TensorFlow

Welcome to TensorFlow 2.0

Josh Gordon (twitter.com/random_forests)

Machine Learning Tokyo

TensorFlow

An open source Deep Learning library

- Released by Google in 2015
- >1800 contributors worldwide

TensorFlow 2.0

- **Easier to use**
- Code styles for beginners and experts
- Alpha released in March, 2019

Topics

For beginners and experts

- Keras Sequential
- Keras Subclassing
- Built-in vs custom training loops

Under the hood

- AutoGraph and tf.function
- TF2 vs TF1

Beyond “Hello World”

- Tutorials for Deep Dream, GANs, Machine Translation

Learning more

- Book recommendations

What exactly is TensorFlow?

- And, what problems are Deep Learning libraries trying to solve?

TensorFlow

Why is Python popular for scientific computing?

Ballpark benchmarks

About how much slower is Python than C?

Ballpark benchmarks

About how much slower is Python than C?

- Multiplying matrices: +/- 100X
- 6 seconds vs. 10 minutes
- Running vs. flying (6 MPH and 600 MPH)

Ballpark benchmarks

About how much slower is Python than C?

- Multiplying matrices: +/- 100X
- 6 seconds vs. 10 minutes
- Running vs. flying (6 MPH and 600 MPH)

Python is a great choice for scientific computing

- Why?

Ballpark benchmarks

About how much slower is Python than C?

- Multiplying matrices: +/- 100X
- 6 seconds vs. 10 minutes
- Running vs. flying (6 MPH and 600 MPH)

Python is a great choice for scientific computing

- Why?

NumPy

- C performance, Python ease of use

TensorFlow is basically

NumPy

+ GPU / TPU support
+ AutoDiff
+ Utilities to help you write neural networks (layers, optimizers)

TensorFlow

- A C++ engine to accelerate code written in Python.
- **Bonus**: your program is compiled to a graph that can run on devices **without a Python interpreter** **(phones, web browsers)**

You can use TF 2.0 like NumPy

```
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'>
```

Exercise 1

Goals

- Install TensorFlow 2.0
- Introduce Colab
- Introduce gradient descent

Visit

- bit.ly/tf-ws1

TensorFlow

For beginners and experts

For beginners

```
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(512, activation='relu'),
  tf.keras.layers.Dense(128, activation='relu'),
  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)
```

TF 1.x

```
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)
```

TF 2.0

```
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)
```

Keras and tf.keras

In my view, the **clearest Deep Learning library** that exists today.

- For fast prototyping, advanced research, and production.

keras.io = reference implementation

- `import keras`

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

- `from tensorflow import keras`

For experts

```
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)
```

TensorFlow

What’s the difference?

Symbolic vs Imperative APIs

Symbolic (Keras Sequential)

- 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 (Keras Sequential)

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

Imperative (Keras Subclassing)

- Your model is Python bytecode
- Complete flexibility and control
- Harder to debug / **harder to maintain**

Use a built-in training loop...

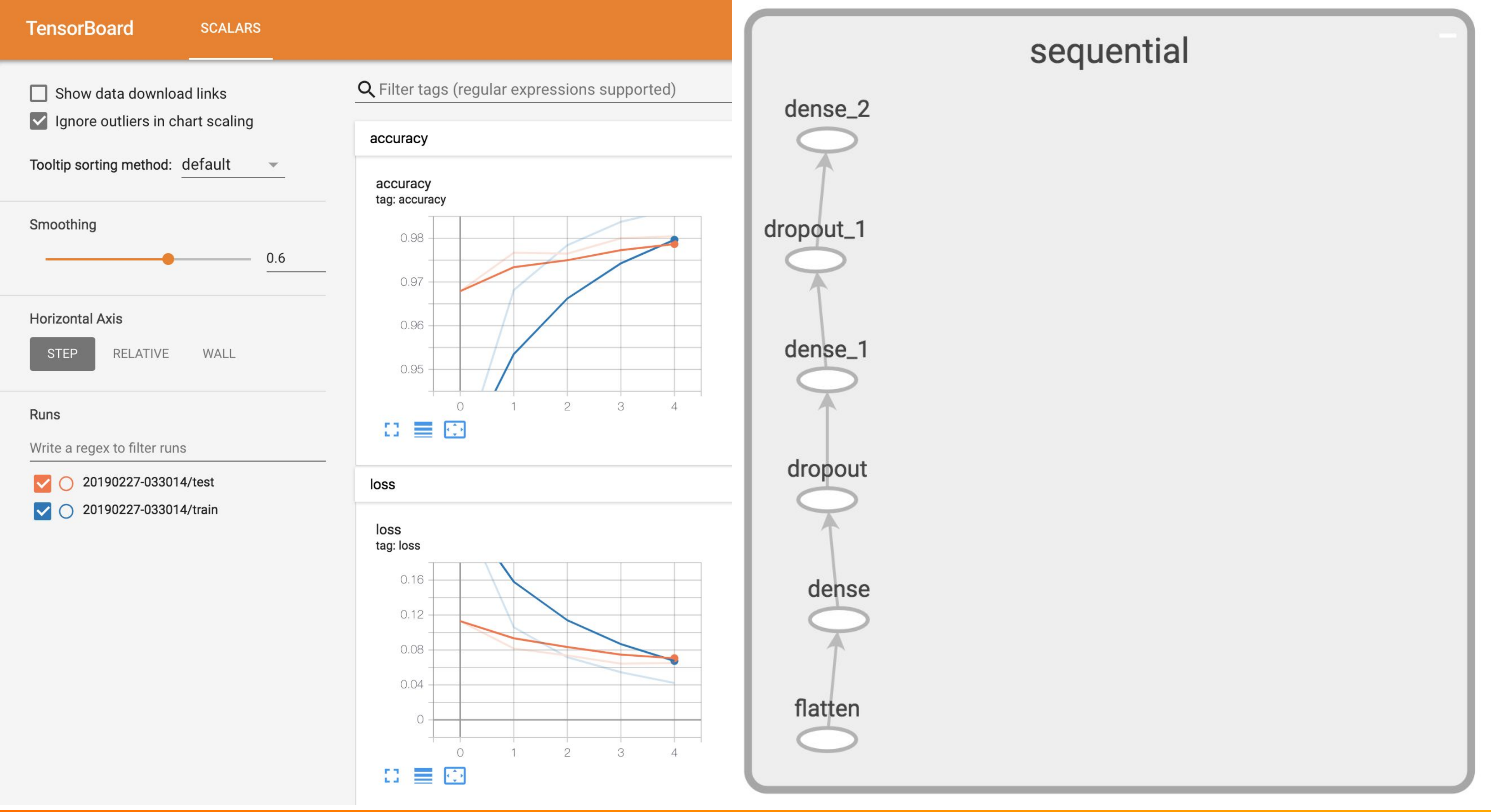
```
model.fit(x_train, y_train, epochs=5)
```

Or define your own

```
model = MyModel()

with tf.GradientTape() as tape:
  logits = model(images)
  loss_value = loss(logits, labels)

grads = tape.gradient(loss_value, model.trainable_variables)
optimizer.apply_gradients(zip(grads, model.trainable_variables))
```

TensorBoard

```
tb_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir)

model.fit(
    x_train, y_train, epochs=5,
    validation_data=[x_test, y_test],
    callbacks=[tb_callback])
```



TensorBoard
SCALARS
Show data download links
Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing
0.6
Horizontal Axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
20190227-033014/test
20190227-033014/train
Filter tags (regular expressions supported)
accuracy
accuracy
tag: accuracy
0.98
0.97
0.96
0.95
0
1
2
3
4
loss
loss
tag: loss
0.16
0.12
0.08
0.04
0
0
1
2
3
4
sequential
dense_2
dropout_1
dense_1
dropout
dense
flatten

Beyond Hello World

A few of my favorites

- Machine Translation
- Image Captioning (incidentally, the decoder is similar!)
- DCGan and Pix2Pix

The docs are code

Tutorials on tf.org/alpha are

- Backed by a Jupyter Notebook
- Can be run directly in Colab

They automatically

- Install the right TensorFlow version
- Download a dataset
- Train a model
- Show you the result

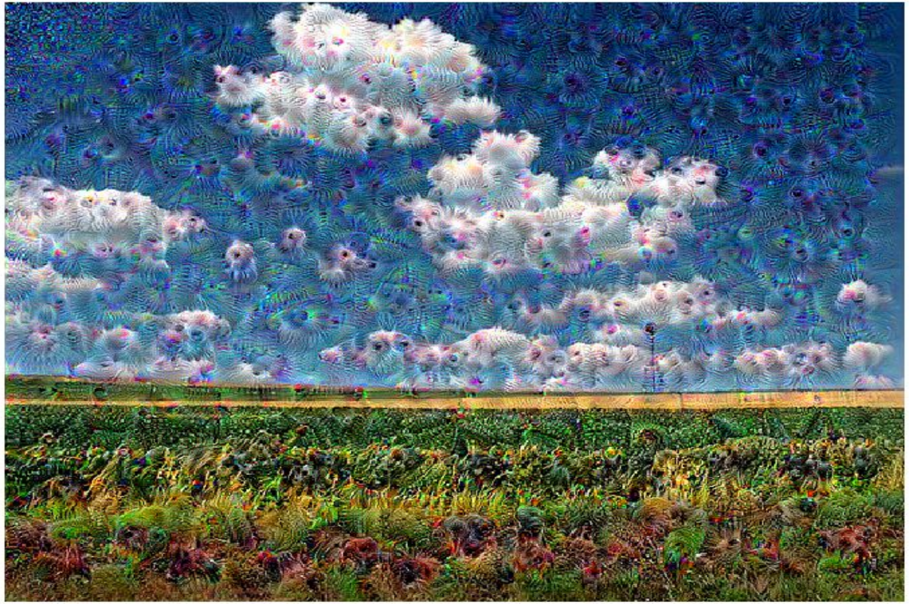
TensorFlow > Learn > TensorFlow Core > TF 2.0 Alpha

Image Captioning with Attention

Run in Google Colab | View source on GitHub

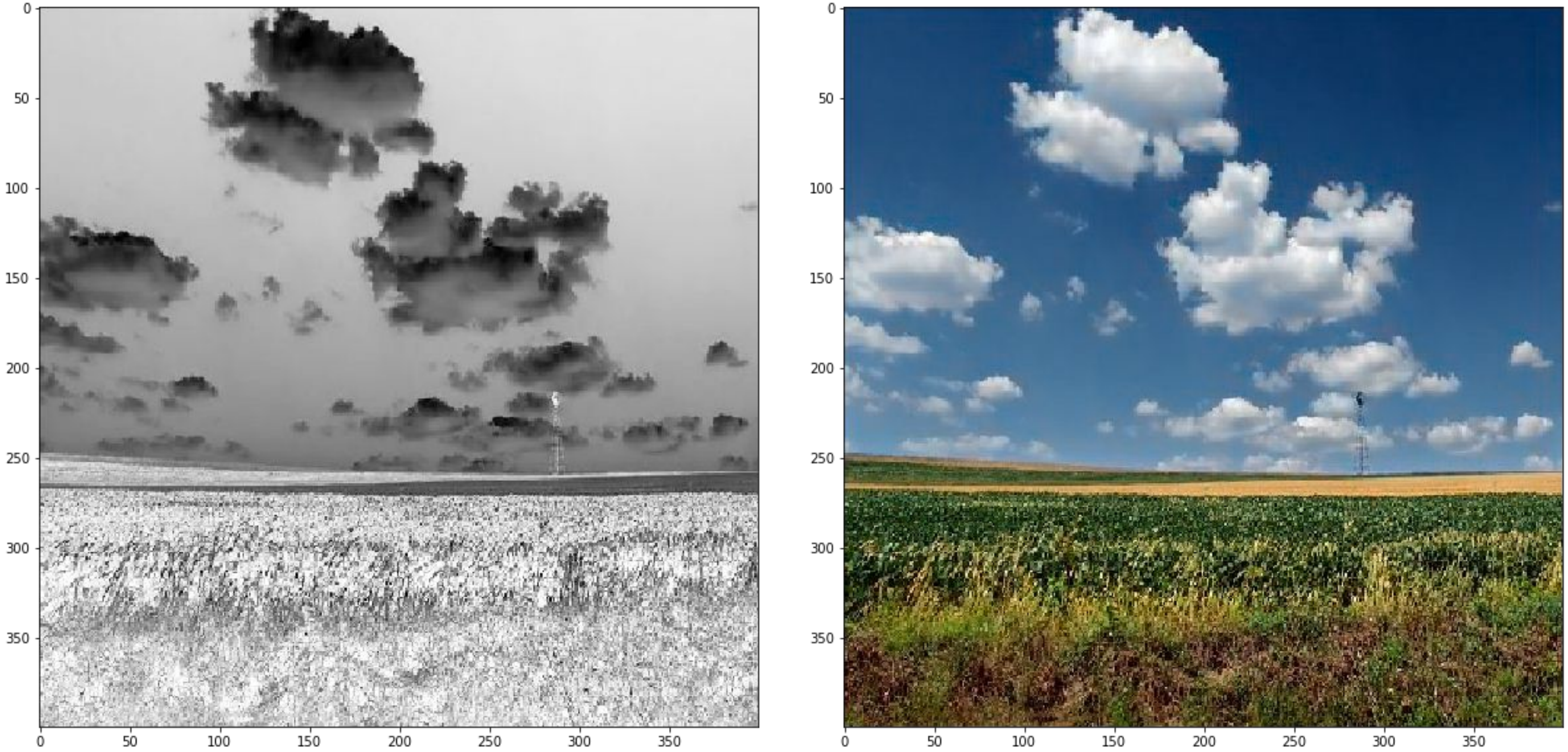
Given an image like the below, our goal is to generate a caption, such as "a surfer riding on a wave".

tensorflow.org/alpha/tutorials/text/image_captioning



https://github.com/random-forests/applied-dl/blob/master/examples/9-deep-dream-minimal.ipynb

Code walkthrough



https://github.com/random-forests/applied-dl/blob/master/examples/9-image-colorization.ipynb

Is anyone bilingual? Trilingual?

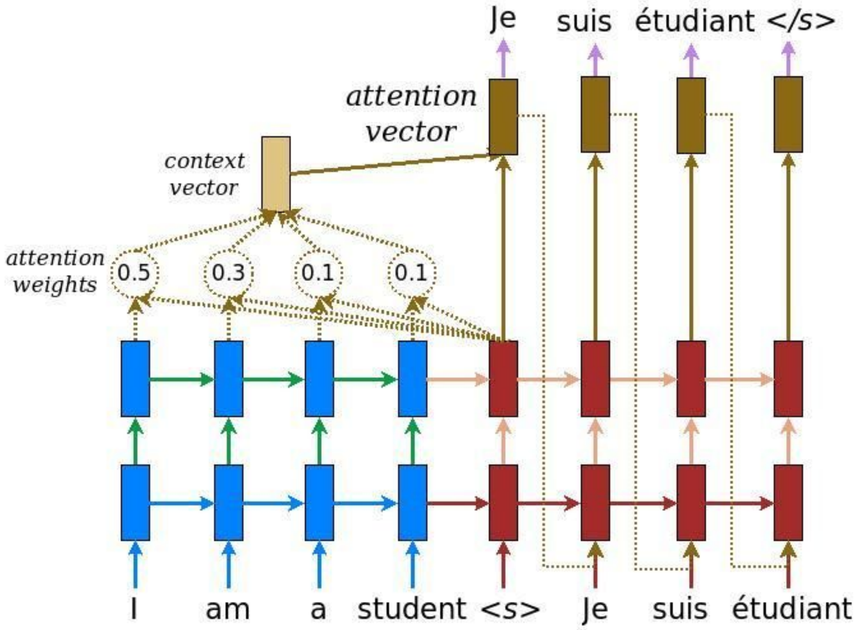
When translating, do you...

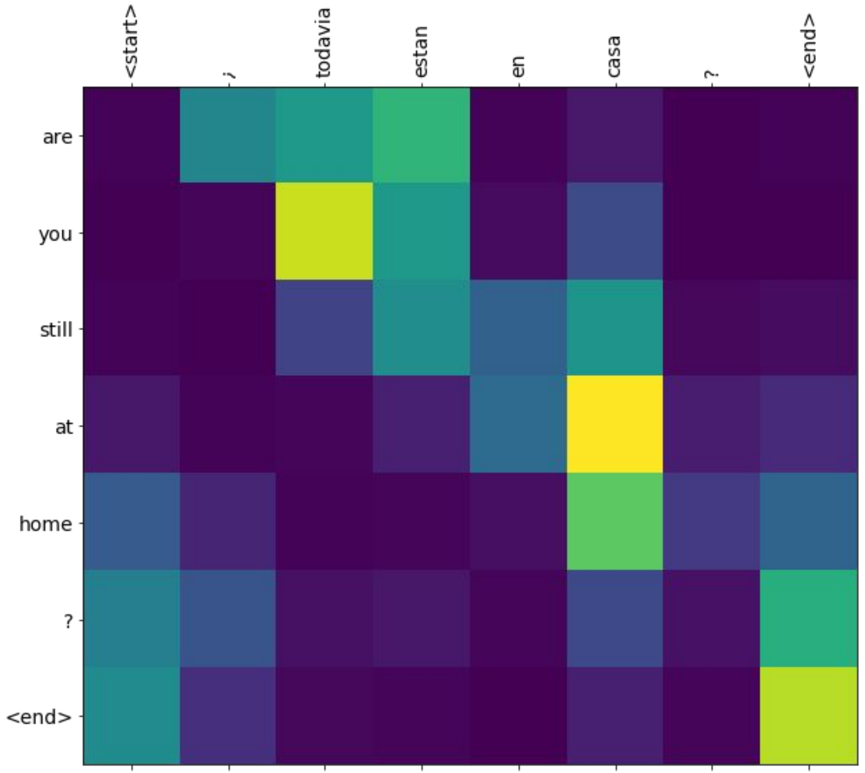
- Go directly from source -> target
- Or, go from source -> **intermediate representation** -> target.

Machine translation tutorials

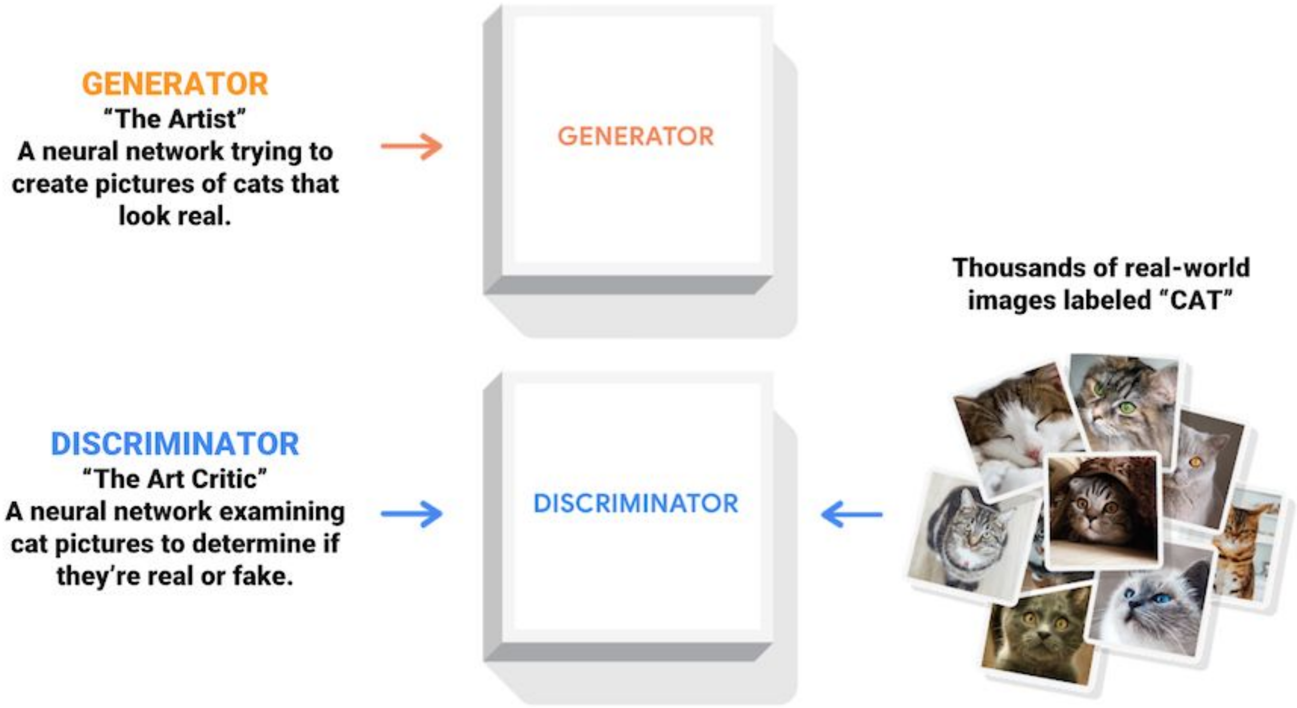
- Hello world (seq2seq), trains in about a minute.
- Neural Machine Translation with Attention
- Transformer

P.S., isn't 2019 cool? It's **amazing** this is possible.

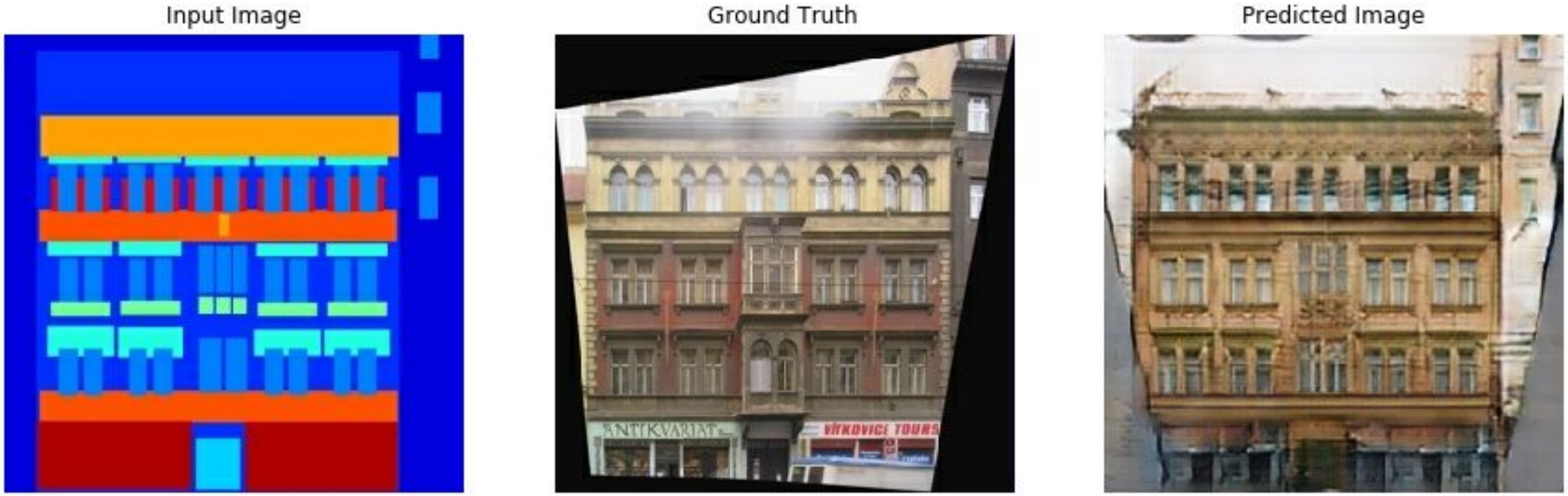




https://www.tensorflow.org/alpha/tutorials/sequences/nmt_with_attention



https://www.tensorflow.org/alpha/tutorials/generative/dcgan



https://www.tensorflow.org/alpha/tutorials/generative/pix2pix

```
Prediction Caption: the person is riding a surfboard in the ocean <end>
```

https://www.tensorflow.org/alpha/tutorials/sequences/image_captioning

Under the hood

Let’s make this faster

```
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

```
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
timeit.timeit(lambda: fn(input, state), number=10) # 0.004
```

AutoGraph makes this possible

```
@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))
```

Generated code

```python
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
```

Going big: tf.distribute.Strategy

```
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

```
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'])
```

What’s different between TF1 and TF2?

Removed

- session.run
- tf.control_dependencies
- tf.global_variables_initializer
- tf.cond, tf.while_loop

Added

- tf.function, AutoGraph

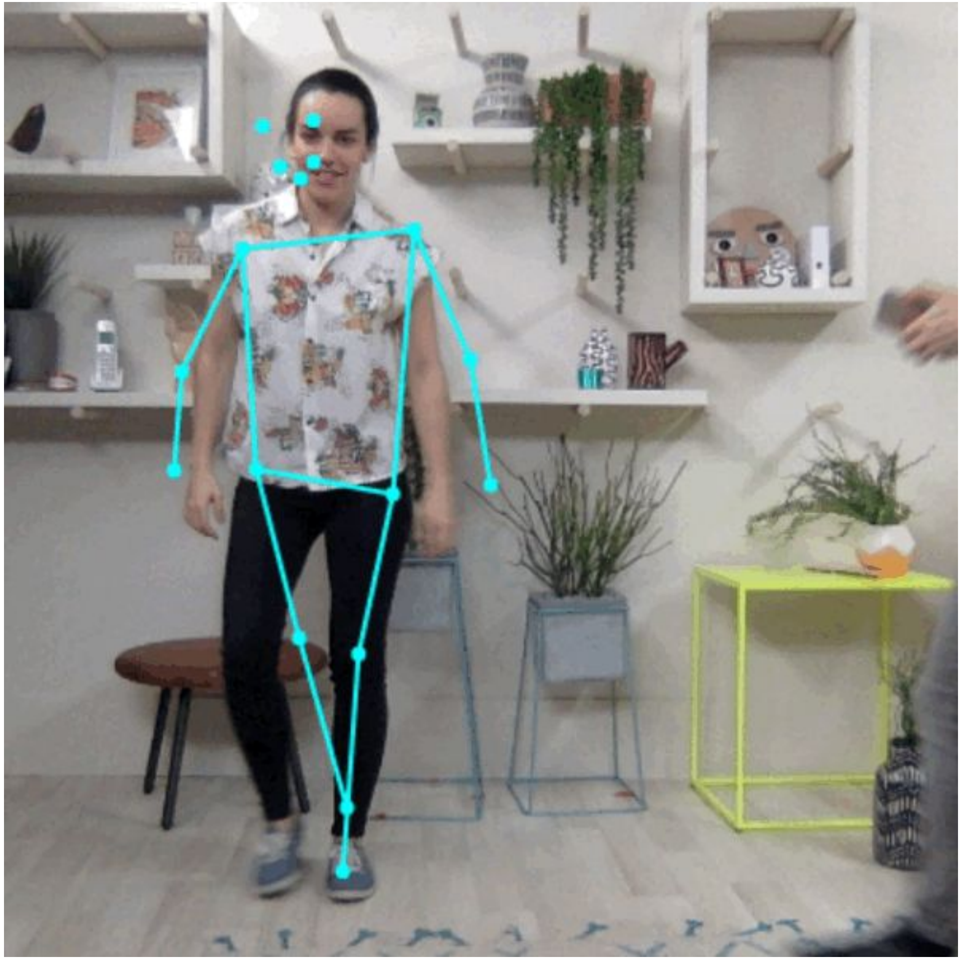
TensorFlow

TensorFlow.js

Demo #1

PoseNet

PoseNet



bit.ly/pose-net

Demo #2

BodyPix

BodyPix

bit.ly/body-pix

TensorFlow

Learning more

Learn more

Tutorials and guides

- tensorflow.org/alpha

Books

- Deep Learning with Python
- Hands-On Machine Learning with Scikit-Learn and TensorFlow (version 2.0 is almost ready)

Courses

- Intro to Deep Learning (MIT)
- Convolutional Neural Networks for Visual Recognition (Stanford)

TensorFlow

tf.thanks!

Josh Gordon (twitter.com/random_forests)