Introduction to TensorFlow

Mor Geva, Apr 2018

Introduction to TensorFlow

Mor Geva, Apr 2018

Plan

- Why TensorFlow
- Basic Code Structure
- Example: Learning Word Embeddings with Skip-gram
- Variable and Name Scopes
- Visualization with TensorBoard

Plan

- Why TensorFlow
- Basic Code Structure
- Example: Learning Word Embeddings with Skip-gram
- Variable and Name Scopes
- Visualization with TensorBoard

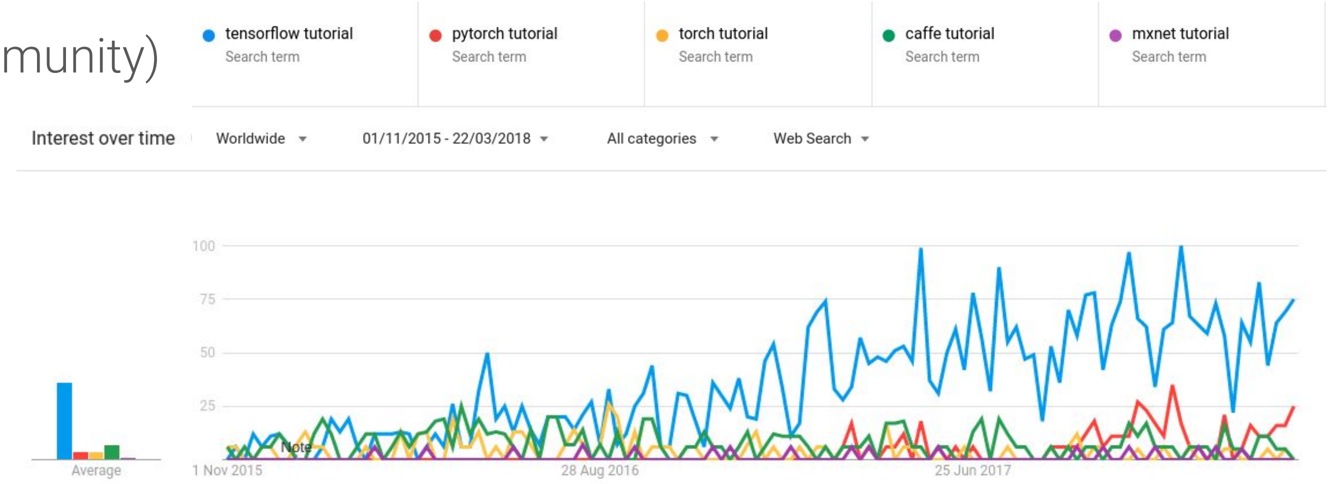
Disclaimer I'm not a TF expert, just passing on knowledge I have

Goals

- Understand the basic structure of a TensorFlow program
- Be familiar with the main code components
- Understand how to assemble them to train a neural model

Why TensorFlow

- “TensorFlow™ is an open source software library for numerical computation using data flow graphs.”
- One of many frameworks for deep learning computations
- Scalable and flexible
- Popular (= big community)



Basic Code Structure

- View functions as computational graphs
- First build a computational **graph**, and then use a **session** to execute operations in the graph
- This is the basic approach, there is also a dynamic approach implemented in the recently introduced eager mode

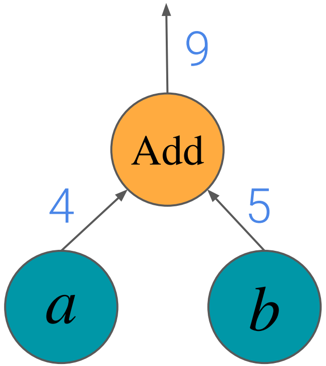
Basic Code Structure

- View functions as computational graphs
- First build a computational **graph**, and then use a **session** to execute operations in the graph
- This is the basic approach, there is also a dynamic approach implemented in the recently introduced eager mode

why graphs?

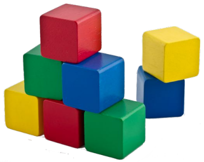
Basic Code Structure - Graphs

- Nodes are operators (ops), variables, and constants
- Edges are tensors
  - 0-d is a scalar
  - 1-d is a vector
  - 2-d is a matrix
  - Etc.
- TensorFlow = Tensor + Flow = Data + Flow



Basic Code Structure - Graphs

- **Constants** are fixed value tensors - not trainable
- **Variables** are tensors initialized in a session - trainable
- **Placeholders** are tensors of values that are unknown during the graph construction, but passed as input during a session
- **Ops** are functions on tensors



Basic Code Structure - Graphs

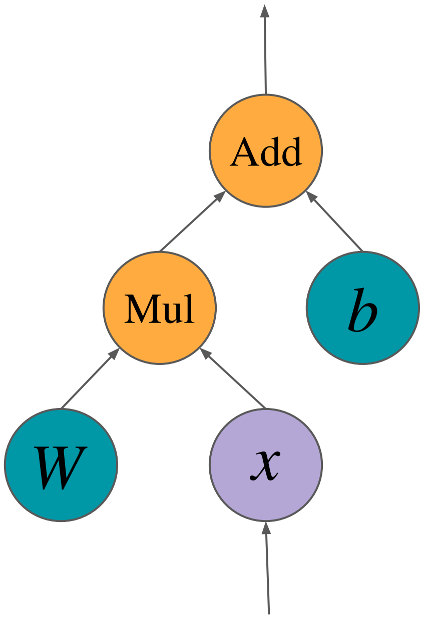
$$\hat{y} = Wx + b$$

variable

ops

placeholder

variable



Basic Code Structure - Sessions

- Session is the runtime environment of a graph, where operations are executed, and tensors are evaluated

```
>>> import tensorflow as tf
>>> a = tf.constant(4)
>>> b = tf.constant(5)
>>> add_op = tf.add(a, b)
>>> print(add_op)
Tensor("Add:0", shape=(), dtype=int32)
```

```
>>> import tensorflow as tf
>>> a = tf.constant(4)
>>> b = tf.constant(5)
>>> add_op = tf.add(a, b)
>>> with tf.Session() as session:
...     print(session.run(add_op))
...
9
```

- `a.eval()` is equivalent to `session.run(a)`, but in general, “eval” is limited to executions of a single op and ops that returns a value

Basic Code Structure - Sessions

- Session is the runtime environment of a graph, where operations are executed, and tensors are evaluated

```
>>> import tensorflow as tf
>>> a = tf.constant(4)
>>> b = tf.constant(5)
>>> add_op = tf.add(a, b)
>>> print(add_op)
Tensor("Add:0", shape=(), dtype=int32)
```

```
>>> import tensorflow as tf
>>> a = tf.constant(4)
>>> b = tf.constant(5)
>>> add_op = tf.add(a, b)
>>> with tf.Session() as session:
...     print(session.run(add_op))
...
9
```

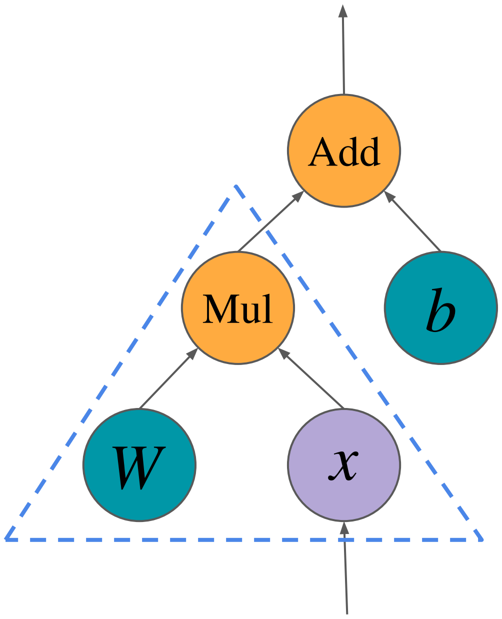
- `a.eval()` is equivalent to `session.run(a)`, but in general, “eval” is limited to executions of a single op and ops that returns a value
- Upon op execution, only the subgraph required for calculating its value is evaluated

Basic Code Structure - Sessions

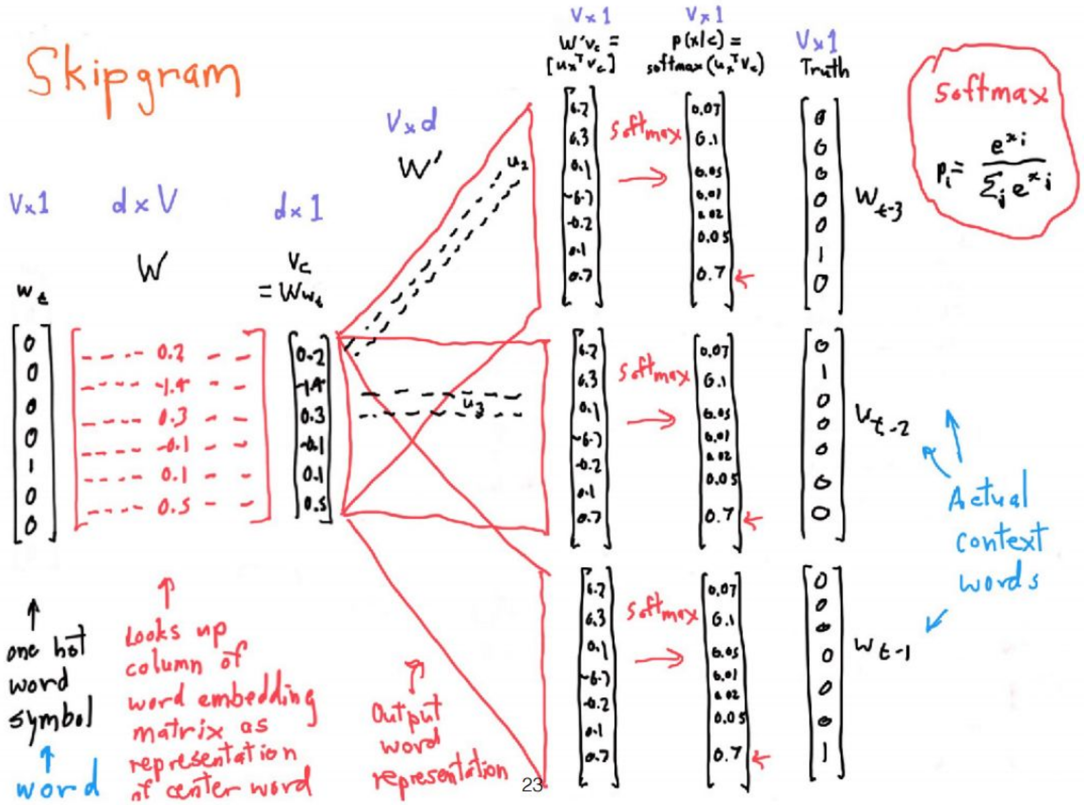
$$\hat{y} = \boxed{Wx} + b$$

variable   variable

ops   placeholder



Example: Learning Word Embeddings with Skip-gram



Recall from lecture 1

Example: Learning Word Embeddings with Skip-gram

- We will use Noise-Constructive Estimation (NCE) as our loss function, it is similar to negative sampling that you implemented in HW 1

Model:

$$p_\theta(y = 1 \mid c, o) = \frac{1}{1 + \exp(-u_o^\top v_c)} = \sigma(u_o^\top v_c)$$

$$p_\theta(y = 0 \mid c, o) = 1 - \sigma(u_o^\top v_c) = \sigma(-u_o^\top v_c)$$

Objective:

$$\sum_{t,j} \left(\log(\sigma(u_{w_{t+j}}^\top v_{w_t})) + \sum_{k \sim p(w)} \log(\sigma(-u_{w^{(k)}}^\top v_{w_t})) \right)$$

(x, y) = ((bank, holds), 1)
(x, y) = ((bank, table), 0)
(x, y) = ((bank, eat), 0)
(x, y) = ((holds, bank), 1)
(x, y) = ((holds, quickly), 0)
(x, y) = ((holds, which), 0)
(x, y) = ((the, mortgage), 1)
(x, y) = ((the, eat), 0)
(x, y) = ((the, who), 0)

Example: Learning Word Embeddings with Skip-gram

1. Assembling the graph
    - Create placeholders
    - Create variables
    - Define a loss function
    - Define an optimizer
2. Training in a session
    - Start a session
    - Initialize variables
    - Run the optimizer over batches

Example: Assembling the Graph

```
import tensorflow as tf

graph = tf.Graph()
with graph.as_default():
  train_inputs = tf.placeholder(tf.int32,shape=[batch_size])
  train_labels = tf.placeholder(tf.int32,shape=[batch_size, 1])

  embeddings = tf.Variable(tf.random_uniform([vocabulary_size, embedding_size],-1.0, 1.0))
  embed = tf.nn.embedding_lookup(embeddings, train_inputs)

  nce_weights = tf.Variable(tf.truncated_normal([vocabulary_size, embedding_size],
                            stddev=1.0 / math.sqrt(embedding_size)))
  nce_biases = tf.Variable(tf.zeros([vocabulary_size]))
  loss = tf.reduce_mean(
        tf.nn.nce_loss(weights=nce_weights, biases=nce_biases, labels=train_labels,
                       inputs=embed, num_sampled=num_sampled, num_classes=vocabulary_size))

  optimizer = tf.train.GradientDescentOptimizer(1.0).minimize(loss)

  init = tf.global_variables_initializer()
```

Adapted from: https://www.tensorflow.org/tutorials/word2vec

Example: Assembling the Graph

```
import tensorflow as tf

graph = tf.Graph()
with graph.as_default():
  train_inputs = tf.placeholder(tf.int32,shape=[batch_size])
  train_labels = tf.placeholder(tf.int32,shape=[batch_size, 1])

  embeddings = tf.Variable(tf.random_uniform([vocabulary_size, embedding_size],-1.0, 1.0))
  embed = tf.nn.embedding_lookup(embeddings, train_inputs)

  nce_weights = tf.Variable(tf.truncated_normal([vocabulary_size, embedding_size],
                          stddev=1.0 / math.sqrt(embedding_size)))
  nce_biases = tf.Variable(tf.zeros([vocabulary_size]))
  loss = tf.reduce_mean(
        tf.nn.nce_loss(weights=nce_weights, biases=nce_biases, labels=train_labels,
                       inputs=embed, num_sampled=num_sampled, num_classes=vocabulary_size))

  optimizer = tf.train.GradientDescentOptimizer(1.0).minimize(loss)

  init = tf.global_variables_initializer()
```

Adapted from: https://www.tensorflow.org/tutorials/word2vec

Example: Assembling the Graph

```
import tensorflow as tf

graph = tf.Graph()
with graph.as_default():
  train_inputs = tf.placeholder(tf.int32,shape=[batch_size])
  train_labels = tf.placeholder(tf.int32,shape=[batch_size, 1])

  embeddings = tf.Variable(tf.random_uniform([vocabulary_size, embedding_size],-1.0, 1.0))
  embed = tf.nn.embedding_lookup(embeddings, train_inputs)

  nce_weights = tf.Variable(tf.truncated_normal([vocabulary_size, embedding_size],
                            stddev=1.0 / math.sqrt(embedding_size)))
  nce_biases = tf.Variable(tf.zeros([vocabulary_size]))
  loss = tf.reduce_mean(
        tf.nn.nce_loss(weights=nce_weights, biases=nce_biases, labels=train_labels,
                       inputs=embed, num_sampled=num_sampled, num_classes=vocabulary_size))

  optimizer = tf.train.GradientDescentOptimizer(1.0).minimize(loss)

  init = tf.global_variables_initializer()
```

Adapted from: https://www.tensorflow.org/tutorials/word2vec

Example: Assembling the Graph

```
import tensorflow as tf

graph = tf.Graph()
with graph.as_default():
  train_inputs = tf.placeholder(tf.int32,shape=[batch_size])
  train_labels = tf.placeholder(tf.int32,shape=[batch_size, 1])

  embeddings = tf.Variable(tf.random_uniform([vocabulary_size, embedding_size],-1.0, 1.0))
  embed = tf.nn.embedding_lookup(embeddings, train_inputs)

  nce_weights = tf.Variable(tf.truncated_normal([vocabulary_size, embedding_size],
                            stddev=1.0 / math.sqrt(embedding_size)))
  nce_biases = tf.Variable(tf.zeros([vocabulary_size]))
  loss = tf.reduce_mean(
        tf.nn.nce_loss(weights=nce_weights, biases=nce_biases, labels=train_labels,
                       inputs=embed, num_sampled=num_sampled, num_classes=vocabulary_size))

  optimizer = tf.train.GradientDescentOptimizer(1.0).minimize(loss)

  init = tf.global_variables_initializer()
```

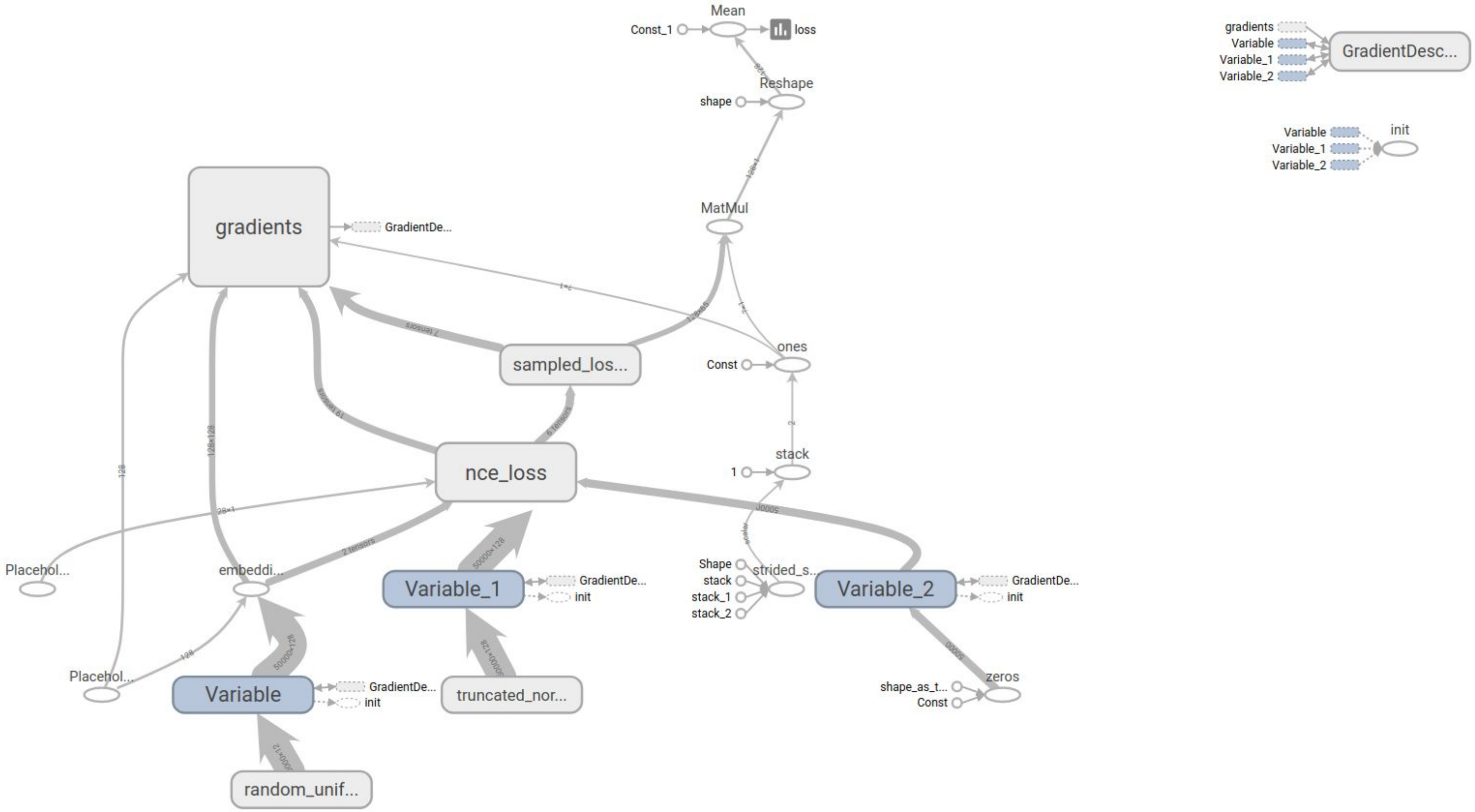
Adapted from: https://www.tensorflow.org/tutorials/word2vec

Example: Assembling the Graph

```
import tensorflow as tf

graph = tf.Graph()
with graph.as_default():
  train_inputs = tf.placeholder(tf.int32,shape=[batch_size])
  train_labels = tf.placeholder(tf.int32,shape=[batch_size, 1])

  embeddings = tf.Variable(tf.random_uniform([vocabulary_size, embedding_size],-1.0, 1.0))
  embed = tf.nn.embedding_lookup(embeddings, train_inputs)

  nce_weights = tf.Variable(tf.truncated_normal([vocabulary_size, embedding_size],
                          stddev=1.0 / math.sqrt(embedding_size)))
  nce_biases = tf.Variable(tf.zeros([vocabulary_size]))
  loss = tf.reduce_mean(
        tf.nn.nce_loss(weights=nce_weights, biases=nce_biases, labels=train_labels,
                       inputs=embed, num_sampled=num_sampled, num_classes=vocabulary_size))

  optimizer = tf.train.GradientDescentOptimizer(1.0).minimize(loss)

  init = tf.global_variables_initializer()
```

Adapted from: https://www.tensorflow.org/tutorials/word2vec

Example: Assembling the Graph



Example: Training in a Session

```
with tf.Session(graph=graph) as session:
  init.run()

  for step in xrange(num_steps):
    batch_inputs, batch_labels = generate_batch(batch_size, num_skips, skip_window)
    feed_dict = {train_inputs: batch_inputs, train_labels: batch_labels}

    _, loss_val = session.run([optimizer, loss],feed_dict=feed_dict)
```

Adapted from: https://www.tensorflow.org/tutorials/word2vec

Example: Training in a Session

```
with tf.Session(graph=graph) as session:
  init.run()

  for step in xrange(num_steps):
    batch_inputs, batch_labels = generate_batch(batch_size, num_skips, skip_window)
    feed_dict = {train_inputs: batch_inputs, train_labels: batch_labels}

    _, loss_val = session.run([optimizer, loss],feed_dict=feed_dict)
```

Adapted from: https://www.tensorflow.org/tutorials/word2vec

Example: Training in a Session

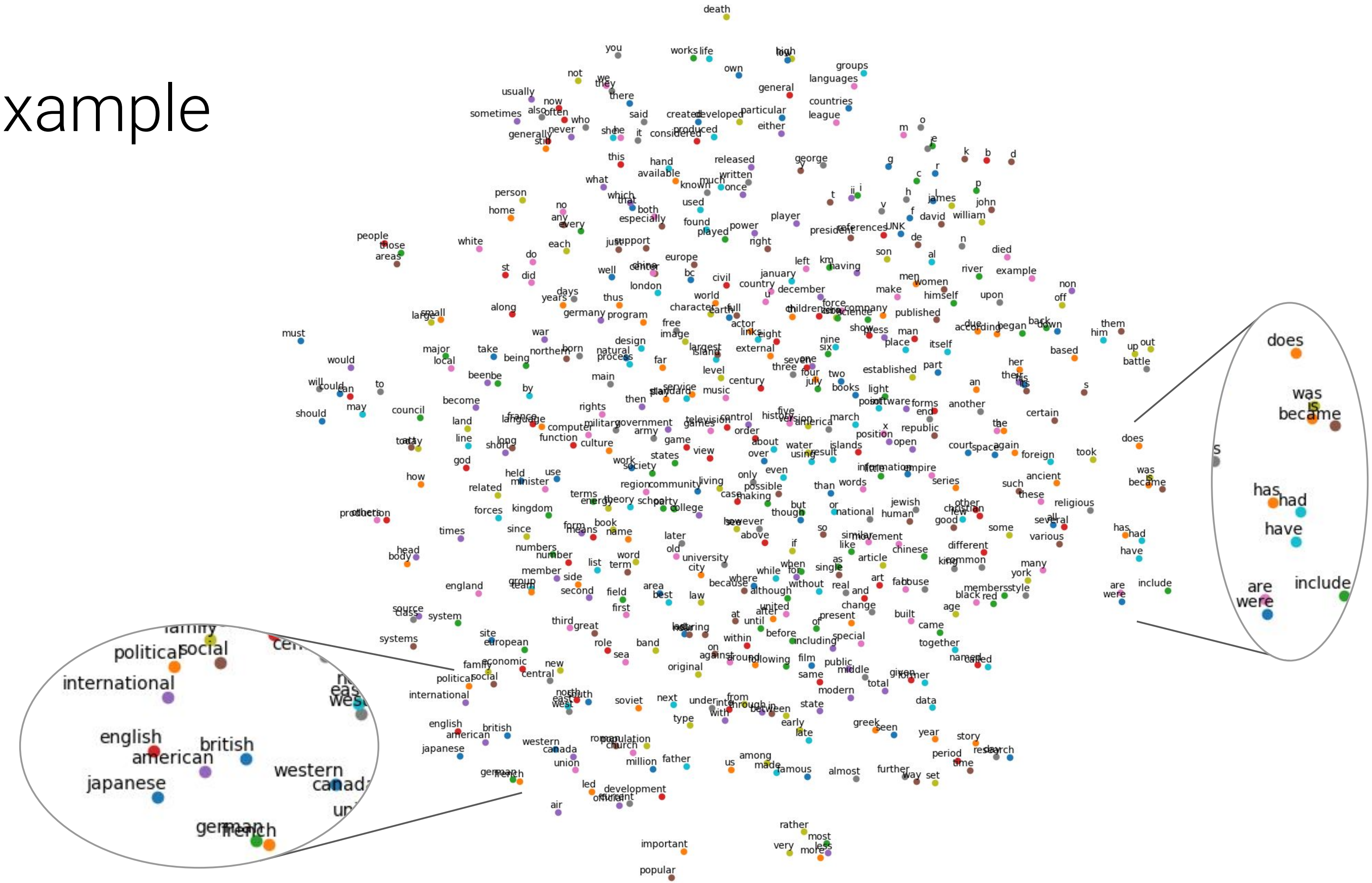
- You will probably want to save the model best parameters or store checkpoints
- Saving and restoring of session variables is done by creating a “saver” node, with `tf.train.Saver()`
- Note that only session variables are stored, and not the graph itself

Example: Training in a Session

```
# assembling the graph
...
saver = tf.train.Saver()

with tf.Session(graph=graph) as session:
  init.run()

  for step in xrange(num_steps):
    ...
    if step % 1000 == 0:
        saver.save(sess, save_path)
```

```
# assembling the graph
...
saver = tf.train.Saver()

with tf.Session(graph=graph) as session:
  saver.restore(sess, save_path)
```

Example



political
social
international
english
british
american
japanese
western
german
french
does
was
is
became
has
had
have
are
were
include

Plan

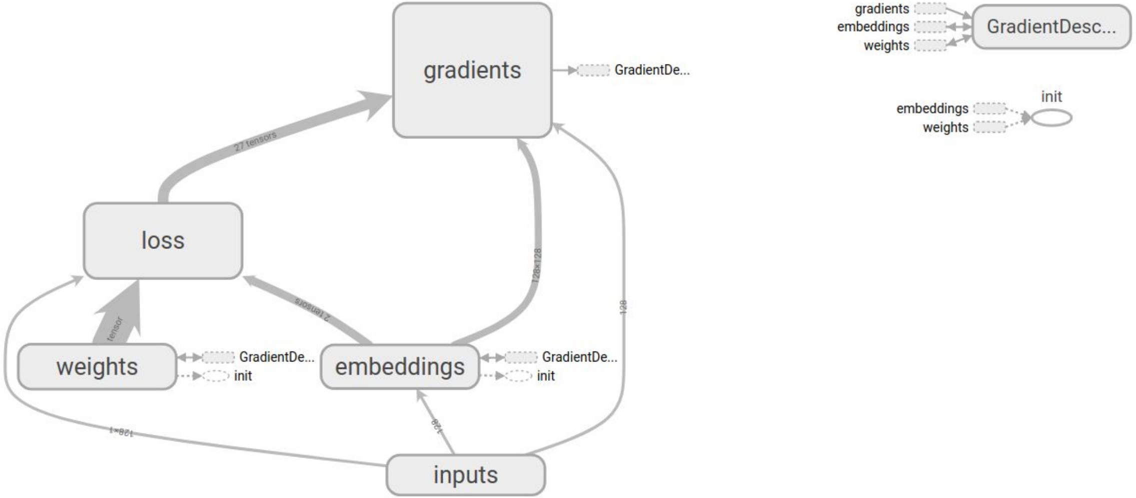
- ✓ Why TensorFlow
- ✓ Basic Code Structure
- ✓ Example: Learning Word Embeddings with Skip-gram
- Variable and Name Scopes
- Visualization with TensorBoard

Variable and Name Scopes

- Scopes allow:
  - Grouping of nodes in the graph
  - Sharing variables between graph components
- This is useful as neural networks can become very complex

Variable and Name Scopes

- Scopes allow:
    - Grouping of nodes in the graph
    - Sharing variables between graph components
- This is useful as neural networks can become very complex



Variable and Name Scopes

- `tf.Variable()` creates a new variable under the current scope
- `tf.get_variable()` creates the shared variable if it does not exist yet, or reuse it if it already exists
- The desired behavior is controlled by the current scope

Example from "*Hands-on machine learning with Scikit-Learn and TensorFlow*"

Variable and Name Scopes

- `tf.Variable()` creates a new variable under the current scope
- `tf.get_variable()` creates the shared variable if it does not exist yet, or reuse it if it already exists
- The desired behavior is controlled by the current scope

```
def relu(X, threshold):
    with tf.name_scope("relu"):
        [...]
        return tf.maximum(z, threshold, name="max")

threshold = tf.Variable(0.0, name="threshold")
X = tf.placeholder(tf.float32, shape=(None, n_features), name="X")
relus = [relu(X, threshold) for i in range(5)]
output = tf.add_n(relus, name="output")
```

Example from "*Hands-on machine learning with Scikit-Learn and TensorFlow*"

Variable and Name Scopes

1

```python
with tf.variable_scope("relu"):
    threshold = tf.get_variable("threshold", shape=(),
                                initializer=tf.constant_initializer(0.0))
```

2

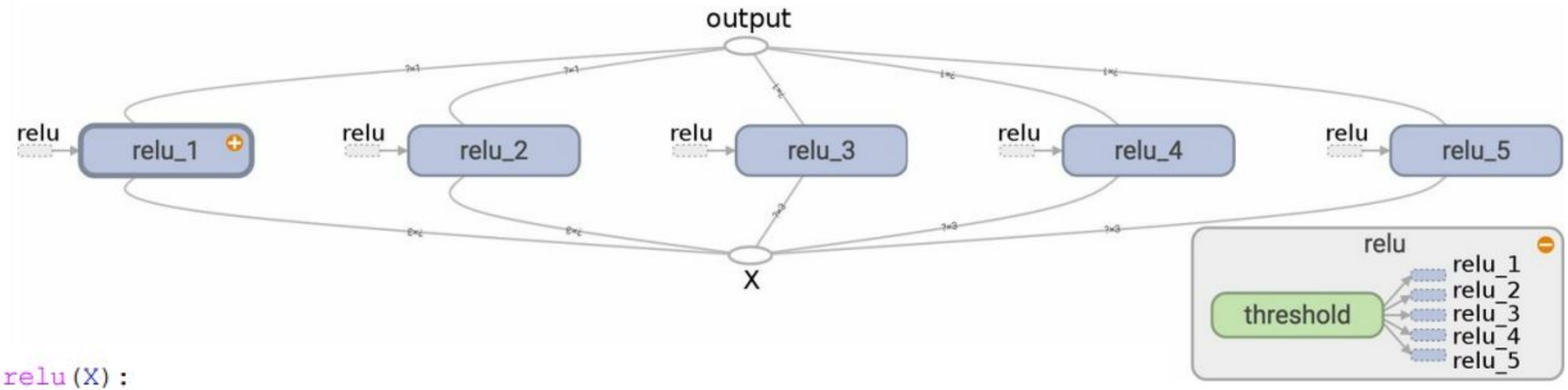
```python
with tf.variable_scope("relu", reuse=True):
    threshold = tf.get_variable("threshold")
```

3

```python
with tf.variable_scope("relu") as scope:
    scope.reuse_variables()
    threshold = tf.get_variable("threshold")
```

Example from “*Hands-on machine learning with Scikit-Learn and TensorFlow*”

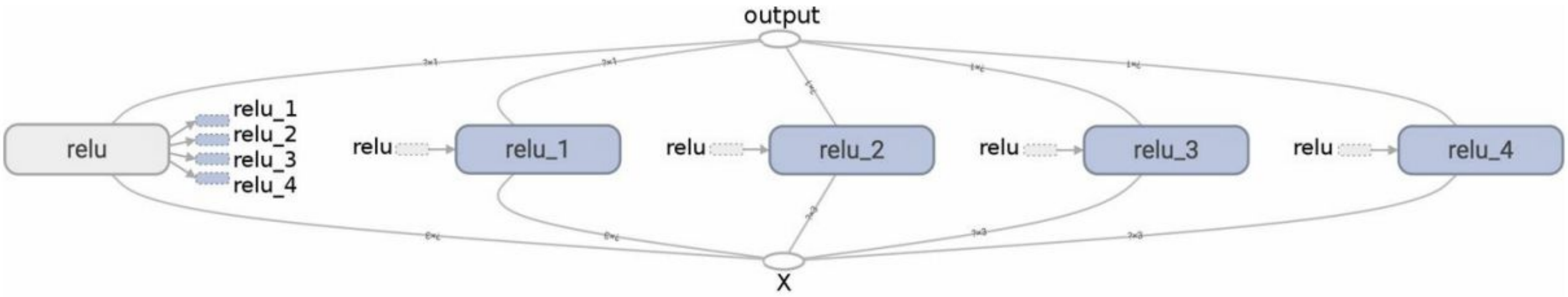
Variable and Name Scopes




```python
def relu(X):
    with tf.variable_scope("relu", reuse=True):
        threshold = tf.get_variable("threshold")  # reuse existing variable
        [...]
        return tf.maximum(z, threshold, name="max")

X = tf.placeholder(tf.float32, shape=(None, n_features), name="X")
with tf.variable_scope("relu"):  # create the variable
    threshold = tf.get_variable("threshold", shape=(),
                                initializer=tf.constant_initializer(0.0))
relus = [relu(X) for relu_index in range(5)]
output = tf.add_n(relus, name="output")
```

Example from *"Hands-on machine learning with Scikit-Learn and TensorFlow"*

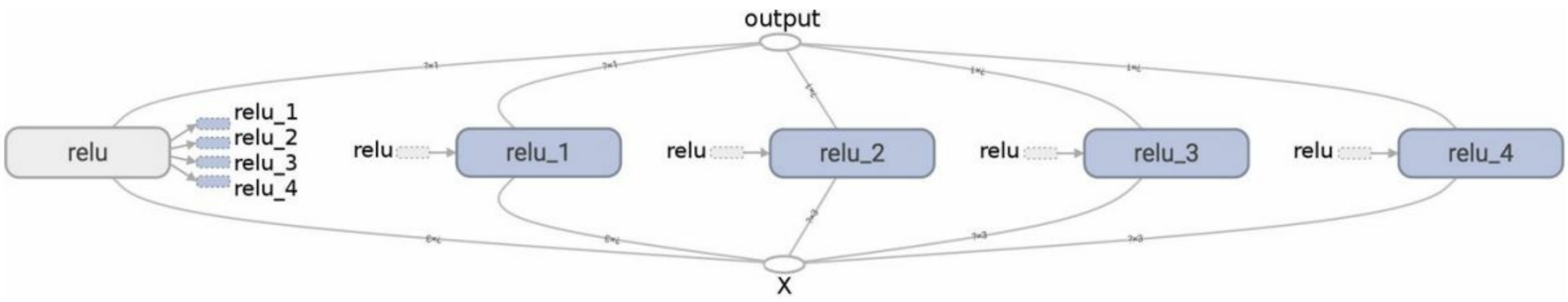
Variable and Name Scopes




```
def relu(X):
    threshold = tf.get_variable("threshold", shape=(),
                                initializer=tf.constant_initializer(0.0))
    [...]
    return tf.maximum(z, threshold, name="max")

X = tf.placeholder(tf.float32, shape=(None, n_features), name="X")
relus = []
for relu_index in range(5):
    with tf.variable_scope("relu", reuse=(relu_index >= 1)) as scope:
        relus.append(relu(X))
output = tf.add_n(relus, name="output")
```

Example from *"Hands-on machine learning with Scikit-Learn and TensorFlow"*

Variable and Name Scopes




```python
def relu(X):
    threshold = tf.get_variable("threshold", shape=(),
                                initializer=tf.constant_initializer(0.0))
    [...]
    return tf.maximum(z, threshold, name="max")

X = tf.placeholder(tf.float32, shape=(None, n_features), name="X")
relus = []
for relu_index in range(5):
    with tf.variable_scope("relu", reuse=(relu_index >= 1)) as scope:
        relus.append(relu(X))
output = tf.add_n(relus, name="output")
```

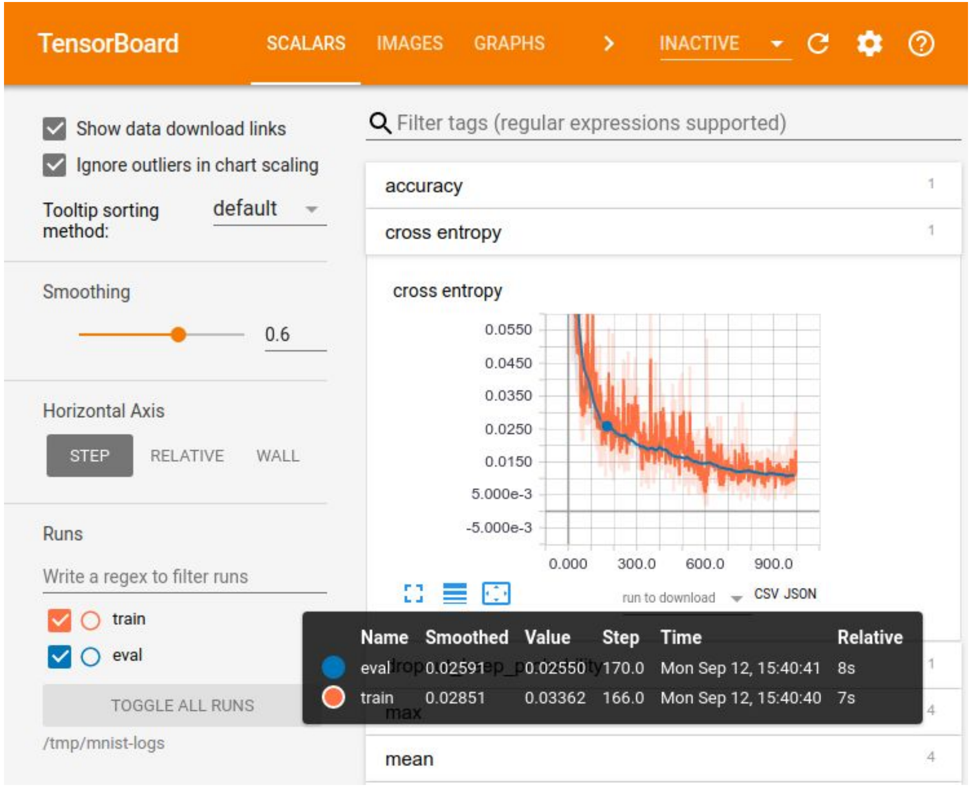
`tf.name_scope` is ignored by `tf.get_variable`

Example from *"Hands-on machine learning with Scikit-Learn and TensorFlow"*

Visualization with TensorBoard

- This is an awesome tool that other frameworks use as well

- It enables browsing the computational graph, monitoring session nodes, and much more



https://www.tensorflow.org/programmers_guide/summaries_and_tensorboard

Visualization with TensorBoard - Logging Stats

1. When assembling the graph:
    - Add summary ops
    - Add merge op
2. In a session:
    - Create a file writer
    - Run the merge op every time you want to log stats
    - Add the returned summary to the file writer
3. Load the log to TensorBoard

Visualization with TensorBoard - Logging Stats

```
import tensorflow as tf

graph = tf.Graph()
with graph.as_default():
  train_inputs = tf.placeholder(tf.int32,shape=[batch_size])
  train_labels = tf.placeholder(tf.int32,shape=[batch_size, 1])

  embeddings = tf.Variable(tf.random_uniform([vocabulary_size, embedding_size],-1.0, 1.0))
  embed = tf.nn.embedding_lookup(embeddings, train_inputs)

  nce_weights = tf.Variable(tf.truncated_normal([vocabulary_size, embedding_size],
                           stddev=1.0 / math.sqrt(embedding_size)))
  nce_biases = tf.Variable(tf.zeros([vocabulary_size]))
  loss = tf.reduce_mean(
        tf.nn.nce_loss(weights=nce_weights, biases=nce_biases, labels=train_labels,
                       inputs=embed, num_sampled=num_sampled, num_classes=vocabulary_size))
  tf.summary.scalar('loss', loss)
  merged = tf.summary.merge_all()

  optimizer = tf.train.GradientDescentOptimizer(1.0).minimize(loss)

  init = tf.global_variables_initializer()
```

Adapted from: https://www.tensorflow.org/tutorials/word2vec

Visualization with TensorBoard - Logging Stats

```
with tf.Session(graph=graph) as session:
  writer = tf.summary.FileWriter(log_dir, session.graph)
  init.run()

  for step in xrange(num_steps):
    batch_inputs, batch_labels = generate_batch(batch_size, num_skips, skip_window)
    feed_dict = {train_inputs: batch_inputs, train_labels: batch_labels}

    _, summary, loss_val = session.run([optimizer,merged, loss], feed_dict=feed_dict)
    writer.add_summary(summary, step)
```

Adapted from: https://www.tensorflow.org/tutorials/word2vec

Visualization with TensorBoard - Logging Stats

```
with tf.Session(graph=graph) as session:
  writer = tf.summary.FileWriter(log_dir, session.graph)
  init.run()

  for step in xrange(num_steps):
    batch_inputs, batch_labels = generate_batch(batch_size, num_skips, skip_window)
    feed_dict = {train_inputs: batch_inputs, train_labels: batch_labels}

    _, summary, loss_val = session.run([optimizer,merged, loss], feed_dict=feed_dict)
    writer.add_summary(summary, step)
```

Practically, it is better to avoid logging stats at every step, since this would slow down training

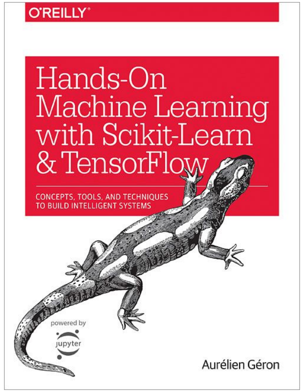
Adapted from: https://www.tensorflow.org/tutorials/word2vec

Plan

✓ - Why TensorFlow
✓ - Basic Code Structure
✓ - Example: Learning Word Embeddings with Skip-gram
✓ - Variable and Name Scopes
✓ - Visualization with TensorBoard

Resources

- Code & Documentation
  - https://www.tensorflow.org/api_docs/
  - https://github.com/tensorflow
- Tutorials / Courses
  - Tensorflow official tutorials
  - CS 20: Tensorflow for Deep Learning Research
- Books
  - Géron, Aurélien. Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems. " O'Reilly Media, Inc.", 2017.



Thank You!

morgeva@mail.tau.ac.il