GDG Carthage

DevFest

Hands on Tensorflow's Eager Execution

InstaDeep Team:

Rihab Gorsane

Mohamed Abid

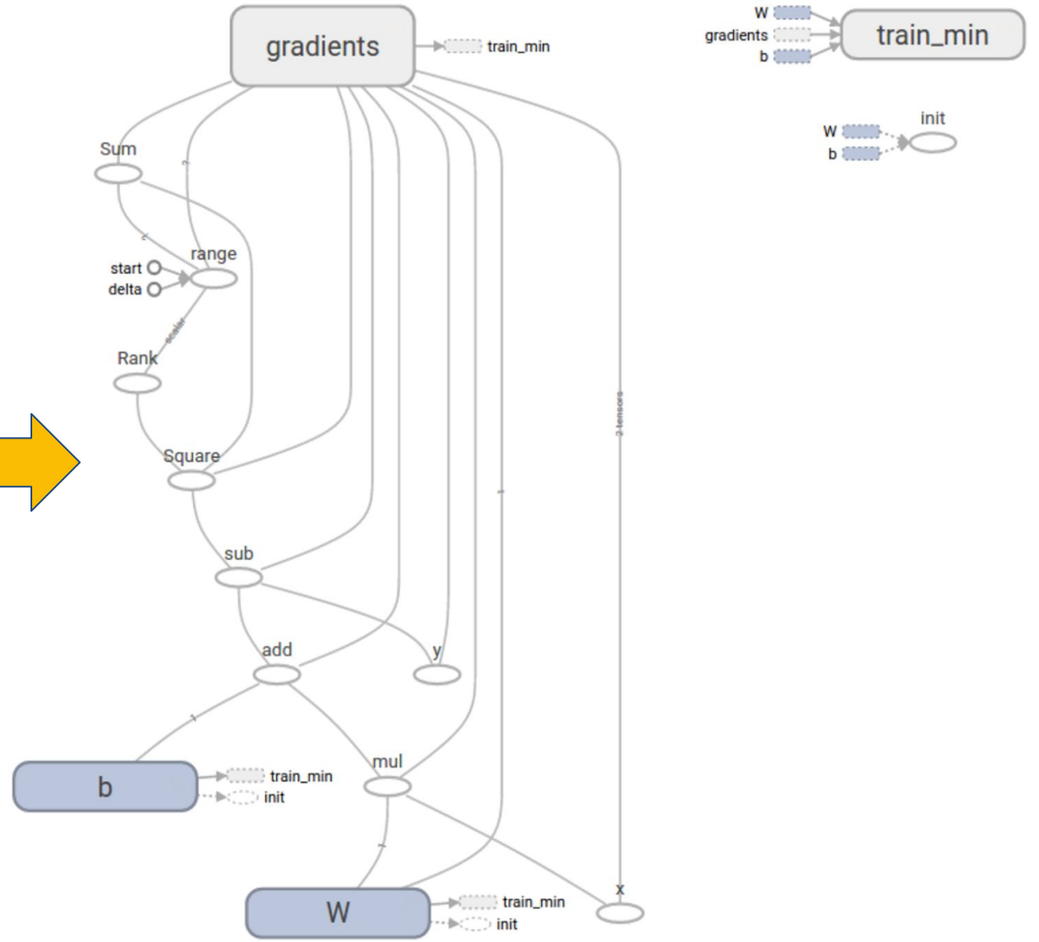
Hamdi Belhassen

Declarative TensorFlow: (Graphs)

```python
import numpy as np
import tensorflow as tf

# Model parameters
W = tf.Variable([.3], tf.float32)
b = tf.Variable([-.3], tf.float32)
# Model input and output
x = tf.placeholder(tf.float32)
linear_model = W * x + b
y = tf.placeholder(tf.float32)
# loss
loss = tf.reduce_sum(tf.square(linear_model - y)) # sum of the squares
# optimizer
optimizer = tf.train.GradientDescentOptimizer(0.01)
train = optimizer.minimize(loss)
# training data
x_train = [1,2,3,4]
y_train = [0,-1,-2,-3]
# training loop
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init) # reset values to wrong
for i in range(1000):
  sess.run(train, {x:x_train, y:y_train})

# evaluate training accuracy
curr_W, curr_b, curr_loss  = sess.run([W, b, loss], {x:x_train, y:y_train})
print("W: %s b: %s loss: %s"%(curr_W, curr_b, curr_loss))
```



Graphs are ...

Optimizable

- automatic buffer reuse
- reduce running time and the memory usage

Deployable

- use the same graph to deploy in different platforms

But graphs are also ...

Difficult to debug

- errors are reported long after graph construction
- execution cannot be debugged with print statements

Un-Pythonic

- cannot use if statements, objects, control-flow ops
- can't easily mix graph construction with custom data structures

Eager Execution

"A NumPy-like library for numerical computation with support for GPU acceleration and automatic differentiation, and a flexible platform for machine learning research and experimentation."

Imports and setup:

```
import tensorflow as tf
import tensorflow.contrib.eager as tfe

tf.enable_eager_execution()
```

Printing tensors: (~~Placeholders~~, ~~Sessions~~)

```python
x = tf.placeholder(tf.float32, shape=[1, 1])
m = tf.matmul(x, x)

print(m)
# Tensor("MatMul:0", shape=(1, 1), dtype=float32)

with tf.Session() as sess:
  m_out = sess.run(m, feed_dict={x: [[2.]]})

print(m_out)
# [[4.]]
```

Code like this...

```python
x = [[2.]]  # No need for placeholders!
m = tf.matmul(x, x)

print(m)  # No sessions!
# tf.Tensor([[4.]], shape=(1, 1), dtype=float32)
```

Becomes this

Variables

```
# Declare the variable
w = tf.get_variable('weights',
initializer=tf.random_normal([2,3], stddev=0.2))

with tf.Session() as session:
# Initialize values of all variable tensors
    tf.global_variables_initializer().run()
    print(sess.run(w))
```

```
# No additional initialization of variables is
required.
w = tfe.Variable(tf.random_normal([2,3], stddev=0.2),
name='weights')
print(w)
```

Iterating over Data:

```
# make a dataset from a list
words = tf.constant(['cat', 'dog', 'house', 'car'])
dataset = tf.data.Dataset.from_tensor_slices(words)

# create the iterator
iter = dataset.make_one_shot_iterator()

with tf.Session() as sess:
    el= iter.get_next()
    print(sess.run(el))
```

```
words = tf.constant(['cat', 'dog', 'house', 'car'])
dataset = tf.data.Dataset.from_tensor_slices(words)

for x in dataset:
    print(x)
```

Gradients:

Automatic differentiation is built into eager execution

Under the hood ...

- Operations are recorded on a **tape**
- The tape is **played back** to compute gradients
  - This is reverse-mode differentiation (backpropagation).

Gradients:

```
variables = [w1, b1, w2, b2]


optimizer = tf.train.AdamOptimizer()
with tf.GradientTape() as tape:
    y_pred = model.predict(x, variables)
    loss = model.compute_loss(y_pred, y)
    grads = tape.gradient(loss, variables)
    optimizer.apply_gradients(zip(grads, variables))
```

Saving:

```
variables = [w1, b1, w2, b2]
saver = tfe.Saver(variables)

# do some training…
saver.save('checkpoints/checkpoint.ckpt', global_step=step)

# to restore from checkpoint
checkpoint_path = tf.train.latest_checkpoint('checkpoints')
saver.restore(checkpoint_path)
```

Writing eager-compatible code

- Use tf.keras.layers
- Use tf.keras.Model
- Use tf.contrib.summary
- Use tfe.metrics
- Use object-based saving

-

GDG Carthage

DevFest

Thank you !!