Eager execution + word2vec

CS 20: TensorFlow for Deep Learning Research
Lecture 4
1/24/2017

- **Assignment 1 is out! (due 1/31)**
- **Gitter chatroom**

Agenda

Eager execution

word2vec

Embedding visualization

Structure your TensorFlow model

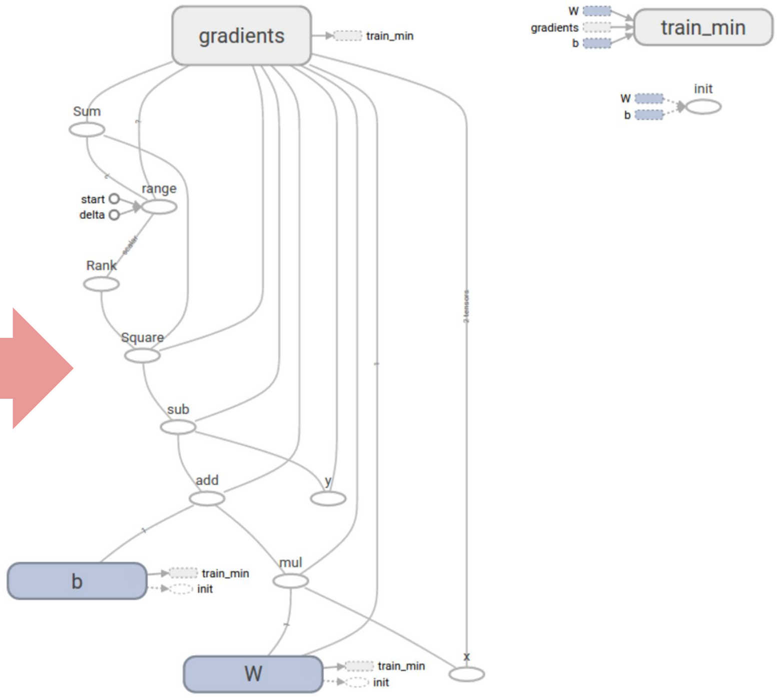
Interactive Coding!

Eager Execution

Presented by Akshay Agrawal
akshayka@{cs.stanford.edu, google.com}

TensorFlow Today: Declarative (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
- constant folding
- inter-op parallelism
- automatic trade-off between compute and memory

Deployable

- the Graph is an intermediate representation for models

Rewritable

- experiment with automatic device placement or quantization

But graphs are also ...

Difficult to debug

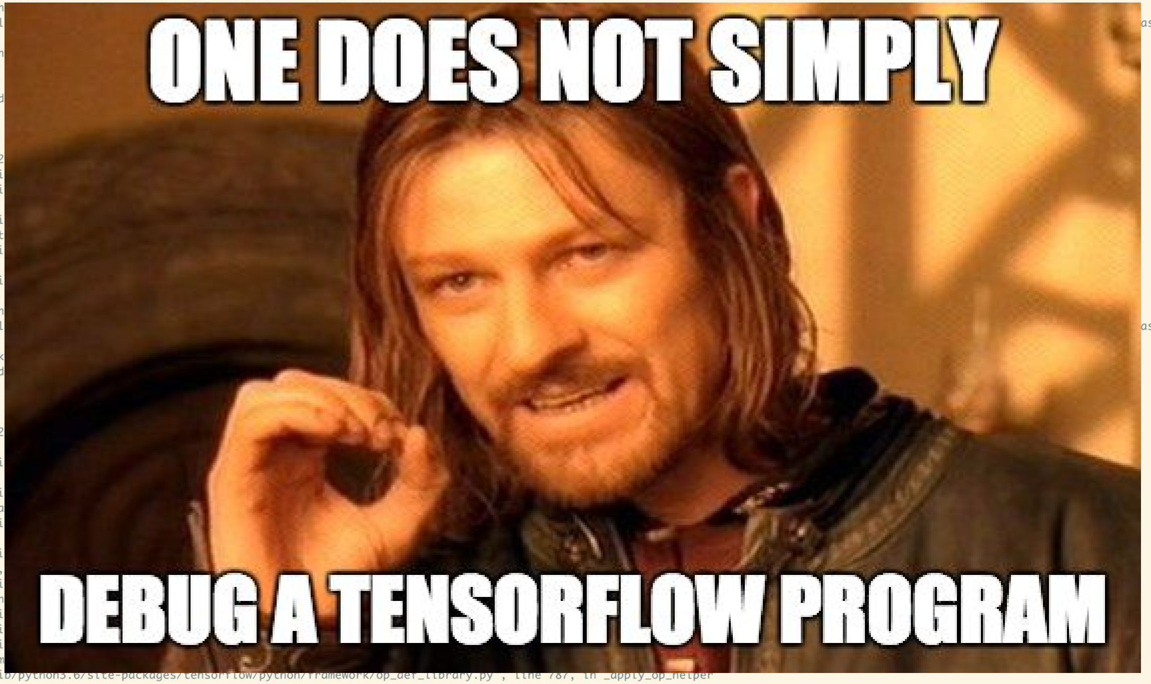
- errors are reported long after graph construction
- execution cannot be debugged with `pdb` or print statements

Un-Pythonic

- writing a TensorFlow program is an exercise in metaprogramming
- control flow (e.g., `tf.while_loop`) differs from Python
- can't easily mix graph construction with custom data structures

```
Traceback (most recent call last):
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/client/session.py", line 1350, in _do_call
    return fn(*args)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/client/session.py", line 1329, in _run_fn
    status, run_metadata)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/framework/errors_impl.py", line 473, in __exit__
    c_api.TF_GetCode(self.status.status))
tensorflow.python.framework.errors_impl.InvalidArgumentError: indices[0] = 3081 is not in [0, 128)
         [[Node: loss/nce_loss/embedding_lookup_1 = Gather[Tindices=DT_INT64, Tparams=DT_FLOAT, _class=["loc:@nce_bias"], validate_indices=true, _device="/job:localhost/replica:0/task:0/device:CPU:0"](nce_bias/read, loss/nce_loss/concat)]]

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "04_word2vec.py", line 102, in <module>
    main()
  File "04_word2vec.py", line 99, in main
    word2vec(dataset)
  File "04_word2vec.py", line 82, in word2vec
    loss_batch, _ = sess.run([loss, optimizer])
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/client/session.py", line 895, in run
    run_metadata_ptr)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/client/session.py", line 1128, in _run
    feed_dict_tensor, options, run_metadata)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/client/session.py", line 1344, in _do_run
    options, run_metadata)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/client/session.py", line 1363, in _do_call
    raise type(e)(node_def, op, message)
tensorflow.python.framework.errors_impl.InvalidArgumentError: indices[0] = 3081 is not in [0, 128)
         [[Node: loss/nce_loss/embedding_lookup_1 = Gather[Tindices=DT_INT64, Tparams=DT_FLOAT, _class=["loc:@nce_bias"], validate_indices=true, _device="/job:localhost/replica:0/task:0/device:CPU:0"](nce_bias/read, loss/nce_loss/concat)]]

Caused by op 'loss/nce_loss/embedding_lookup_1', defined at:
  File "04_word2vec.py", line 102, in <module>
    main()
  File "04_word2vec.py", line 99, in main
    word2vec(dataset)
  File "04_word2vec.py", line 65, in word2vec
    num_classes=VOCAB_SIZE), name='loss')
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/ops/nn_impl.py", line 1212, in nce_loss
    name=name)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/ops/nn_impl.py", line 1046, in _compute_sampled_logits
    biases, all_ids, partition_strategy=partition_strategy)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/ops/embedding_ops.py", line 325, in embedding_lookup
    transform_fn=None)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/ops/embedding_ops.py", line 150, in _embedding_lookup_and_transform
    result = _clip(_gather(params[0], ids, name=name), ids, max_norm)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/ops/embedding_ops.py", line 54, in _gather
    return array_ops.gather(params, ids, name=name)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/ops/array_ops.py", line 2585, in gather
    params, indices, validate_indices=validate_indices, name=name)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/ops/gen_array_ops.py", line 1864, in gather
    validate_indices=validate_indices, name=name)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/framework/op_def_library.py", line 787, in _apply_op_helper
    op_def=op_def)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/framework/ops.py", line 3160, in create_op
    op_def=op_def)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/framework/ops.py", line 1625, in __init__
    self._traceback = self._graph._extract_stack()  # pylint: disable=protected-access

InvalidArgumentError (see above for traceback): indices[0] = 3081 is not in [0, 128)
         [[Node: loss/nce_loss/embedding_lookup_1 = Gather[Tindices=DT_INT64, Tparams=DT_FLOAT, _class=["loc:@nce_bias"], validate_indices=true, _device="/job:localhost/replica:0/task:0/device:CPU:0"](nce_bias/read, loss/nce_loss/concat)]]
```

```
Traceback (most recent call last):
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/client/session.py", line 1350, in _do_call
    return fn(*args)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/client/session.py", line 1329, in _run_fn
    status, run_metadata)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/framework/errors_impl.py", line 473, in __exit__
    c_api.TF_GetCode(self.status.status))
tensorflow.python.framework.errors_impl.In
        [[Node: loss/nce_loss/embedding_l                                                                        as/read, loss/nce_loss/concat)]]

During handling of the above exception, an

Traceback (most recent call last):
  File "04_word2vec.py", line 102, in <mod
    main()
  File "04_word2vec.py", line 99, in main
    word2vec(dataset)
  File "04_word2vec.py", line 82, in word2
    loss_batch, _ = sess.run([loss, optimi
  File "/Users/Akshay/pyenvs/tf-1.50rc1/li
    run_metadata_ptr)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/li
    feed_dict_tensor, options, run_metadat
  File "/Users/Akshay/pyenvs/tf-1.50rc1/li
    options, run_metadata)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/li
    raise type(e)(node_def, op, message)
tensorflow.python.framework.errors_impl.In
        [[Node: loss/nce_loss/embedding_l                                                                        as/read, loss/nce_loss/concat)]]

Caused by op 'loss/nce_loss/embedding_look
  File "04_word2vec.py", line 102, in <mod
    main()
  File "04_word2vec.py", line 99, in main
    word2vec(dataset)
  File "04_word2vec.py", line 65, in word2
    num_classes=VOCAB_SIZE), name='loss')
  File "/Users/Akshay/pyenvs/tf-1.50rc1/li
    name=name)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/li
    biases, all_ids, partition_strategy=pa
  File "/Users/Akshay/pyenvs/tf-1.50rc1/li
    transform_fn=None)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/li
    result = _clip(_gather(params[0], ids,
  File "/Users/Akshay/pyenvs/tf-1.50rc1/li
    return array_ops.gather(params, ids, n
  File "/Users/Akshay/pyenvs/tf-1.50rc1/li
    params, indices, validate_indices=vali
  File "/Users/Akshay/pyenvs/tf-1.50rc1/li
    validate_indices=validate_indices, nam
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/framework/op_def_library.py", line 787, in _apply_op_helper
    op_def=op_def)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/framework/ops.py", line 3160, in create_op
    op_def=op_def)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/framework/ops.py", line 1625, in __init__
    self._traceback = self._graph._extract_stack()  # pylint: disable=protected-access

InvalidArgumentError (see above for traceback): indices[0] = 3081 is not in [0, 128)
        [[Node: loss/nce_loss/embedding_lookup_1 = Gather[Tindices=DT_INT64, Tparams=DT_FLOAT, _class=["loc:@nce_bias"], validate_indices=true, _device="/job:localhost/replica:0/task:0/device:CPU:0"](nce_bias/read, loss/nce_loss/concat)]]
```



What if…

You could execute TensorFlow operations imperatively, *directly from Python*?

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."

- the eager execution user guide

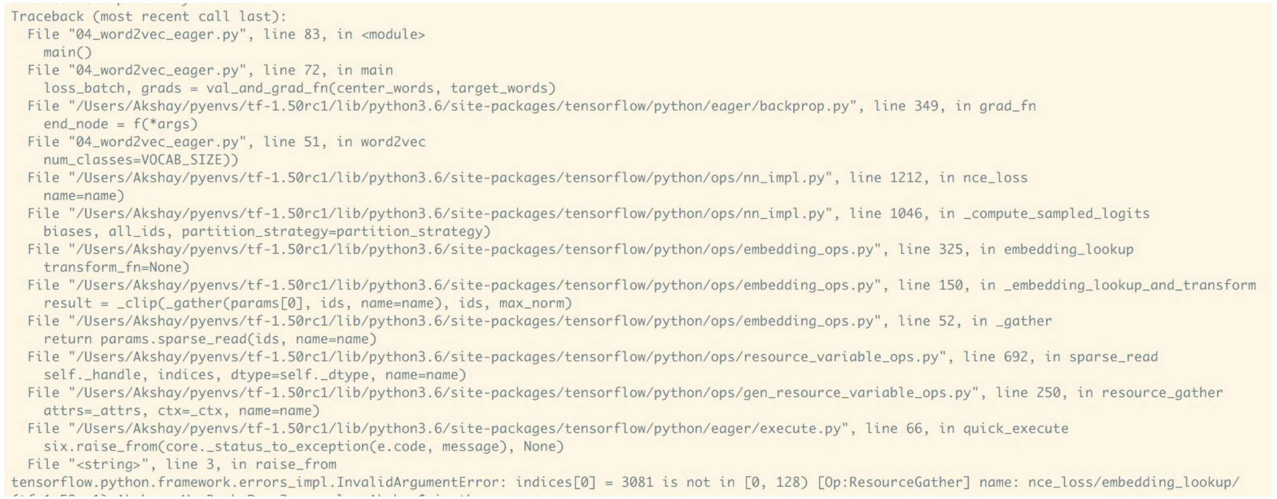
Live Demo

```
$python
import tensorflow # version >= 1.50
import tensorflow.contrib.eager as tfe
tfe.enable_eager_execution()
```

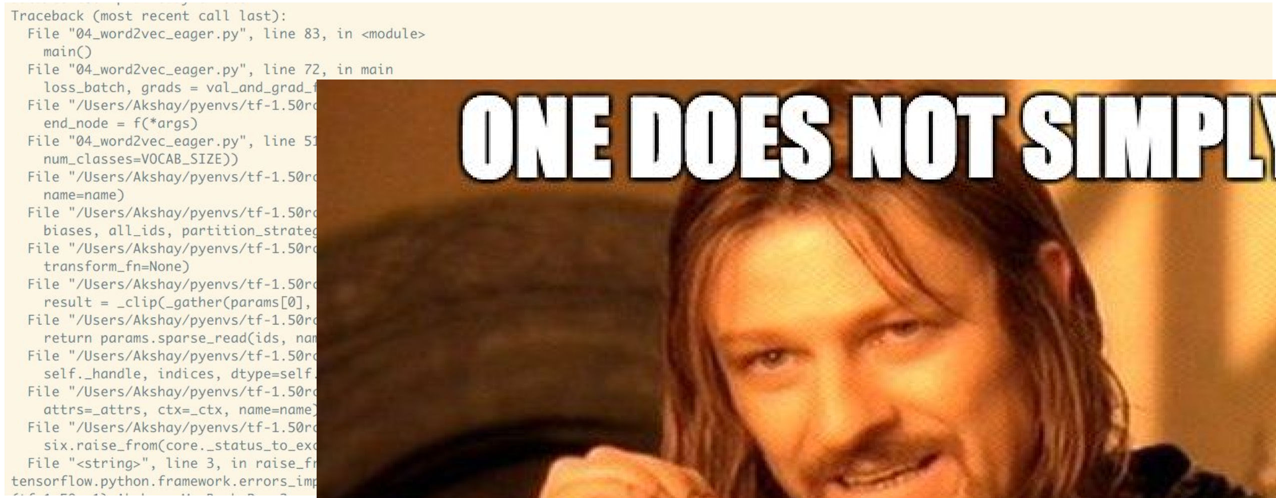
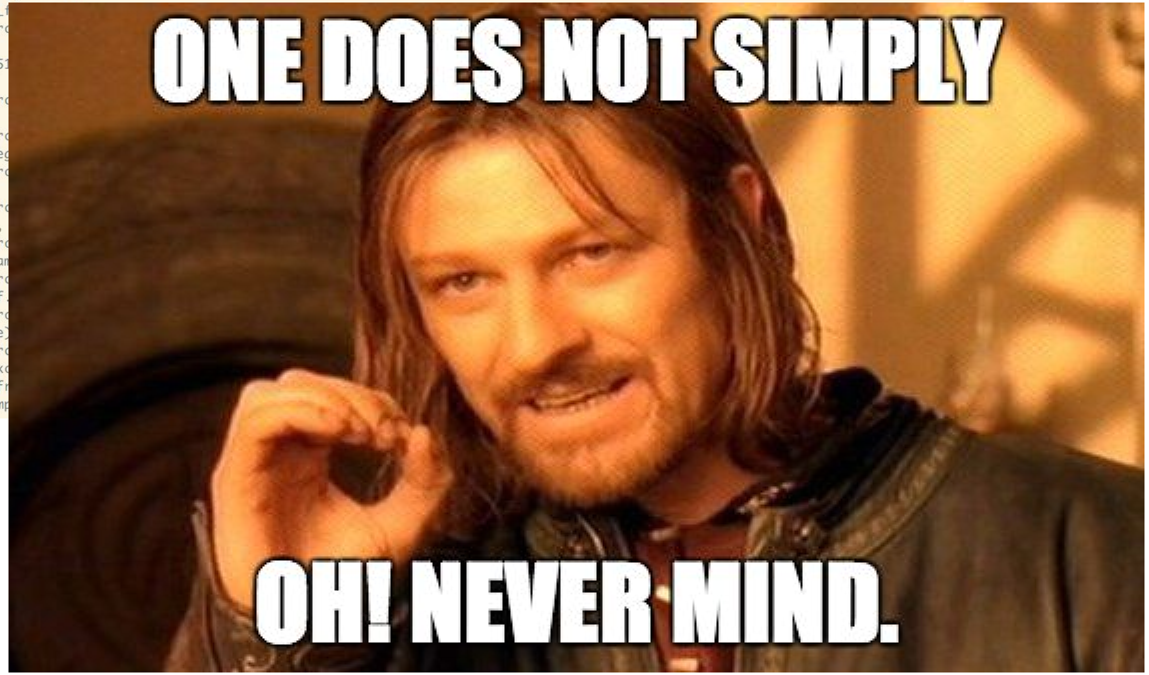
Key Advantages

- Compatible with Python debugging tools
    - `pdb.set_trace()` to your heart's content!
- Provides immediate error reporting
- Permits use of Python data structures
    - e.g., for structured input
- Enables easy, Pythonic control flow
    - `if` statements, `for` loops, recursion, oh my!

```
i = tf.constant(0)
while i < 1000:
  i = tf.add(i, 1)
  print("I could do this all day! %d" % i)
```



```
Traceback (most recent call last):
  File "04_word2vec_eager.py", line 83, in <module>
    main()
  File "04_word2vec_eager.py", line 72, in main
    loss_batch, grads = val_and_grad_fn(center_words, target_words)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/eager/backprop.py", line 349, in grad_fn
    end_node = f(*args)
  File "04_word2vec_eager.py", line 51, in word2vec
    num_classes=VOCAB_SIZE))
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/ops/nn_impl.py", line 1212, in nce_loss
    name=name)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/ops/nn_impl.py", line 1046, in _compute_sampled_logits
    biases, all_ids, partition_strategy=partition_strategy)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/ops/embedding_ops.py", line 325, in embedding_lookup
    transform_fn=None)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/ops/embedding_ops.py", line 150, in _embedding_lookup_and_transform
    result = _clip(_gather(params[0], ids, name=name), ids, max_norm)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/ops/embedding_ops.py", line 52, in _gather
    return params.sparse_read(ids, name=name)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/ops/resource_variable_ops.py", line 692, in sparse_read
    self._handle, indices, dtype=self._dtype, name=name)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/ops/gen_resource_variable_ops.py", line 250, in resource_gather
    attrs=_attrs, ctx=_ctx, name=name)
  File "/Users/Akshay/pyenvs/tf-1.50rc1/lib/python3.6/site-packages/tensorflow/python/eager/execute.py", line 66, in quick_execute
    six.raise_from(core._status_to_exception(e.code, message), None)
  File "<string>", line 3, in raise_from
tensorflow.python.framework.errors_impl.InvalidArgumentError: indices[0] = 3081 is not in [0, 128) [Op:ResourceGather] name: nce_loss/embedding_lookup/
```


```
Traceback (most recent call last):
  File "04_word2vec_eager.py", line 83, in <module>
    main()
  File "04_word2vec_eager.py", line 72, in main
    loss_batch, grads = val_and_grad_f
  File "/Users/Akshay/pyenvs/tf-1.50rc
    end_node = f(*args)
  File "04_word2vec_eager.py", line 51
    num_classes=VOCAB_SIZE))
  File "/Users/Akshay/pyenvs/tf-1.50rc
    name=name)
  File "/Users/Akshay/pyenvs/tf-1.50rc
    biases, all_ids, partition_strateg
  File "/Users/Akshay/pyenvs/tf-1.50rc
    transform_fn=None)
  File "/Users/Akshay/pyenvs/tf-1.50rc
    result = _clip(_gather(params[0],
  File "/Users/Akshay/pyenvs/tf-1.50rc
    return params.sparse_read(ids, nam
  File "/Users/Akshay/pyenvs/tf-1.50rc
    self._handle, indices, dtype=self.
  File "/Users/Akshay/pyenvs/tf-1.50rc
    attrs=_attrs, ctx=_ctx, name=name)
  File "/Users/Akshay/pyenvs/tf-1.50rc
    six.raise_from(core._status_to_exc
  File "<string>", line 3, in raise_fr
tensorflow.python.framework.errors_imp
```


Eager execution simplifies your code

You no longer need to worry about ...

1. placeholders
2. sessions
3. control dependencies
4. "lazy loading"
5. {name, variable, op} scopes

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

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

Becomes this

"Lazy Loading"

```
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

~~"Lazy Loading"~~

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

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

Tensors Act Like NumPy Arrays

```
x = tf.constant([1.0, 2.0, 3.0])


# Tensors are backed by NumPy arrays
assert type(x.numpy()) == np.ndarray
squared = np.square(x) # Tensors are compatible with NumPy functions


# Tensors are iterable!
for i in x:
  print(i)
```

Caveat: use tf.equal to compare Tensors, not ==

Gradients

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

```
def square(x):
  return x ** 2

grad = tfe.gradients_function(square)

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

Differentiate w.r.t. input of square

Gradients

Use **tfe**.`Variable` when eager execution is enabled.

```
x = tfe.Variable(2.0)
def loss(y):
  return (y - x ** 2) ** 2
```

Differentiate w.r.t. variables used to compute `loss`

```
grad = tfe.implicit_gradients(loss)


print(loss(7.))  # tf.Tensor(9., shape=(), dtype=float32)
print(grad(7.))  # [(<tf.Tensor: -24.0, shape=(), dtype=float32>,
                     <tf.Variable 'Variable:0' shape=()
                      dtype=float32, numpy=2.0>)]
```

Gradients

APIs for computing gradients work even when eager execution is not enabled

- `tfe.gradients_function()`
- `tfe.value_and_gradients_function()`
- `tfe.implicit_gradients()`
- `tfe.implicit_value_and_gradients()`

See the user guide for documentation

Huber Regression with Eager Execution

Interactive Coding

`04_regression_eager_starter.py`

It's not *that* different

A Collection of Operations

TensorFlow = Operation Kernels + Execution

- Graph construction: Execute compositions of operations with Sessions
- Eager execution: Execute compositions with Python

A Collection of Operations

Majority of TF API works regardless of whether eager execution is enabled.

- But, when eager execution is enabled ...
    - prefer **`tfe`**`.Variable` under eager execution (compatible with graph construction)
    - manage your own variable storage — variable collections are not supported!
    - use `tf.contrib.summary`
    - use **`tfe`**`.Iterator` to iterate over datasets under eager execution
    - prefer object-oriented layers (e.g., `tf.layers.Dense`)
        - functional layers (e.g., `tf.layers.dense`) only work if wrapped in **`tfe`**`.make_template`
    - prefer **`tfe`**`.py_func` over `tf.py_func`

- See the user guide for details and updates

What if I like graphs?

Graphs are ...

- Optimizable
  - automatic buffer reuse
  - constant folding
  - inter-op parallelism
  - automatic trade-off between compute and memory
- Deployable
  - the Graph is an *intermediate representation* for models
- Rewritable
  - experiment with automatic device placement or quantization

Imperative to declarative and back

- **Write model definition code once**
    - The same code can execute operations in one Python process and construct graphs in another (see user guide/examples)

- **Checkpoints are compatible**
    - Train eagerly, checkpoint, load in a graph, or vice-versa

- **Create graphs while eager execution is enabled**:
    - `tfe.defun:` "Compile" computation into graphs and execute them.

So when should I use eager execution?

Use eager if you're ...

- **a researcher and want a flexible framework**
  - python control flow and data structures enable experimentation
- **developing a new model**
  - immediate error reporting simplifies debugging
- **new to TensorFlow**
  - eager execution lets you explore the TF API in the Python REPL

Status

- Available in version 1.5 of TensorFlow (`import tf.contrib.eager as tfe`)
- Single GPU, ResNet benchmark performance comparable to graphs
- Under active development
  - Overheads on smaller operations are significant
  - Distributed support is in the works
  - Not all TF APIs are eager-compatible

Further reading

Read the user guide to learn about ...

- High-level, Keras-like APIs for constructing models
  - `tfe.Network, tf.layers.Layer`
- Checkpointing variables
- Summaries and tensorboard
- Custom gradients for numerical stability
- Using GPUs

Check out the examples folder for idiomatic code

Links

- Research blog post
- README
- User guide
- Idiomatic model examples
- Survey paper on autodiff for machine learning
- Github issues page
  - Found a bug? Want a feature? Create an issue!
- Feedback: akshayka@google.com

Word Embedding in TensorFlow

How do we represent words in an efficient way?

One-hot Representation

Each word is represented by one vector with a single 1 and the rest is 0

One-hot Representation

Each word is represented by one vector with a single 1 and the rest is 0

Example

Vocab: i, it, california, meh

i = [1 0 0 0]

it = [0 1 0 0]

california = [0 0 1 0]

meh = [0 0 0 1]

Problems with one-hot representation

- Vocabulary can be large

=> massive dimension, inefficient computation

- Can’t represent relationship between words

=> “anxious” and “nervous” are similar but would have completely different representations

Word Embedding

- Distributed representation
- Continuous values
- Low dimension
- Capture the semantic relationships between words

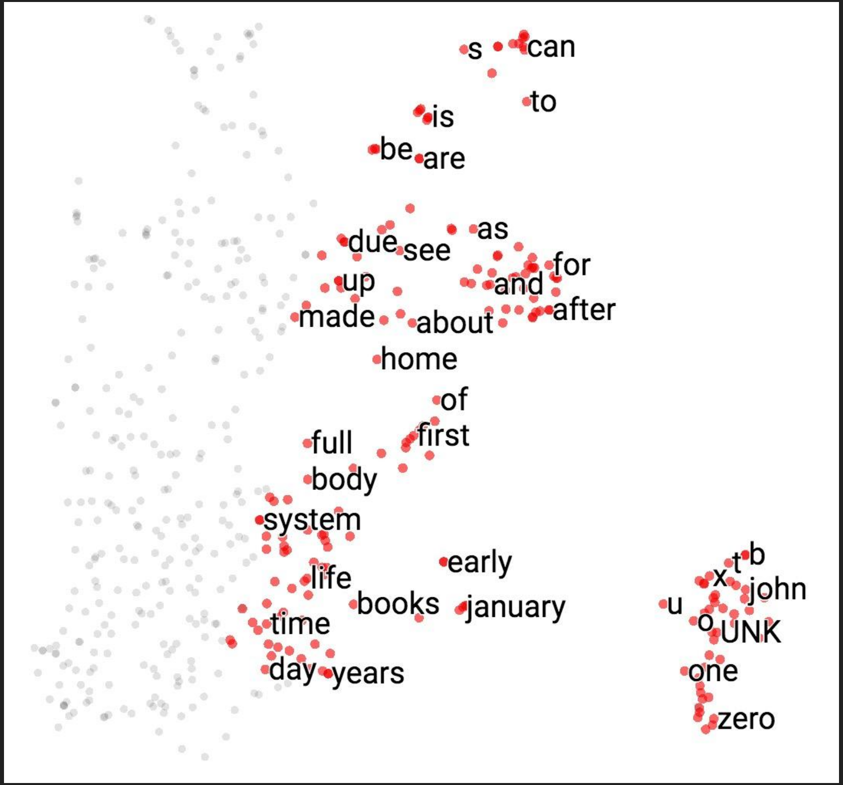
How?

Representing a word by means of its neighbors

“You shall know a word by the company it keeps.”

- Firth, J. R. 1957:11

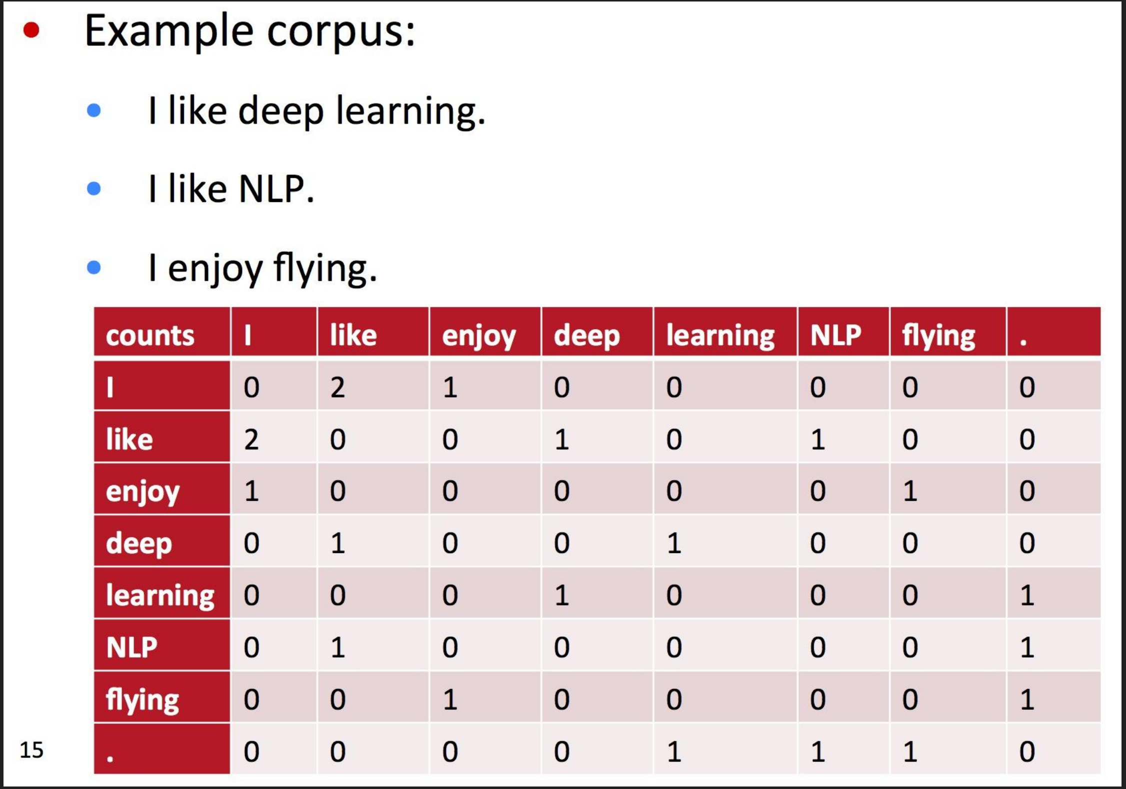
Word Embeddings



Live visualization

Count vs Predict

Counting

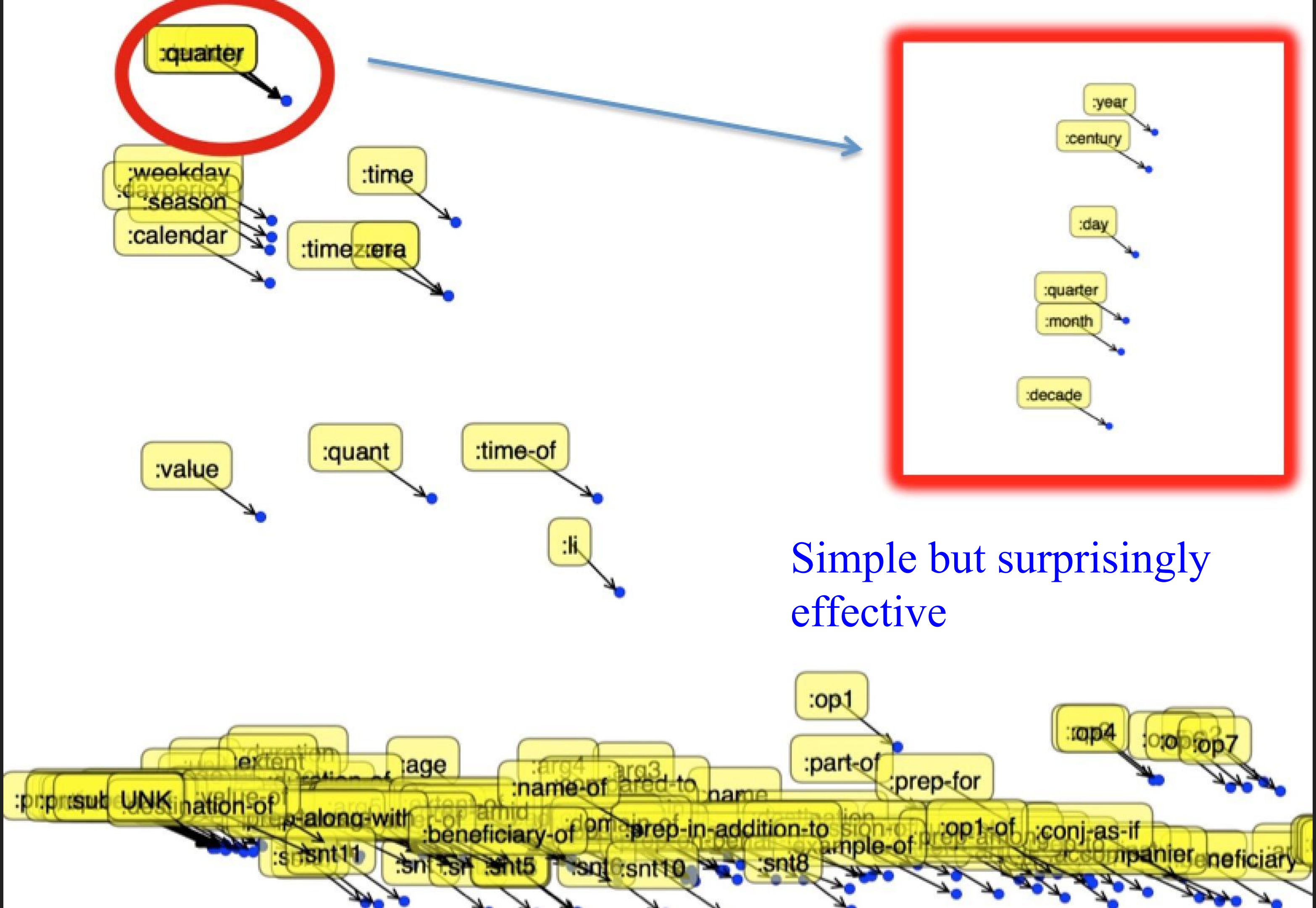


- Example corpus:
  - I like deep learning.
  - I like NLP.
  - I enjoy flying.

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

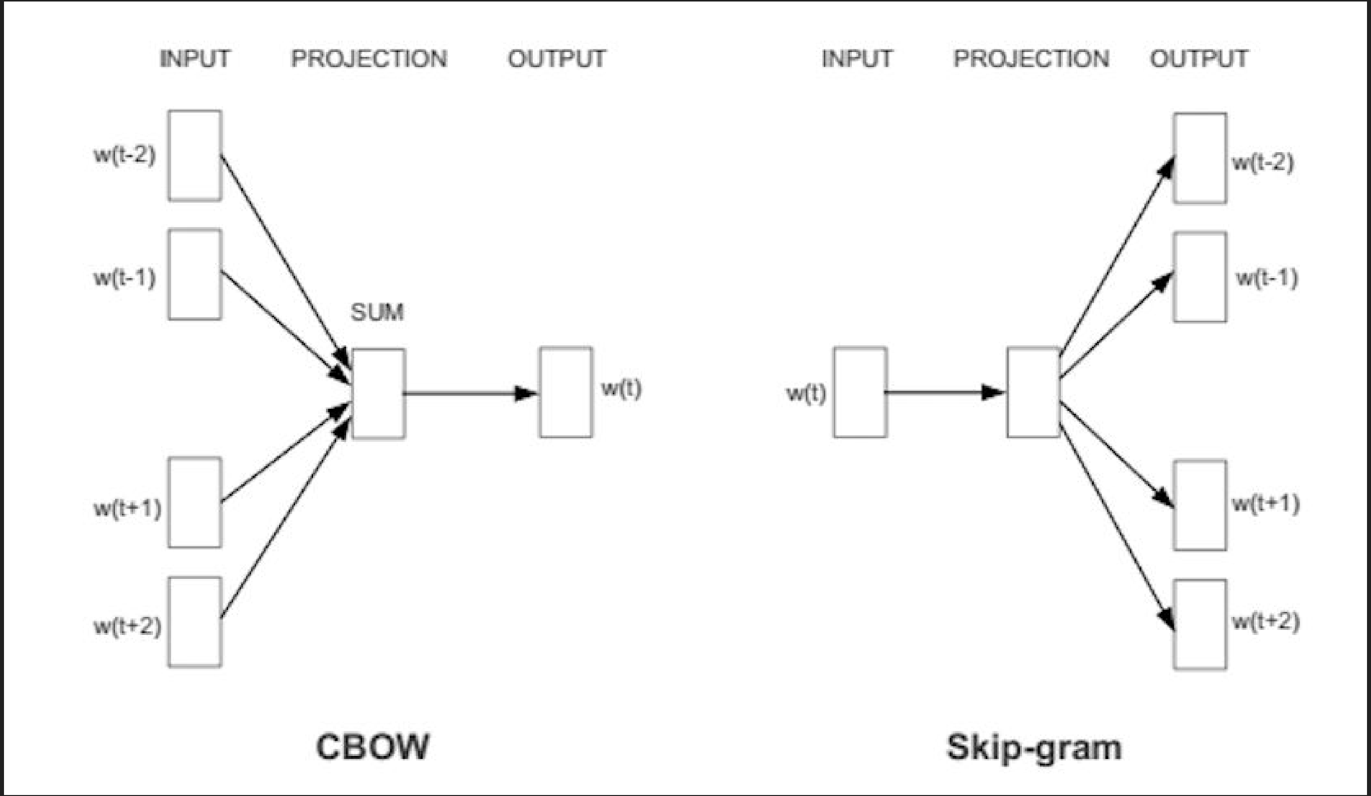
15

Graph by Richard Socher for CS224N



Simple but surprisingly effective

Predicting



Graph by deeplearning4j.org

Implementing word2vec skip-gram

Softmax vs Sample-based Approaches

Softmax

$$p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V} \exp(u_w^T v_c)}$$

Computationally expensive

Handwritten formula from a slide for CS224N

Sample-based Approaches

Negative Sampling

is a simplified version of

Noise Contrastive Estimation

Sample-based Approaches

NCE guarantees approximation to softmax

Negative Sampling doesn't

For more information, see:

Sebastian Rudder's "On word embeddings - Part 2: Approximating the Softmax"

Chris Dyer's "Notes on Noise Contrastive Estimation and Negative Sampling"

Embedding Lookup

$$
\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = \begin{bmatrix} 10 & 12 & 19 \end{bmatrix}
$$

Illustration by Chris McCormick

Embedding Lookup

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = \begin{bmatrix} 10 & 12 & 19 \end{bmatrix}$$

```
tf.nn.embedding_lookup(params, ids, partition_strategy='mod', name=None,
validate_indices=True, max_norm=None)
```

Illustration by Chris McCormick

NCE Loss

```
tf.nn.nce_loss(
    weights,
    biases,
    labels,
    inputs,
    num_sampled,
    num_classes,
    num_true=1,
    sampled_values=None,
    remove_accidental_hits=False,
    partition_strategy='mod',
    name='nce_loss'
)
```

Word2vec in TensorFlow

Interactive Coding

`word2vec_utils.py`

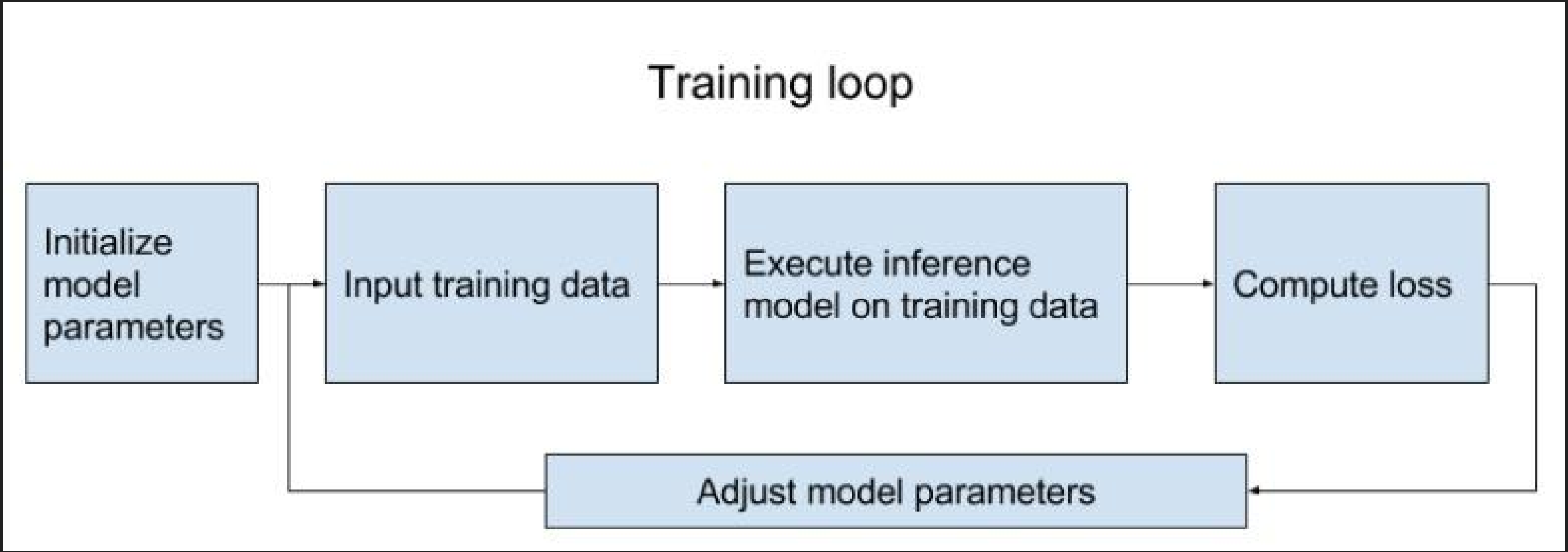
`04_word2vec_eager_starter.py`

Structure your TensorFlow model

Phase 1: Assemble graph

1. Import data (with tf.data or placeholders)
2. Define the weights
3. Define the inference model
4. Define loss function
5. Define optimizer

Phase 2: Compute


Training loop
Initialize model parameters
Input training data
Execute inference model on training data
Compute loss
Adjust model parameters

Graph from the book “TensorFlow for Machine Intelligence”

Need models to be reusable

Reusable models

- Define a class for your model
- Set up your model in a collection (e.g. map)

If you want to really reuse a model (without rebuilding it)

- For big models that take a long time to build, save the graph_def in a file and then load it

Model as a class

Yay, object oriented programming!!

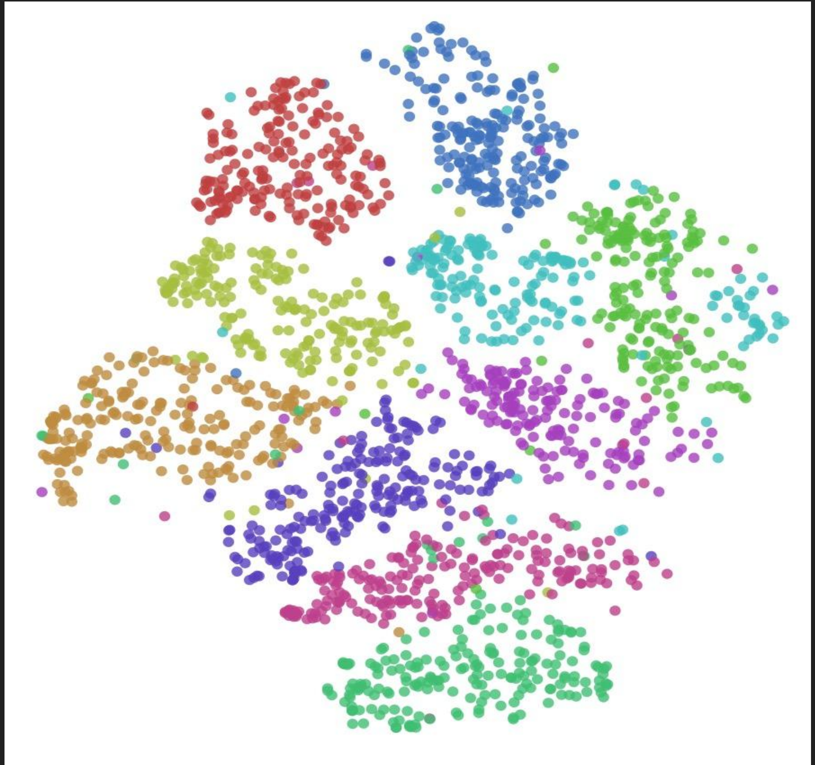
```
class SkipGramModel:
    """ Build the graph for word2vec model """
    def __init__(self, params):
        pass

    def _import_data(self):
        """ Step 1: import data """
        pass

    def _create_embedding(self):
        """ Step 2: define weights. In word2vec, it's actually the weights that we care about """
        pass

    def _create_loss(self):
        """ Step 3 + 4: define the inference + the loss function """
        pass

    def _create_optimizer(self):
        """ Step 5: define optimizer """
        pass
```

Embedding visualization

Interactive Coding

`04_word2vec_visualize.py`

Visualize vector representation of anything



A t-SNE plot of MNIST

Visualization from Chris Olah's blog

Next class

Variable sharing

Manage experiments

Autodiff

Feedback: huyenn@stanford.edu

Thanks!