Introduction to TensorFlow 2.0

Brad Miro - @bradmiro
Google
NVIDIA GTC DC - November 2019
TensorFlow

Deep Learning

Intro to TensorFlow

TensorFlow @ Google

2.0 and Examples

Getting Started
Deep Learning

Doodles courtesy of @dalequark
A scatter plot showing the relationship between weight and height. The equation of the line is $y = (0.06)x + (0.6)$. 

The x-axis represents weight, and the y-axis represents height.
Examples of cats

Examples of dogs
CAT PHOTO PIXEL VALUES

"NEURON"

SUM

"ACTIVATION FUNCTION"

STACK 'EM

(\text{REPEAT } + \text{ AND } \_ )

20\% \text{ CAT}

80\% \text{ DOG}

MAKE PREDICTIONS

INPUT TO NETWORK
IS THIS A CAT or DOG?

CAT  DOG

OUTPUT LAYER

ACTIVATED NEURONS

INPUT LAYER

DEEP NEURAL NETWORK
Use Deep Learning When...

You have *lots* of data (~10k+ examples)
Use Deep Learning When...

You have lots of data (~ 10k+ examples)
The problem is “complex” - speech, vision, natural language
Use Deep Learning When...

You have **lots** of data (≈ 10k+ examples)

The problem is “complex” - speech, vision, natural language

The data is unstructured
Use Deep Learning When...

You have lots of data (~ 10k+ examples)
The problem is “complex” - speech, vision, natural language
The data is unstructured
You need the absolute “best” model
Don’t Use Deep Learning When...

You don’t have a large dataset
Don’t Use Deep Learning When...

You *don’t* have a large dataset

You are performing sufficiently well with traditional ML methods
Don’t Use Deep Learning When...

You don’t have a large dataset

You are performing sufficiently well with traditional ML methods

Your data is structured and you possess the proper domain knowledge
Don’t Use Deep Learning When...

You *don’t* have a large dataset

You are performing sufficiently well with traditional ML methods

Your data is structured and you possess the proper domain knowledge

Your model should be explainable
TensorFlow

Open source deep learning library
Utilities to help you write neural networks
GPU / TPU support
Released by Google in 2015

>2200 Contributors

2.0 released September 2019
41,000,000+
downloads

70,000+
commits

12,000+
pull requests

2,200+
contributors
TensorFlow @ Google
AI-powered data center efficiency

Global localization in Google Maps

Portrait Mode on Google Pixel
TensorFlow 2.0

Easy
- Simplified APIs.
- Focused on Keras and eager execution

Powerful
- Flexibility and performance.
- Power to do cutting edge research and scale to > 1 exaflops

Scalable
- Tested at Google-scale.
- Deploy everywhere
Deploy anywhere

Servers

Edge devices

JavaScript

TensorFlow
  Extended

TensorFlow
  Lite

TensorFlow
  .JS
TF Probability
TF Agents
Tensor2Tensor
TF Ranking
TF Text
TF Federated
TF Privacy
...
You can use TF 2.0 like NumPy

```python
import tensorflow as tf  # Assuming TF 2.0 is installed

a = tf.constant([[1, 2], [3, 4]])
b = tf.matmul(a, a)

print(b)
# tf.Tensor( [[ 7 10] [15 22]], shape=(2, 2), dtype=int32)

print(type(b.numpy()))
# <class 'numpy.ndarray'>
```
Specifics

What’s Gone

- Session.run
- tf.control_dependencies
- tf.global_variables_initializer
- tf.cond, tf.while_loop
- tf.contrib
Specifics

What’s Gone

- Session.run
- tf.control_dependencies
- tf.global_variables_initializer
- tf.cond, tf.while_loop
- tf.contrib

What’s New

- Eager execution by default
- tf.function
- Keras as main high-level api
**Keras and tf.keras**

Fast prototyping, advanced research, and production

**keras.io** = *reference implementation*

```python
import keras
```

**tf.keras** = *TensorFlow’s implementation* (a superset, built-in to TF, no need to install Keras separately)

```python
from tensorflow import keras
```
For Beginners

```python
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation='softmax')
])
```
For Beginners

```python
model = tf.keras.models.Sequential(
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation='softmax')
)
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```
For Beginners

```python
model = tf.keras.models.Sequential([  
    tf.keras.layers.Flatten(),  
    tf.keras.layers.Dense(512, activation='relu'),  
    tf.keras.layers.Dropout(0.2),  
    tf.keras.layers.Dense(10, activation='softmax')
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5)
```
model = tf.keras.models.Sequential([tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation='softmax')])
model.compile(optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
class MyModel(tf.keras.Model):
    def __init__(self, num_classes=10):
        super(MyModel, self).__init__(name='my_model')
        self.dense_1 = layers.Dense(32, activation='relu')
        self.dense_2 = layers.Dense(num_classes, activation='sigmoid')
class MyModel(tf.keras.Model):
    def __init__(self, num_classes=10):
        super(MyModel, self).__init__(name='my_model')
        self.dense_1 = layers.Dense(32, activation='relu')
        self.dense_2 = layers.Dense(num_classes, activation='sigmoid')

def call(self, inputs):
    # Define your forward pass here,
    x = self.dense_1(inputs)
    return self.dense_2(x)
What’s the difference?
Symbolic vs Imperative APIs

**Symbolic** (For Beginners)

Your model is a graph of layers

Any graph you compile will run

*TensorFlow helps you debug* by catching errors at **compile time**
Symbolic vs Imperative APIs

**Symbolic** (For Beginners)
- Your model is a graph of layers
- Any graph you compile will run
  - **TensorFlow helps you debug** by catching errors at **compile time**

**Imperative** (For Experts)
- Your model is Python bytecode
- Complete flexibility and control
- Harder to debug / **harder to maintain**
tf.function
Let’s make this faster

```python
lstm_cell = tf.keras.layers.LSTMCell(10)

def fn(input, state):
    return lstm_cell(input, state)

input = tf.zeros([10, 10]); state = [tf.zeros([10, 10])] * 2
lstm_cell(input, state); fn(input, state) # warm up

# benchmark

timeit.timeit(lambda: lstm_cell(input, state), number=10) # 0.03
```
Let’s make this faster

```python
lstm_cell = tf.keras.layers.LSTMCell(10)

@tf.function
def fn(input, state):
    return lstm_cell(input, state)

input = tf.zeros([10, 10]); state = [tf.zeros([10, 10])] * 2
lstm_cell(input, state); fn(input, state)  # warm up

# benchmark
timeit.timeit(lambda: lstm_cell(input, state), number=10)  # 0.03
```

```python
timeit.timeit(lambda: fn(input, state), number=10)  # 0.004
```
AutoGraph makes this possible

```python
@tf.function
def f(x):
    while tf.reduce_sum(x) > 1:
        x = tf.tanh(x)
    return x

# you never need to run this (unless curious)
print(tf.autograph.to_code(f))
```
def tf__f(x):
    def loop_test(x_1):
        with ag__.function_scope('loop_test'):
            return ag__.gt(tf.reduce_sum(x_1), 1)
    def loop_body(x_1):
        with ag__.function_scope('loop_body'):
            with ag__.utils.control_dependency_on_returns(tf.print(x_1)):
                tf_1, x = ag__.utils.alias_tensors(tf, x_1)
                x = tf_1.tanh(x)
                return x,
        x = ag__.while_stmt(loop_test, loop_body, (x,), (tf,))
    return x
tf.distribution.Strategy
Going big: tf.distribute.Strategy

```python
model = tf.keras.models.Sequential([tf.keras.layers.Dense(64, input_shape=[10]),
                                 tf.keras.layers.Dense(64, activation='relu'),
                                 tf.keras.layers.Dense(10, activation='softmax')])

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```
Going big: Multi-GPU

```python
strategy = tf.distribute.MirroredStrategy()

with strategy.scope():
    model = tf.keras.models.Sequential(
        [
            tf.keras.layers.Dense(64, input_shape=[10]),
            tf.keras.layers.Dense(64, activation='relu'),
            tf.keras.layers.Dense(10, activation='softmax')
        ]
    )

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```
tensorflow_datasets
# Load data

```python
import tensorflow_datasets as tfds
dataset = tfds.load('cats_vs_dogs', as_supervised=True)
mnist_train, mnist_test = dataset['train'], dataset['test']

def scale(image, label):
    image = tf.cast(image, tf.float32)
    image /= 255
    return image, label

mnist_train = mnist_train.map(scale).batch(64)
mnist_test = mnist_test.map(scale).batch(64)
```
TensorFlow Datasets

- audio
  - "nsynth"
- image
  - "cifar10"
  - "diabetic_retinopathy_detection"
  - "imagenet2012"
  - "mnist"
- structured
  - "titanic"
- text
  - "imdb_reviews"
  - "lm1b"
  - "squad"
- translate
  - "wmt_translate_ende"
  - "wmt_translate_enfr"
- video
  - "bair_robot_pushing_small"
  - "moving_mnist"
  - "starcraft_video"

More at tensorflow.org/datasets
Transfer Learning
Transfer Learning

```python
import tensorflow as tf

base_model = tf.keras.applications.SequentialMobileNetV2(
    input_shape=(160, 160, 3),
    include_top=False,
    weights='imagenet')

base_model.trainable = False

model = tf.keras.models.Sequential([base_model,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(1)])

# Compile and fit
```
Text

Embedding

Image

Classification
Feature Vector
Generator

Video

Classification

Publishers

Google
DeepMind

Text embedding

universal-sentence-encoder  By Google
text-embedding  DAN  en
Encoder of greater-than-word length text trained on a variety of data.

nnlm-en-dim128  By Google
text-embedding  GoogleNews  NNLm  en
Token based text embedding trained on English Google News 200B corpus.

elmo  By Google
text-embedding  1BillionWordBenchmark  ELMo  en
Embeddings from a language model trained on the 1 Billion Word Benchmark.

View more text embeddings

Image feature vectors

imagenet/inception_v3/feature_vector  By Google
image-feature-vector  ImageNet (ILSVRC-2012-CLS)  InceptionV3
Upgrading

Migration guides

\texttt{tf.compat.v1} for backwards compatibility

\texttt{tf_upgrade_v2} script
INFO line 4:0: Renamed 'tf.enable_eager_execution' to 'tf.compat.v1.enable_eager_execution'
INFO line 240:16: Renamed 'tf.train.AdamOptimizer' to 'tf.compat.v1.train.AdamOptimizer'
INFO line 332:21: Added keywords to args of function 'tf.multinomial'
INFO line 332:21: Renamed 'tf.multinomial' to 'tf.random.categorical'
INFO line 375:12: Renamed 'tf.train.AdamOptimizer' to 'tf.compat.v1.train.AdamOptimizer'
INFO line 392:21: tf.losses.sparse_softmax_cross_entropy requires manual check. tf.losses
INFO line 392:21: Renamed 'tf.losses.sparse_softmax_cross_entropy' to 'tf.compat.v1.losses'

TensorFlow 2.0 Upgrade Script
---------------------------------------
Converted 1 files
Detected 0 issues that require attention
---------------------------------------

Make sure to read the detailed log 'report.txt'
Getting Started
TensorFlow 2.0

pip install tensorflow
An end-to-end open source machine learning platform

Get started with TensorFlow

The core open-source library to help you develop and train ML models. Get started quickly by running Colab notebooks directly in your browser.
New Courses

Introduction to TensorFlow for AI, ML and DL

coursera.org/learn/introduction-tensorflow

Intro to TensorFlow for Deep Learning

udacity.com/tensorflow
Go build.

`pip install tensorflow`

tensorflow.org
tf.thanks!

Brad Miro - @bradmiro
tensorflow.org
NVIDIA GTC DC - November 2019