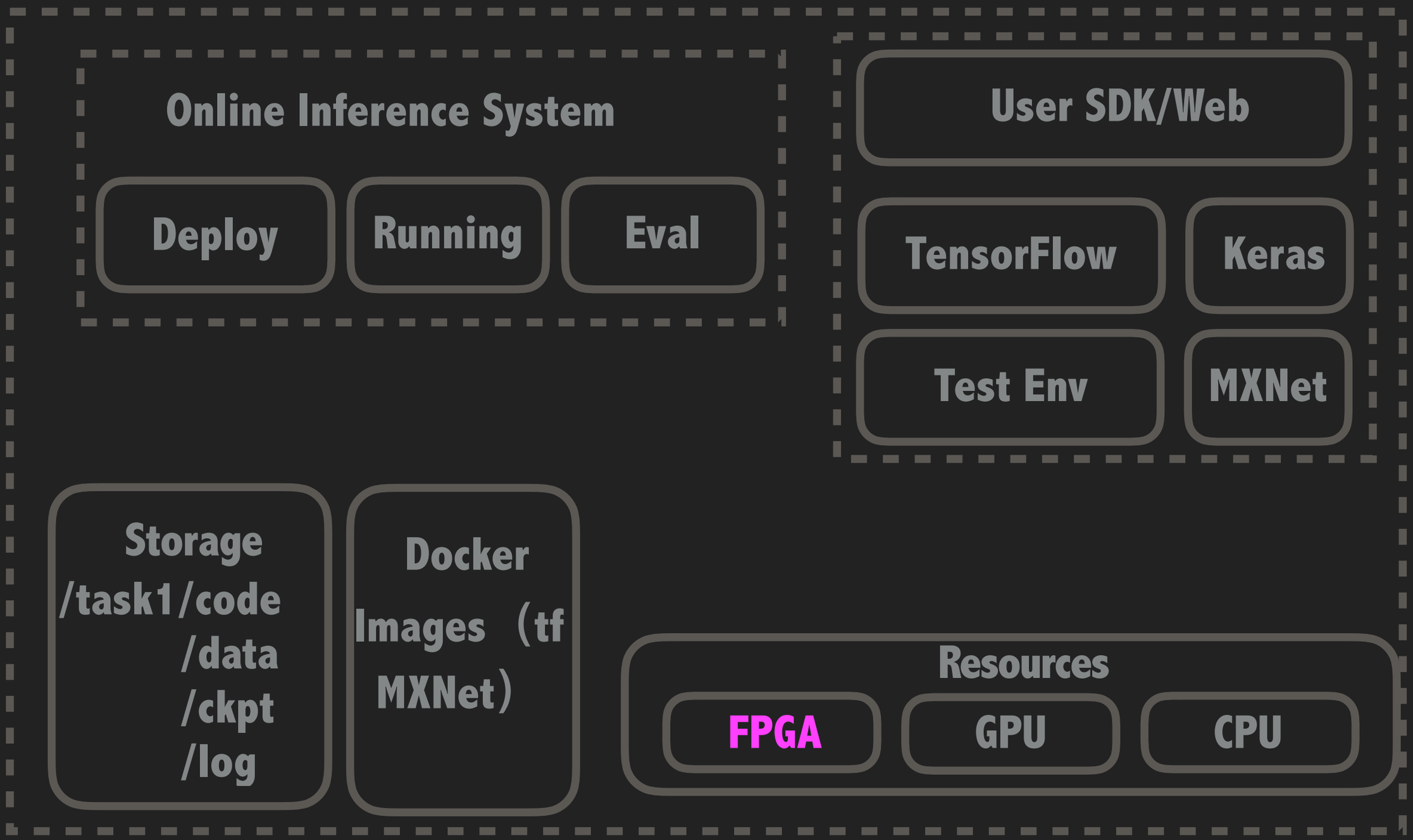
Distributed Train In UCloud II



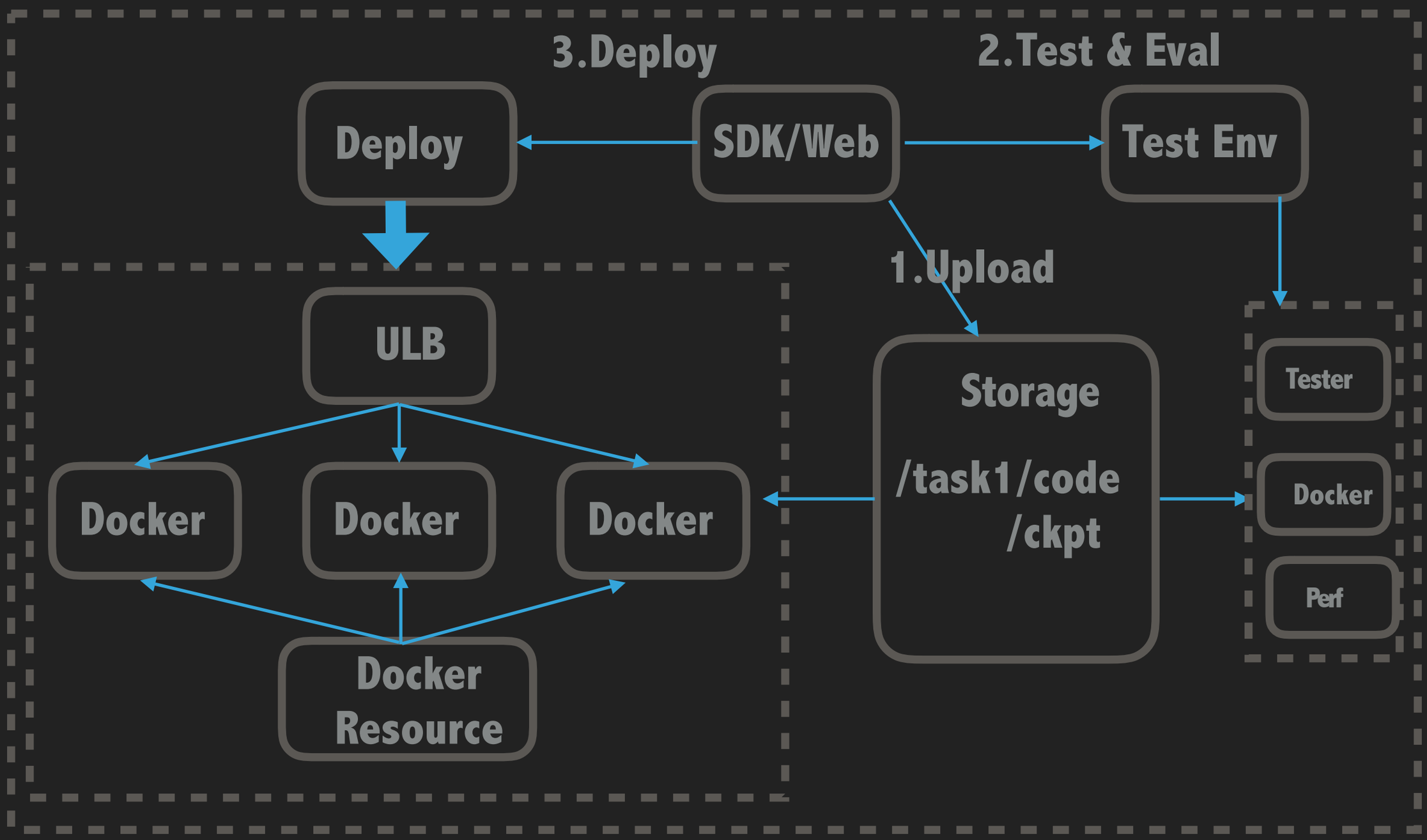
Distributed Train In UCloud III

- Data Prefetch: Computing/Data Loading parallel
- PS Network enhance: PS net pressure high, PS VM net enhance
- Auto fail restart: detect fail + checkpoint restore

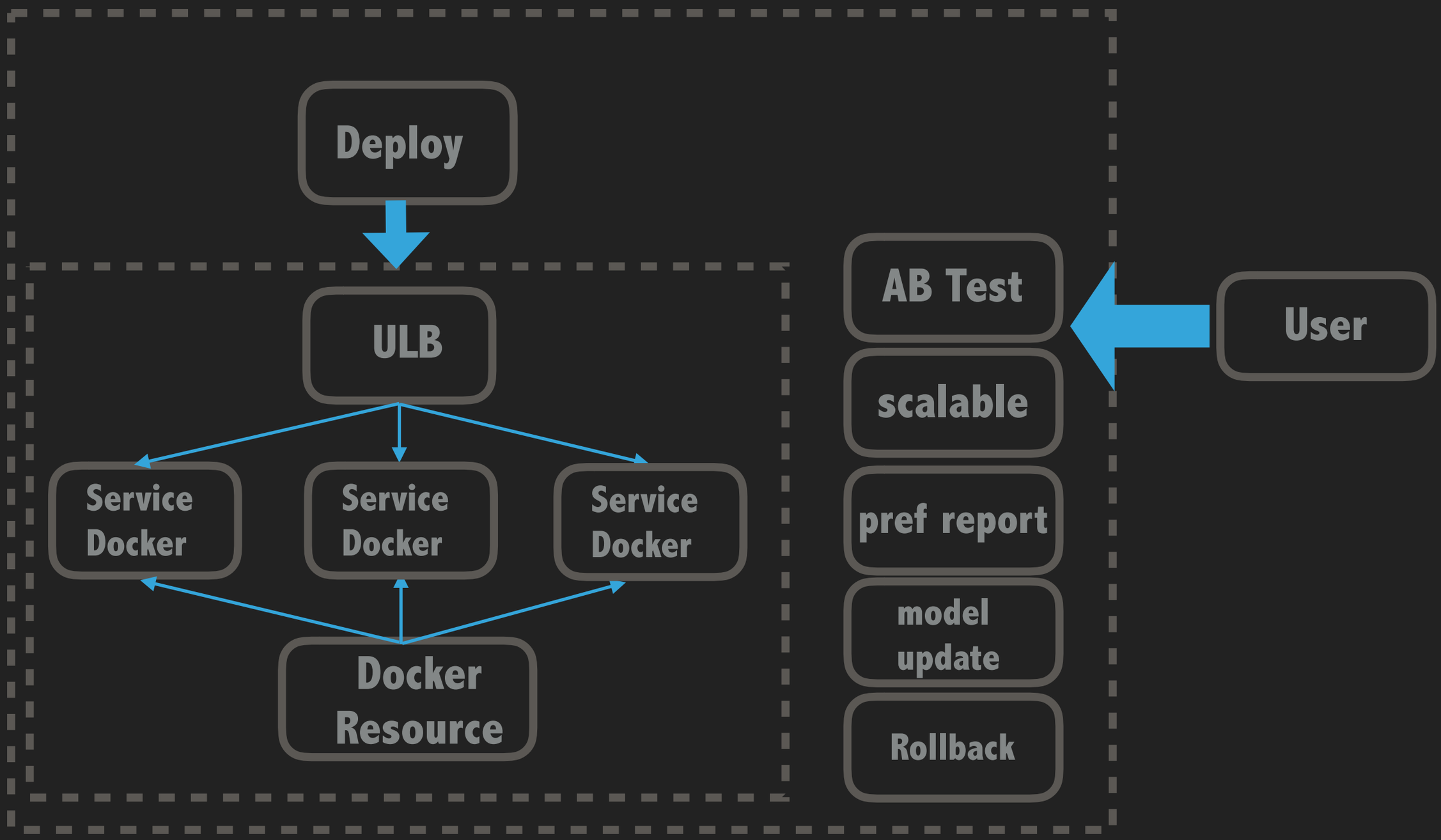
Online Inference I



Online Inference II



Online Inference III



Summary

- A toy Example
- TensorFlow Application In UCloud
  - Distributed Training
  - Online Service

Thank you

Q&A Offline

- Q1: preprocess详解[1]
  - data normalization
    - simple rescalling(scale the feature to a new [min, max])
    - per-example mean subtraction(each dimension follow the sample distribution)
    - feature standardization(zero-mean and unit var, svm)
  - Whitening
    - PCA
    - ZCA
  - ...

PS 图像因为0-255，通常在tf中是转成0-1.0的float32，在我自己的工作中没有做相应的simple rescalling的工作，MXNet中默认是使用0-255会使用per-example mean sub，通常会有mean.bin来做相应图像的sub后，去更新网络。白化，因为要计算cov matrix，所以通常不适合比较大的图像

- Q2: Image Augmentation
  - flip (left->right, up->down)
  - random brightness
  - random rotation
  - random zoom
  - random channal
  - random_crop
  - ...

- Q3: Ticks in DeepLearning

details in [4]

preprocess and augmentation in Keras

```
keras.preprocessing.image.ImageDataGenerator(featurewise_center=False,
    samplewise_center=False,
    featurewise_std_normalization=False,
    samplewise_std_normalization=False,
    zca_whitening=False,
    rotation_range=0.,
    width_shift_range=0.,
    height_shift_range=0.,
    shear_range=0.,
    zoom_range=0.,
    channel_shift_range=0.,
    fill_mode='nearest',
    cval=0.,
    horizontal_flip=False,
    vertical_flip=False,
    rescale=None,
    dim_ordering=K.image_dim_ordering())
```


[1]: http://ufldl.stanford.edu/wiki/index.php/Data_Preprocessing

[2]: https://keras.io/preprocessing/image/

[3]:random crop in keras: https://github.com/fchollet/keras/issues/3338

[4]: http://lamda.nju.edu.cn/weixs/project/CNNTricks/CNNTricks.html