Eager Execution

github.com/tensorflow/early-eager

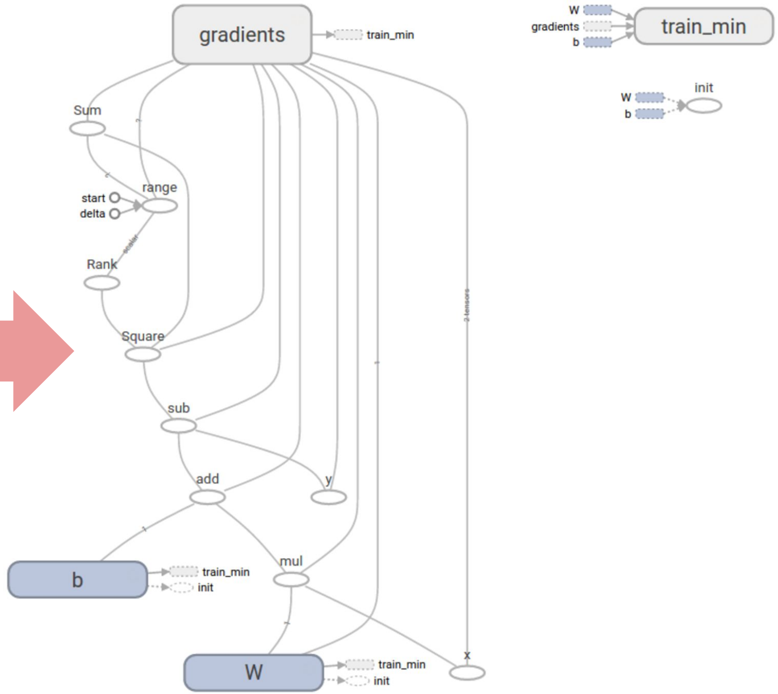
TensorFlow
FALL SYMPOSIUM

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 can be annoying

Delayed feedback

- Error reporting much after graph construction
- Not friendly to host-language debugger/tools

TensorFlow
FALL SYMPOSIUM

Graphs can be annoying

Metaprogramming

- Control flow concepts (`tf.while_loop`) are different than the host language
- Can't use Python data structures easily

TensorFlow
FALL SYMPOSIUM

Eager Execution

As simple as possible

TensorFlow
FALL SYMPOSIUM

Boilerplate

```
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...

~~Boilerplate~~

```
x = [[2.]]
m = tf.matmul(x, x)

print(m)
# tf.Tensor([[4.]], dtype=float32, shape=(1,1))
```

Becomes this

Instant Errors

```
x = tf.gather([0, 1, 2], 7)

InvalidArgumentError: indices = 7 is not in [0, 3) [Op:Gather]
```

Metaprogramming boo boos

```
x = tf.random_uniform([2, 2])

with tf.Session() as sess:
  for i in range(x.shape[0]):
    for j in range(x.shape[1]):
      print(sess.run(x[i, j]))
```

Each iteration
adds nodes to the graph

~~Metaprogramming boo boos~~

```
x = tf.random_uniform([2, 2])

for i in range(x.shape[0]):
  for j in range(x.shape[1]):
    print(x[i, j])
```

Python Control Flow

```
a = tf.constant(6)
while a != 1:
  if a % 2 == 0:
    a = a / 2
  else:
    a = 3 * a + 1
  print(a)
```

```
# Outputs
tf.Tensor(3, dtype=int32)
tf.Tensor(10, dtype=int32)
tf.Tensor(5, dtype=int32)
tf.Tensor(16, dtype=int32)
tf.Tensor(8, dtype=int32)
tf.Tensor(4, dtype=int32)
tf.Tensor(2, dtype=int32)
tf.Tensor(1, dtype=int32)
```

Gradients

TensorFlow
FALL SYMPOSIUM

Gradients

- Operations executed are recorded on a tape
- Tape is played back to compute gradients

TensorFlow
FALL SYMPOSIUM

Gradients

```
def square(x):
  return tf.multiply(x, x)  # Or x * x

grad = tfe.gradients_function(square)



print(square(3.))    # tf.Tensor(9., dtype=tf.float32
print(grad(3.))      # [tf.Tensor(6., dtype=tf.float32))]
```

Gradients

```
def square(x):
  return tf.multiply(x, x)  # Or x * x

grad = tfe.gradients_function(square)
gradgrad = tfe.gradients_function(lambda x: grad(x)[0])


print(square(3.))     # tf.Tensor(9., dtype=tf.float32)
print(grad(3.))       # [tf.Tensor(6., dtype=tf.float32)]
print(gradgrad(3.))   # [tf.Tensor(2., dtype=tf.float32))]
```

Customizing Gradients

```
def f(x):
  y = tf.square(x)
  return y



grad = tfe.gradients_function(f)
print(f(3.))       # 9.
print(grad(3.))   # [6.]
```

Customizing Gradients

```
@tfe.custom_gradient
def f(x):
  y = tf.square(x)
  def grad_fn(dresult):
    return [x + y]
  return prod, grad_fn

grad = tfe.gradients_function(f)
print(f(3.))      # 9.
print(grad(3.))   # [12.]
```

It’s not *that* different

TensorFlow
FALL SYMPOSIUM

TensorFlow = Operation Kernels + Composition

- Session: One way to compose operations
- Eager execution: Compose using Python

Using GPUs

`tf.device()` for manual placement

```
with tf.device(“/gpu:0”):
  x = tf.random_uniform([10, 10])
  y = tf.matmul(x, x)
  # x and y reside in GPU memory
```

Building Models

The same APIs as graph building (`tf.layers`, `tf.train.Optimizer`, `tf.data` etc.)

```
model = tf.layers.Dense(units=1, use_bias=True)
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1)
```

Building Models

```python
model = tf.layers.Dense(units=1, use_bias=True)
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1)

# Define a loss function
def loss(x, y):
  return tf.reduce_mean(tf.square(y - model(x)))
```

Training Models

Compute and apply gradients

```
for (x, y) in get_next_batch():
  optimizer.apply_gradients(grad_fn(x, y))
```

Training Models

Compute and apply gradients

```
grad_fn = tfe.implicit_gradients(loss)

for (x, y) in get_next_batch():
  optimizer.apply_gradients(grad_fn(x, y))
```

No more graphs then?

Graphs are

Optimizable

- Automatic buffer reuse
- Constant folding
- Inter-op parallelism
- Automatic trade-off between compute and memory

Graphs are

Deployable

- TensorFlow Serving
- Mobile
- Any other C++/Java/other program

Without loss in translation between runtimes

TensorFlow FALL SYMPOSIUM

Graphs are

Transformable

- Carve out subgraphs to offload to accelerators
- Train with quantization in mind

Graph Functions

TensorFlow
FALL SYMPOSIUM

Graph Functions

“Compile” Python functions into graphs

- Mix eager execution with calls to “compiled” graphs
- Differentiate through graphs

TensorFlow
FALL SYMPOSIUM

LSTM Cell

```python
def lstm_cell(x, w, h, c):
  xhw = tf.matmul(tf.concat([x, h], axis=1), w)
  y = tf.split(xhw, 4, axis=1)
  in_value = tf.tanh(y[0])
  in_gate, forget_gate, out_gate = [tf.sigmoid(x) for x in y[1:]]
  c = (forget_gate * c) + (in_gate * in_value)
  h = out_gate * tf.tanh(c)
  return h, c

h, c = lstm_cell(x, w, h, c)
print(h)
```

LSTM Cell

```
@tfe.graph_callable
def lstm_cell(x, w, h, c):
  xhw = tf.matmul(tf.concat([x, h], axis=1), w)
  y = tf.split(xhw, 4, axis=1)
  in_value = tf.tanh(y[0])
  in_gate, forget_gate, out_gate = [tf.sigmoid(x) for x in y[1:]]
  c = (forget_gate * c) + (in_gate * in_value)
  h = out_gate * tf.tanh(c)
  return h, c

h, c = lstm_cell(x, w, h, c)
print(h)
```

LSTM Cell

```
@tfe.graph_callable
def lstm_cell(x, w, h, c):
  xhw = tf.matmul(tf.concat([x, h], axis=1), w)
  y = tf.split(xhw, 4, axis=1)
  in_value = tf.tanh(y[0])
  in_gate, forget_gate, out_gat                          x in y[1:]]
  c = (forget_gate * c) + (in_g
  h = out_gate * tf.tanh(c)
  return h, c

h, c = lstm_cell(x, w, h, c)
print(h)
```

tanh executed in-place

Use existing graph code

```
@tfe.graph_callable
def inception(image):
    logits = inception.inception_v3(image,
                                    num_classes=1001,
                                    is_training=False)[0]
    return tf.softmax(logits)

inception.restore("/path/to/checkpoint")
print(len(inception.variables))
```

Mixing graphs and eager

TensorFlow
FALL SYMPOSIUM

Define an objective

```
def objective(x):
  # Label 283 is that of a tabby cat
  log_p = tf.log(inception(x)[0, 283])
  return -tf.reshape(log_p, []) # scalar


# Gradient with respect to input
grad_fn = tfe.gradients_function(objective)
```

inception is a graph

Differentiating through the graph

Line search

```
image = load_image("myimage.jpg")
for _ in range(10):
  val, grad = objective(image), grad_fn(image)
  print("-log(tabby cat-iness): {}".format(val))
  g_norm = tf.reduce_sum(grad * grad)
```

Line search

```
image = load_some_image()
for _ in range(10):
  val, grad = objective(image), grad_fn(image)
  print("-log(tabby cat-iness): {}".format(val))
  g_norm = tf.reduce_sum(grad*grad)

  (init_val, init_image, rate) = (val, image, 1.0)
  while val > init_val - rate * g_norm:
    image, rate = init_image - rate * grad, rate / 2.0
    val = objective(image)
```

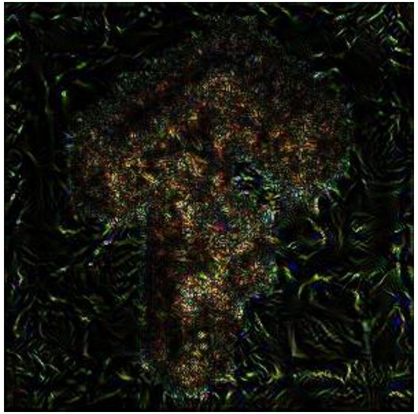
Nudging pixels towards a "tabby cat"

Cat generator?

\+ ? = CA
T

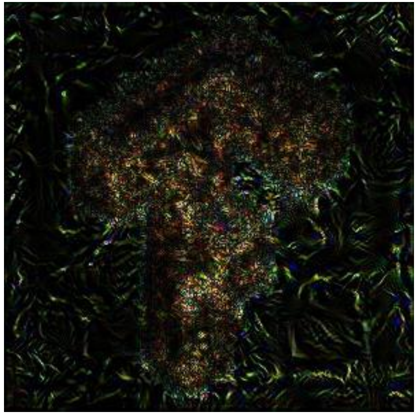
Cat generator?

+



= CAT

Cat generator?

\+  =

How does my code change?

TensorFlow
FALL SYMPOSIUM

One call to change behavior

```
import tensorflow.contrib.eager as tfe
tfe.enable()
```

- `tf.Tensor` objects hold a concrete value on a device
- Operations execute immediately

TensorFlow
FALL SYMPOSIUM

A few new functions

Work whether eager execution is enabled or not:

- `gradients_function(f)`
- `implicit_gradients(f)`
- `custom_gradients` decorator

TensorFlow FALL SYMPOSIUM

...and lots of benefits

- Python control flow
- Structured programming
- Instant errors
- Friendly to standard tools (Python debugger, print)

TensorFlow
FALL SYMPOSIUM

Status

Status

- Alpha/Preview version out soon
- Single GPU, ResNet benchmark performance comparable to graphs
- Overheads on smaller models is higher

- Watch the release notes for upcoming TensorFlow releases for updates
- Or follow along on Github: `tensorflow/contrib/eager/README.md`
- github.com/tensorflow/early-eager

TensorFlow

FALL SYMPOSIUM

Thank you!