TensorFlow

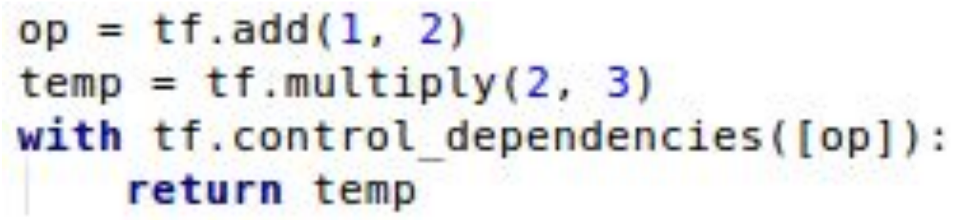
Курс “Практическое применение по TensorFlow”

Шигапова Фирюза Зинатуллаевна

1-й семестр, 2019 г.

https://github.com/Firyuza/TensorFlowPractice

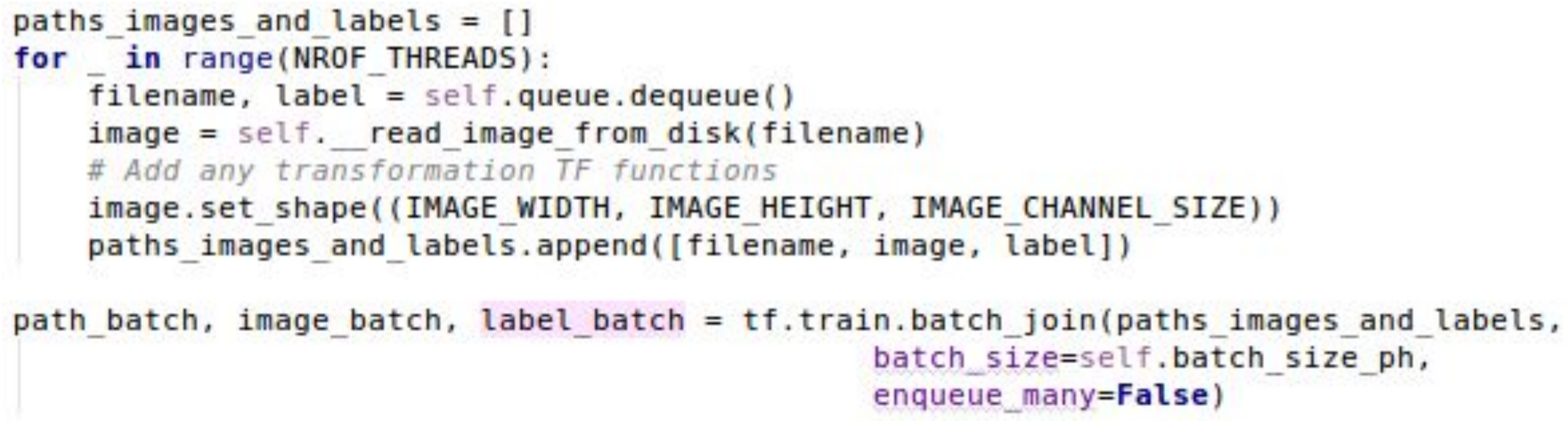
Quiz. How these operations will run? Explain.



```python
op = tf.add(1, 2)
temp = tf.multiply(2, 3)
with tf.control_dependencies([op]):
    return temp
```

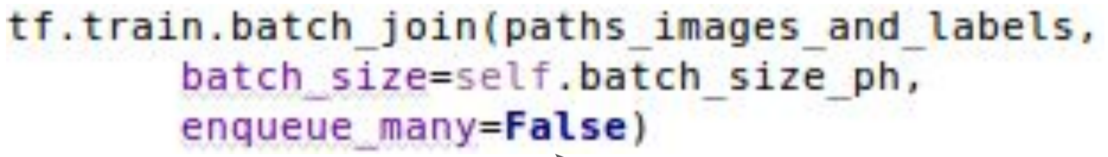
TF Queue. Dequeue multithreading

1. **Extract** from queue one element: queue.**dequeue**
2. **Transform** extracted item: apply any **tf functions**
3. **Load** into batch: **tf.train.batch**
4. **Parallelize** all previous steps via “*for loop*”



```python
paths_images_and_labels = []
for _ in range(NROF_THREADS):
    filename, label = self.queue.dequeue()
    image = self.__read_image_from_disk(filename)
    # Add any transformation TF functions
    image.set_shape((IMAGE_WIDTH, IMAGE_HEIGHT, IMAGE_CHANNEL_SIZE))
    paths_images_and_labels.append([filename, image, label])

path_batch, image_batch, label_batch = tf.train.batch_join(paths_images_and_labels,
                                                           batch_size=self.batch_size_ph,
                                                           enqueue_many=False)
```

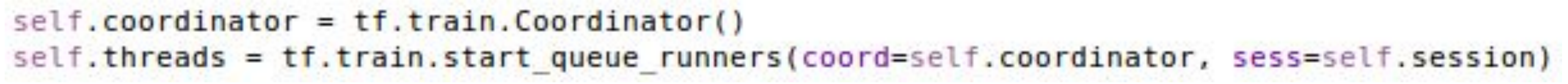
TF Queue. Dequeue multithreading



```
tf.train.batch_join(paths_images_and_labels,
          batch_size=self.batch_size_ph,
          enqueue_many=False)
```

enqueue_many: **False** means that every item in *paths_images_and_labels* list is **single** item itself.
True means that every item in *paths_images_and_labels* list is **batch** of elements

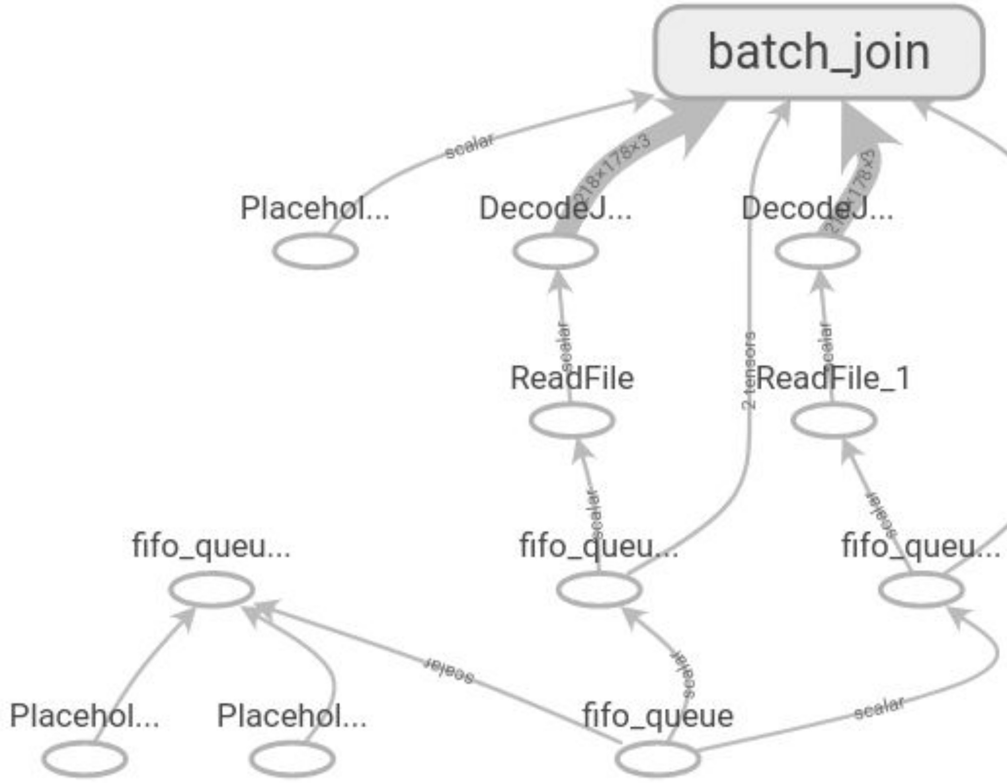
TF Queue. Dequeue multithreading



```
self.coordinator = tf.train.Coordinator()
self.threads = tf.train.start_queue_runners(coord=self.coordinator, sess=self.session)
```

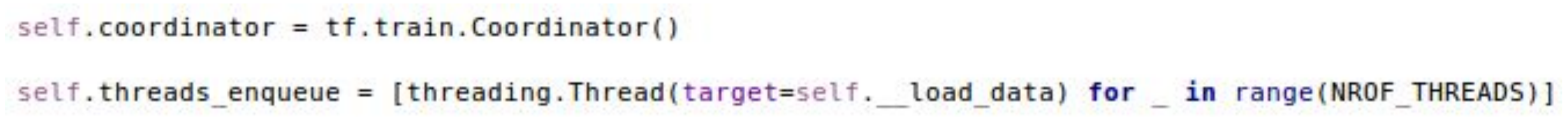
Coordinator for thread controlling
QueueRunner for creating multithreading for *dequeue* operation

TF Queue. Dequeue multithreading. Graph



TF Queue. Enqueue multithreading

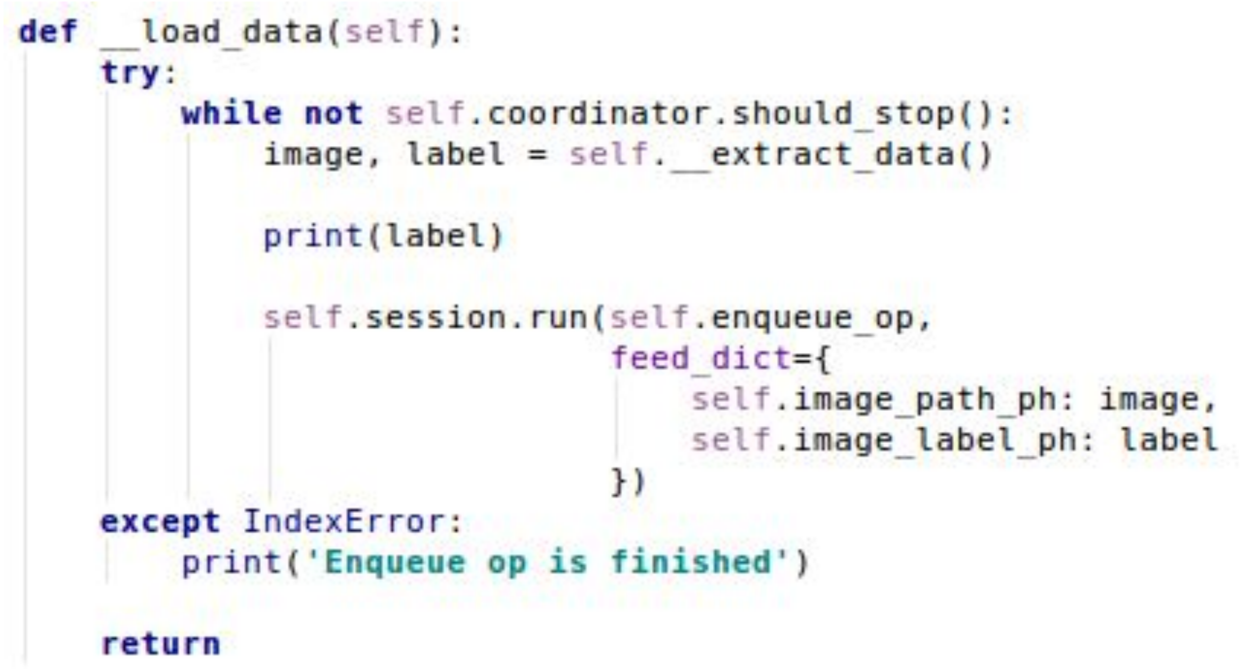
1. **Extract** data asynchronously: python **threading** library



```python
self.coordinator = tf.train.Coordinator()

self.threads_enqueue = [threading.Thread(target=self.__load_data) for _ in range(NROF_THREADS)]
```

TF Queue. Enqueue multithreading

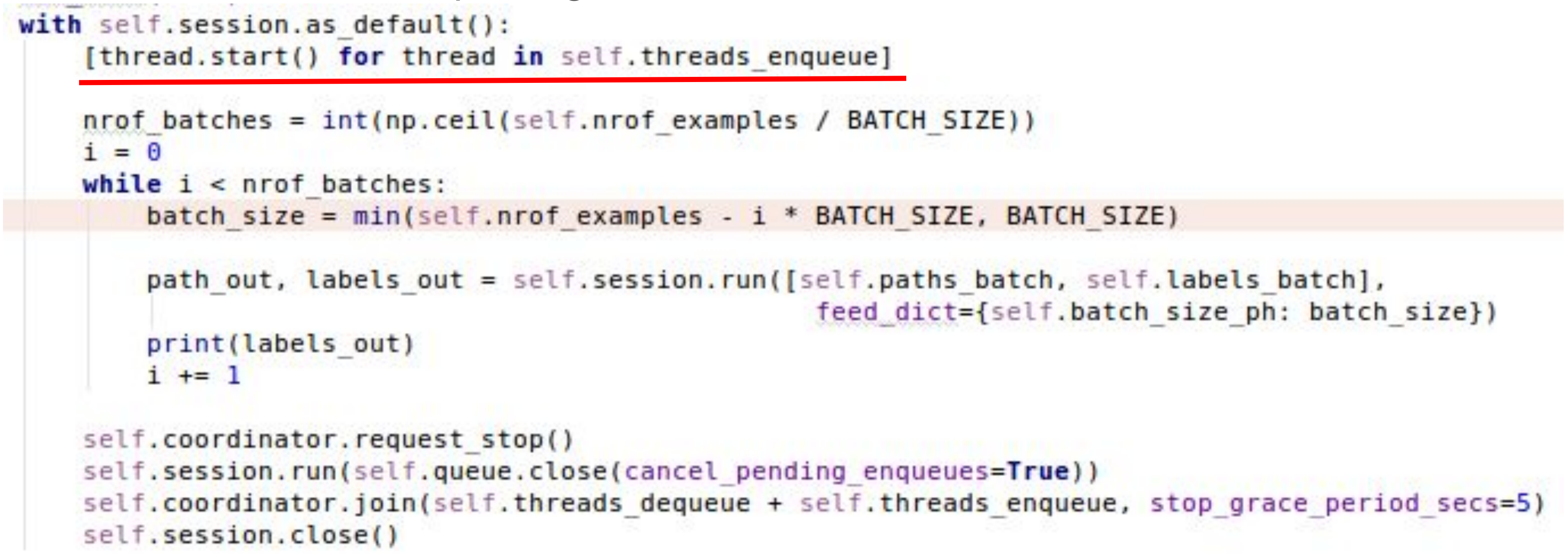
2. And **add** into queue



```python
def __load_data(self):
    try:
        while not self.coordinator.should_stop():
            image, label = self.__extract_data()

            print(label)

            self.session.run(self.enqueue_op,
                             feed_dict={
                                 self.image_path_ph: image,
                                 self.image_label_ph: label
                             })
    except IndexError:
        print('Enqueue op is finished')

    return
```

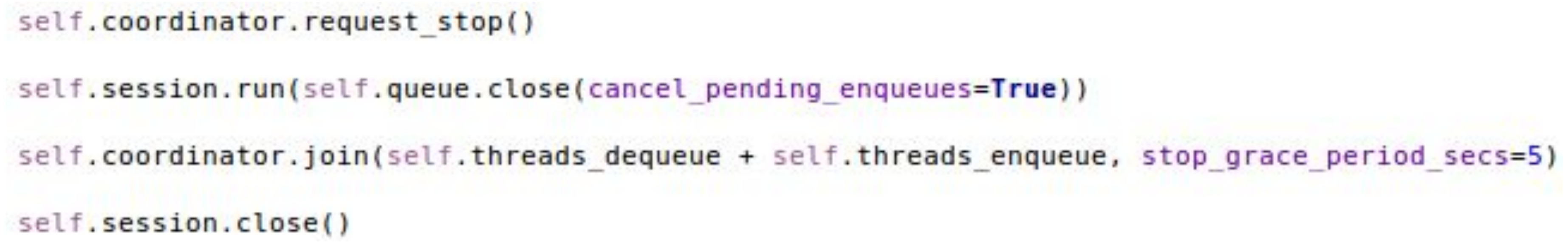
TF Queue. Enqueue multithreading

2. Run threads for enqueuing



```
with self.session.as_default():
    [thread.start() for thread in self.threads_enqueue]

    nrof_batches = int(np.ceil(self.nrof_examples / BATCH_SIZE))
    i = 0
    while i < nrof_batches:
        batch_size = min(self.nrof_examples - i * BATCH_SIZE, BATCH_SIZE)

        path_out, labels_out = self.session.run([self.paths_batch, self.labels_batch],
                                                 feed_dict={self.batch_size_ph: batch_size})
        print(labels_out)
        i += 1

    self.coordinator.request_stop()
    self.session.run(self.queue.close(cancel_pending_enqueues=True))
    self.coordinator.join(self.threads_dequeue + self.threads_enqueue, stop_grace_period_secs=5)
    self.session.close()
```

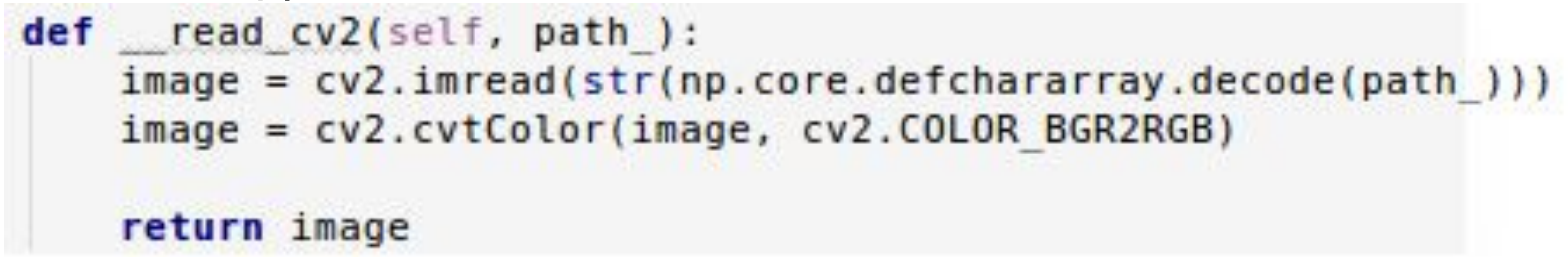
How to Finish



```
self.coordinator.request_stop()

self.session.run(self.queue.close(cancel_pending_enqueues=True))

self.coordinator.join(self.threads_dequeue + self.threads_enqueue, stop_grace_period_secs=5)

self.session.close()
```

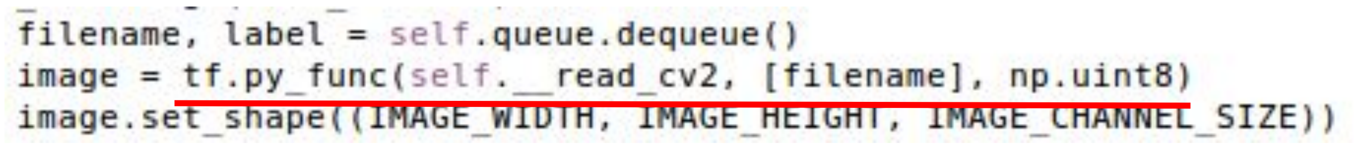
How to Insert Python Function

1. **Define** python function:



```
def __read_cv2(self, path_):
    image = cv2.imread(str(np.core.defchararray.decode(path_)))
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    return image
```

2. Call it through **tf.py_func:**



```
filename, label = self.queue.dequeue()
image = tf.py_func(self.__read_cv2, [filename], np.uint8)
image.set_shape((IMAGE_WIDTH, IMAGE_HEIGHT, IMAGE_CHANNEL_SIZE))
```

tf.py_func to Debug

tf.py_func (or **tf.py_function**) can be used as a way of **Debugging**

TF FIFO Queue for image processing

1. 5-10 images with labels.
2. Use TF **FIFO** Queue for preprocessing images through **batch** via QueueRunner & Coordinator.
3. Apply any transformation functions to images

   (rotation, crop, resize etc. -- **tf.image**).

4. Save processed images.