TensorFlow

Hands-on Introduction to TensorFlow 2.0

Josh Gordon (@random_forests)
Robert Crowe (@robert_crowe)

At ICSE Montreal

Getting started

You will need

- A laptop with an internet connection.
- There's nothing to install in advance.

Agenda

- **Beginner** examples in the first half.
- **Advanced** examples in the second.

Goal

- Write and run a few basic examples, know where to go to learn more.

Agenda

Beginner examples

- Installing TensorFlow 2.0 and using Colab
- Linear regression
- MNIST (for beginners); MNIST (for researchers)

Intermediate examples

- Serving a model with a REST API
- Serving a model in a browser

Advanced examples (as time remains)

- Deep Dream, Neural machine translation, Image Colorization

TensorFlow

An open source Deep Learning library

- Released by Google in 2015
- >1800 contributors worldwide

TensorFlow 2.0

- **Easier to use**
- For beginners and experts
- In alpha, final version coming in a few months

TensorFlow is basically

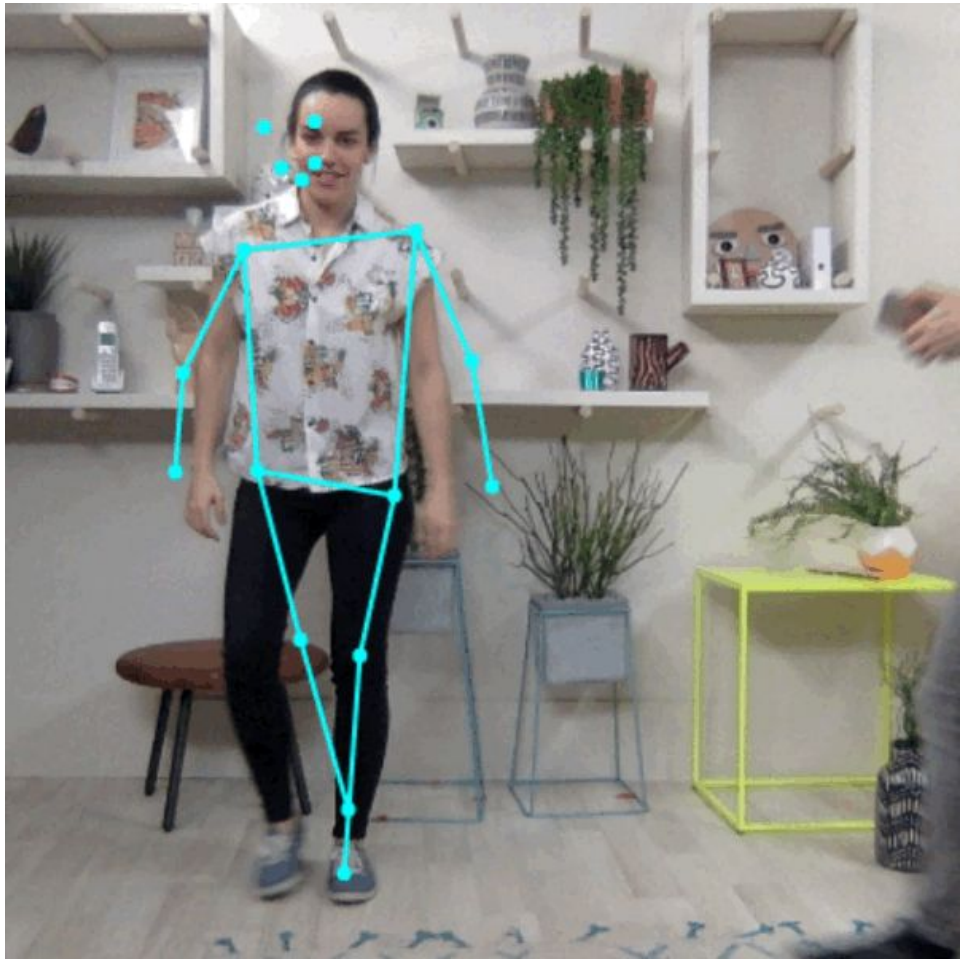
NumPy

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

TensorFlow

- A C++ engine to accelerate code written in Python.
- Your program is represented as a **graphs** that can run on devices **without a Python interpreter** (**phones**, **browsers, etc**)

PoseNet



bit.ly/pose-net

The person is riding a surfboard in the waves.

Topics you'll learn about

Building models

- Sequential models
- Subclassed models

Under the hood

- AutoGraph and tf.function
- TF2 vs TF1

Training models

- Built-in vs custom training loops

Learning more

- Book recommendations

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'>
```

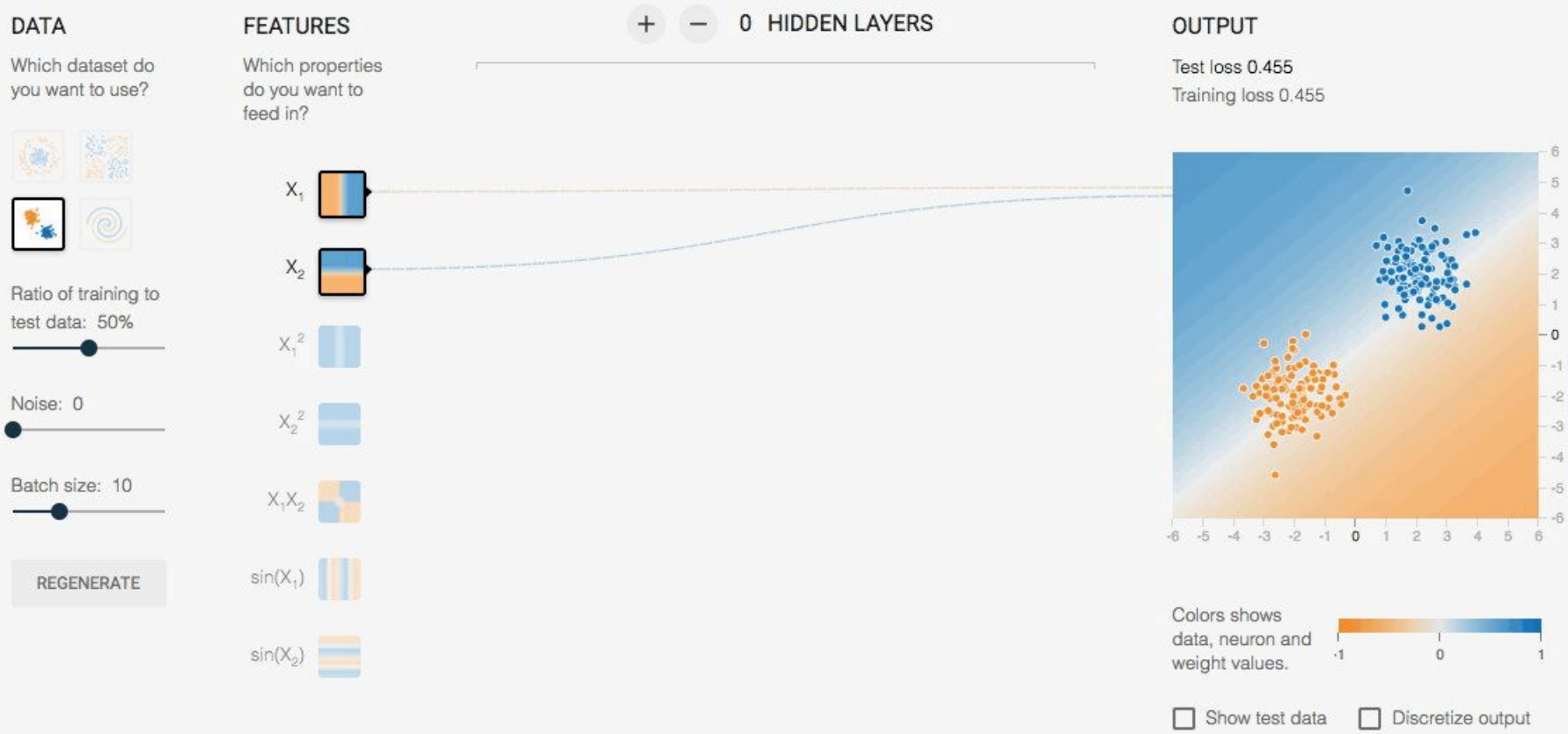
What's Deep Learning?

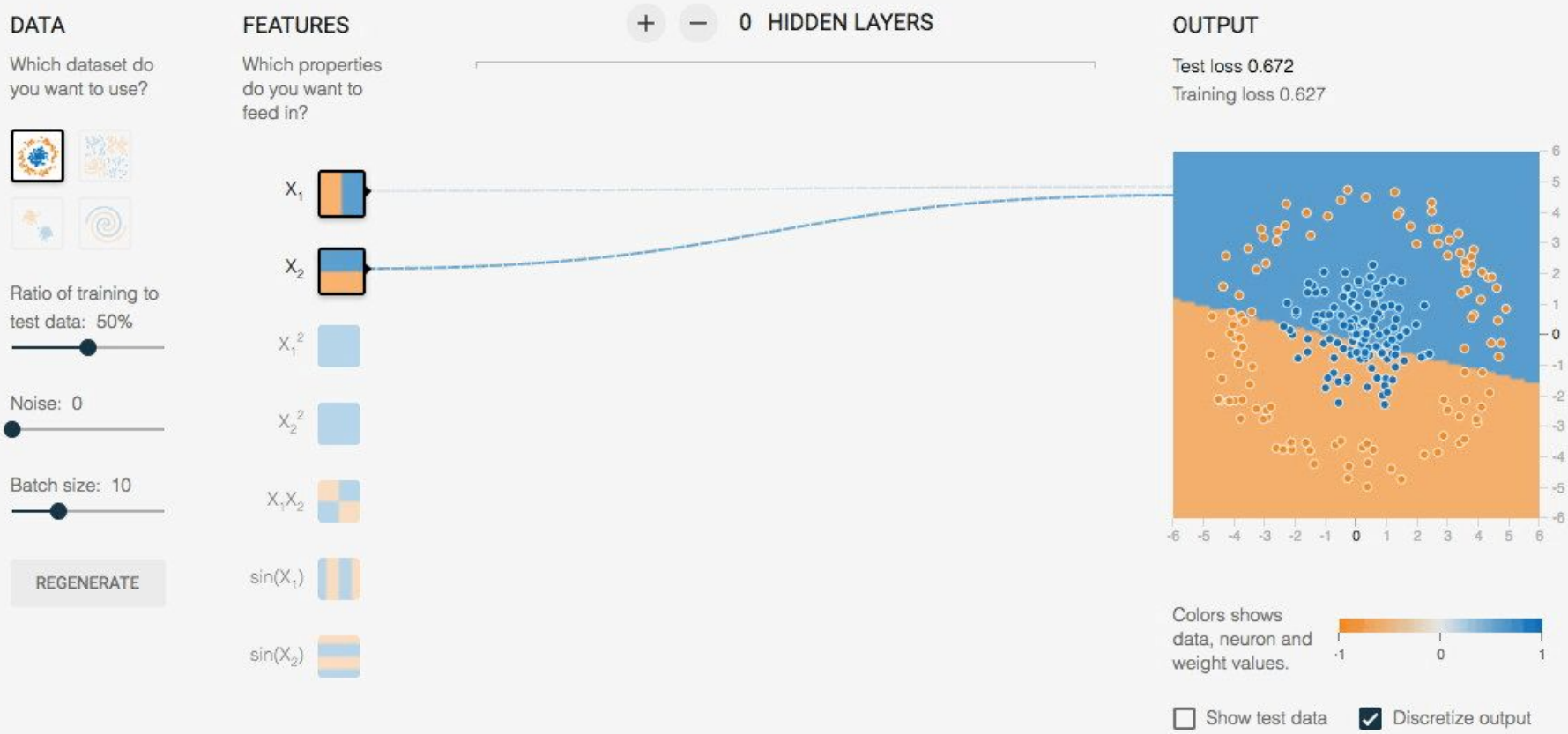
Representation learning

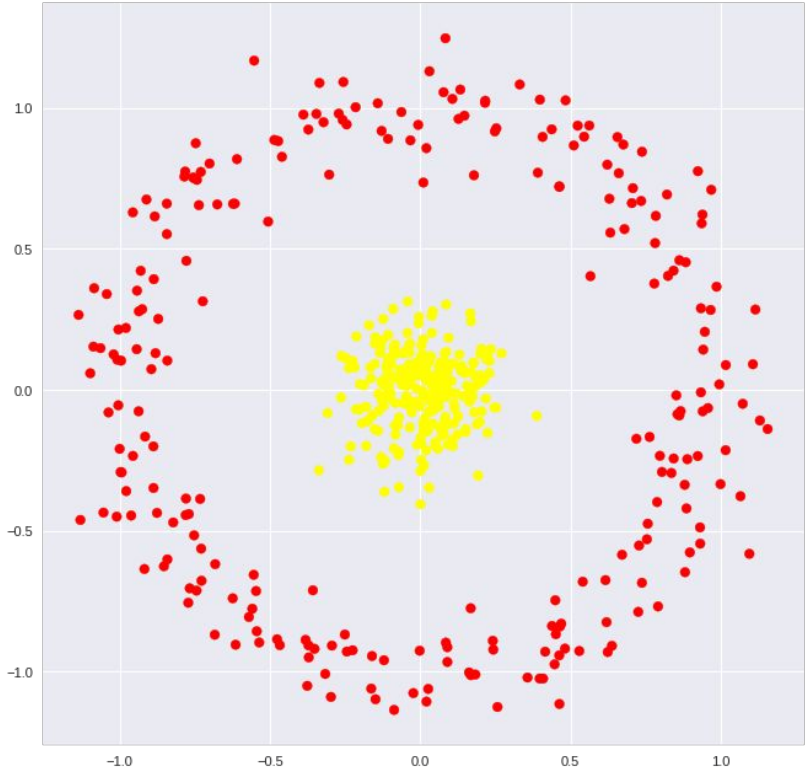
What is Deep Learning?

Quick demos with TensorFlow Playground

playground.tensorflow.org


DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
X1
X2
X1²
X2²
X1X2
sin(X1)
sin(X2)
+
−
0 HIDDEN LAYERS
OUTPUT
Test loss 0.455
Training loss 0.455
Colors shows data, neuron and weight values.
-1
0
1
Show test data
Discretize output


DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
X1
X2
X1²
X2²
X1X2
sin(X1)
sin(X2)
0 HIDDEN LAYERS
OUTPUT
Test loss 0.672
Training loss 0.627
Colors shows data, neuron and weight values.
-1
0
1
Show test data
Discretize output

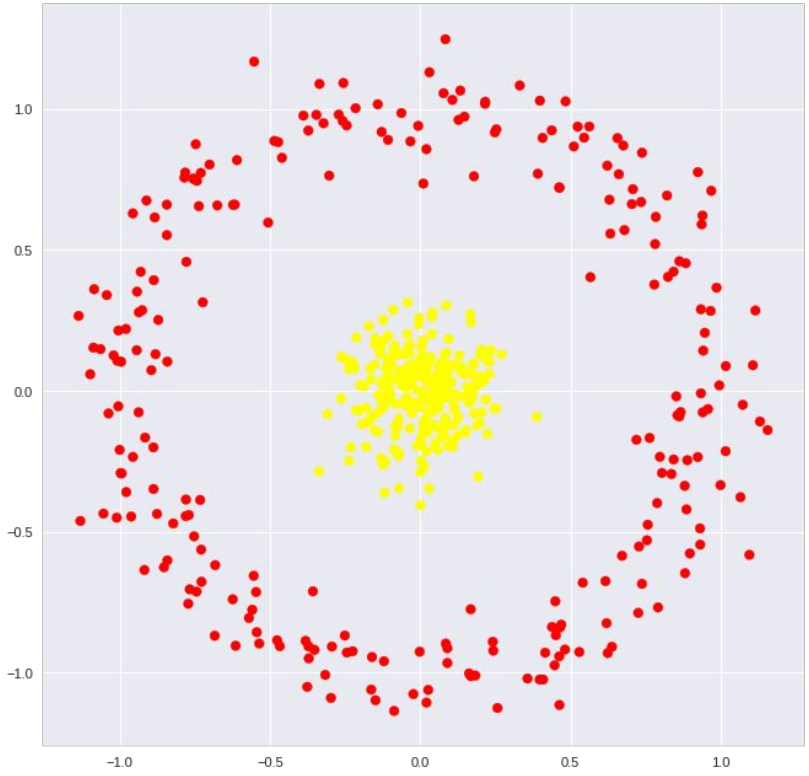


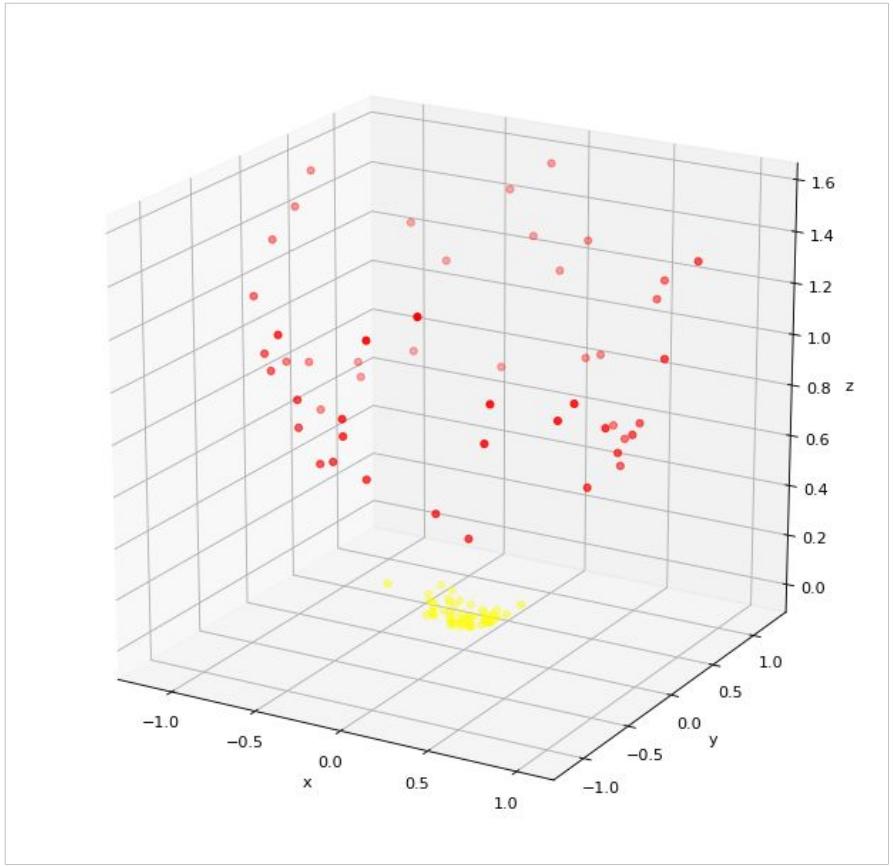
Feature engineering. What if we add a new feature:

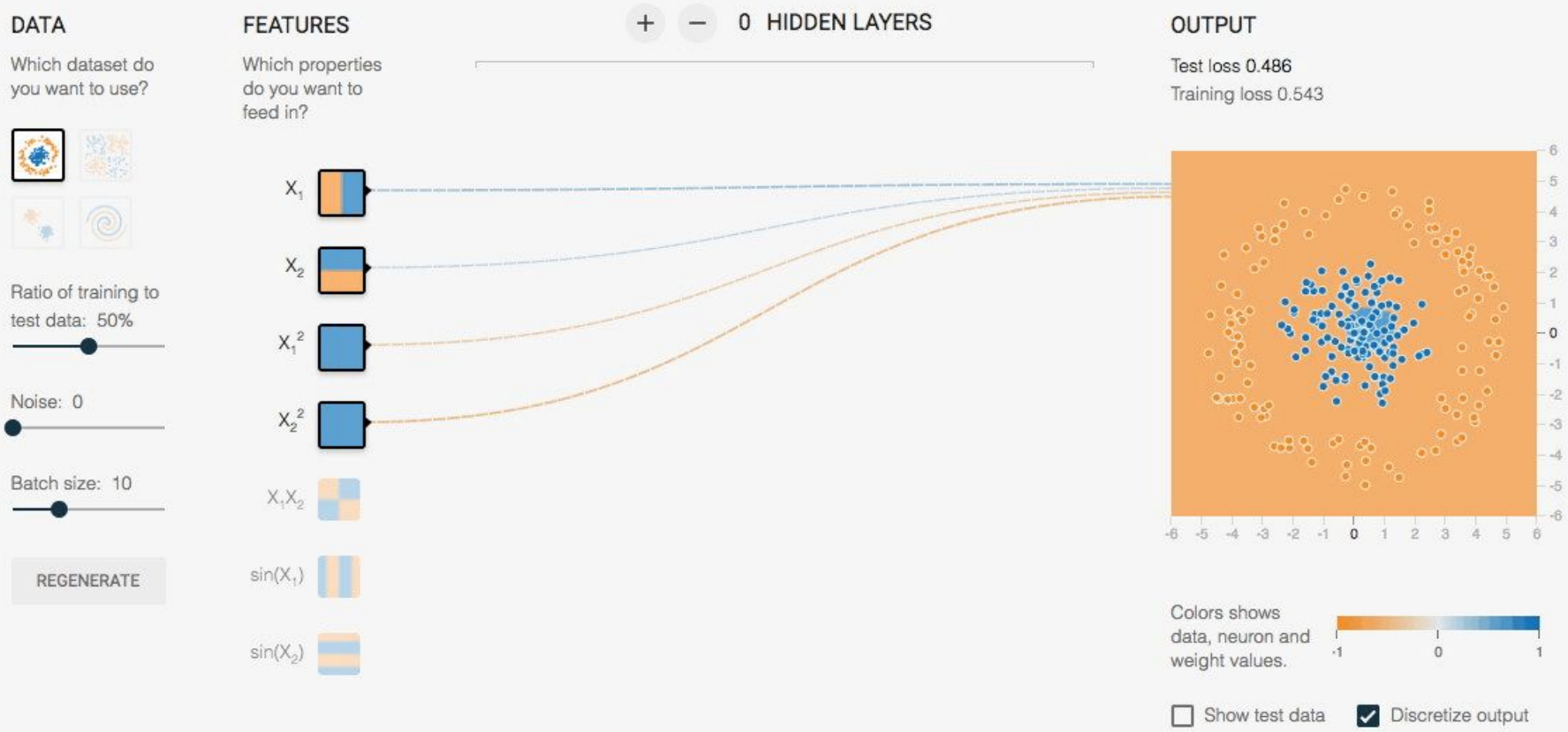
`z = x^2 + y^2.`

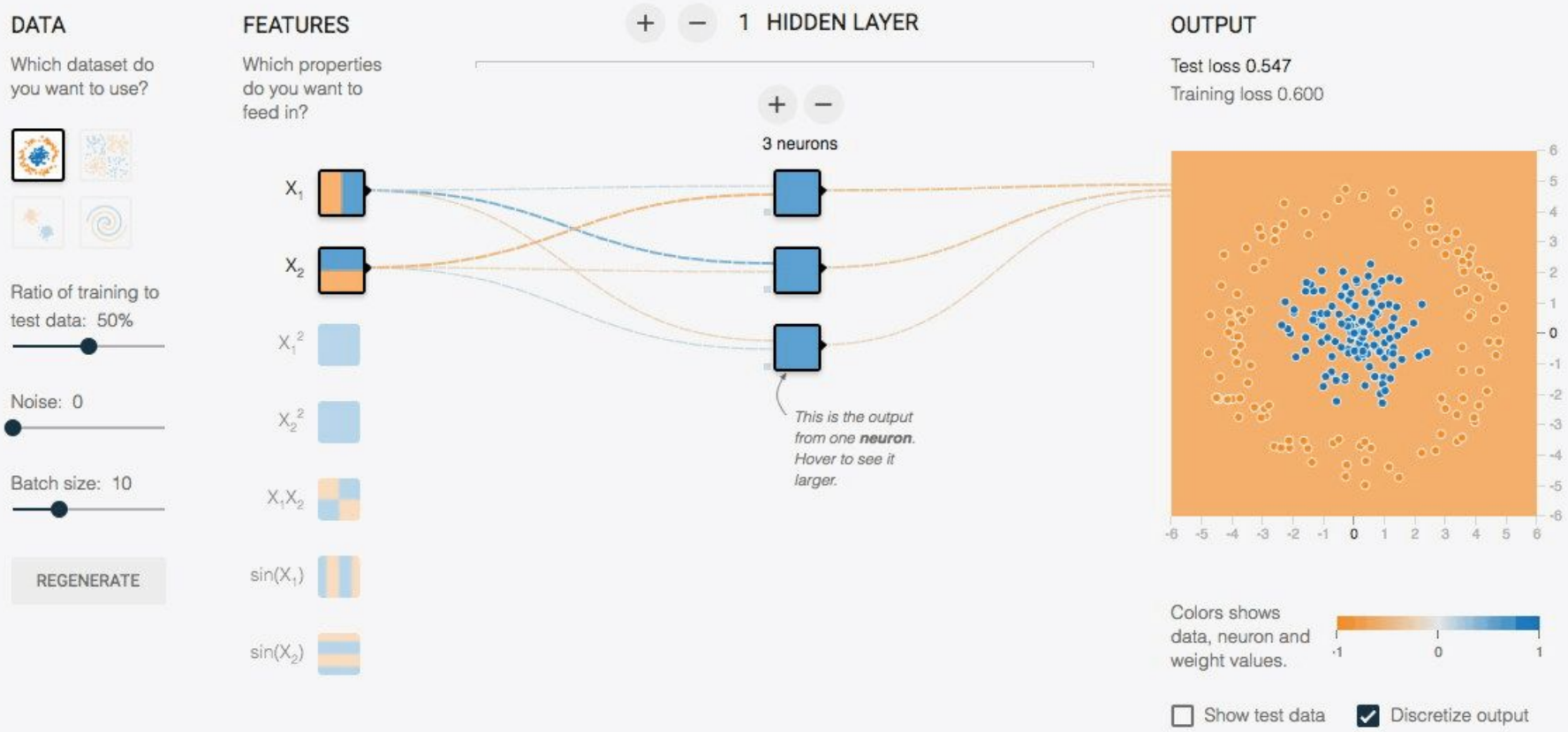
Intuition

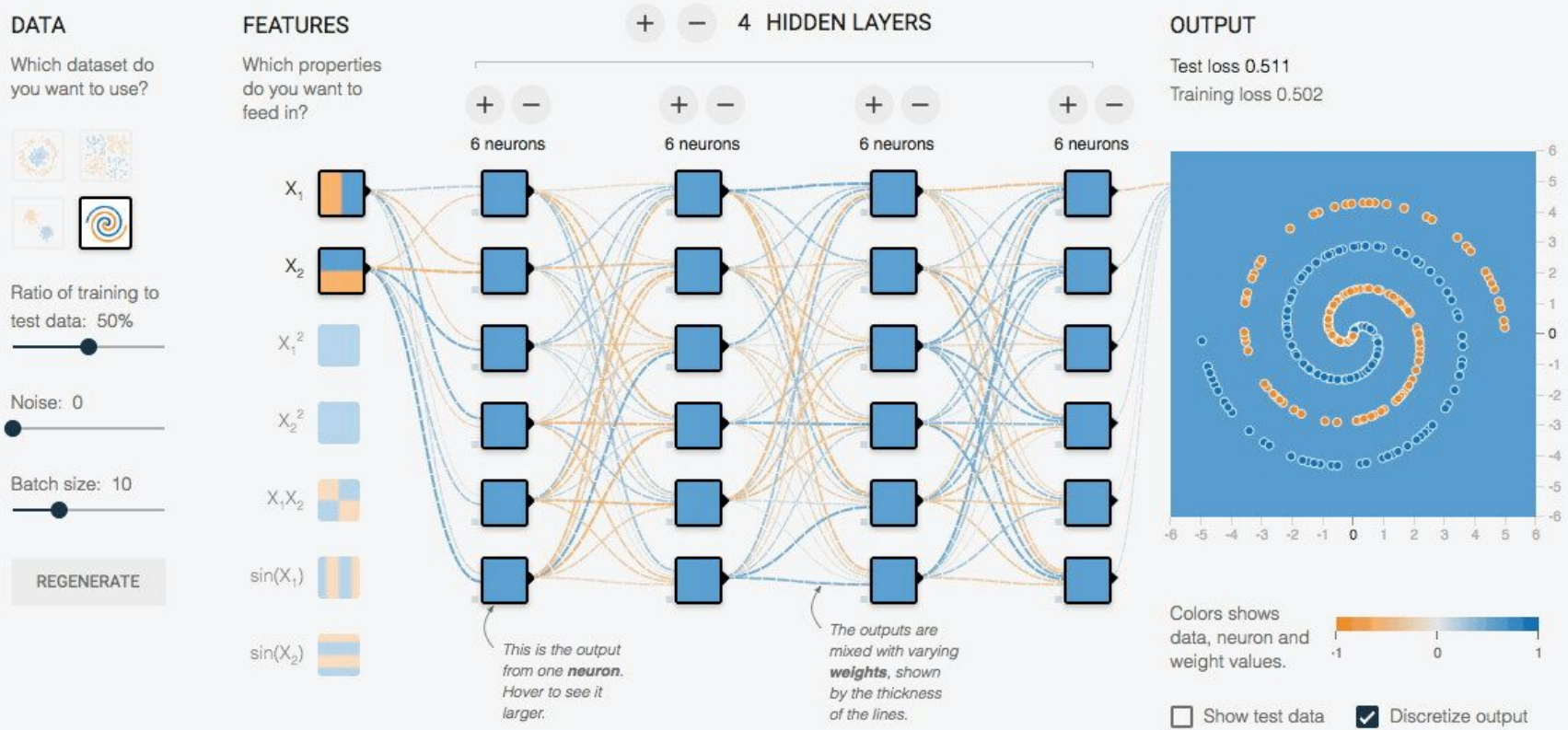
- All values of *z* will be positive.
- Yellow points are closer to the origin…
- So sum of their squared coords will be lower than red!

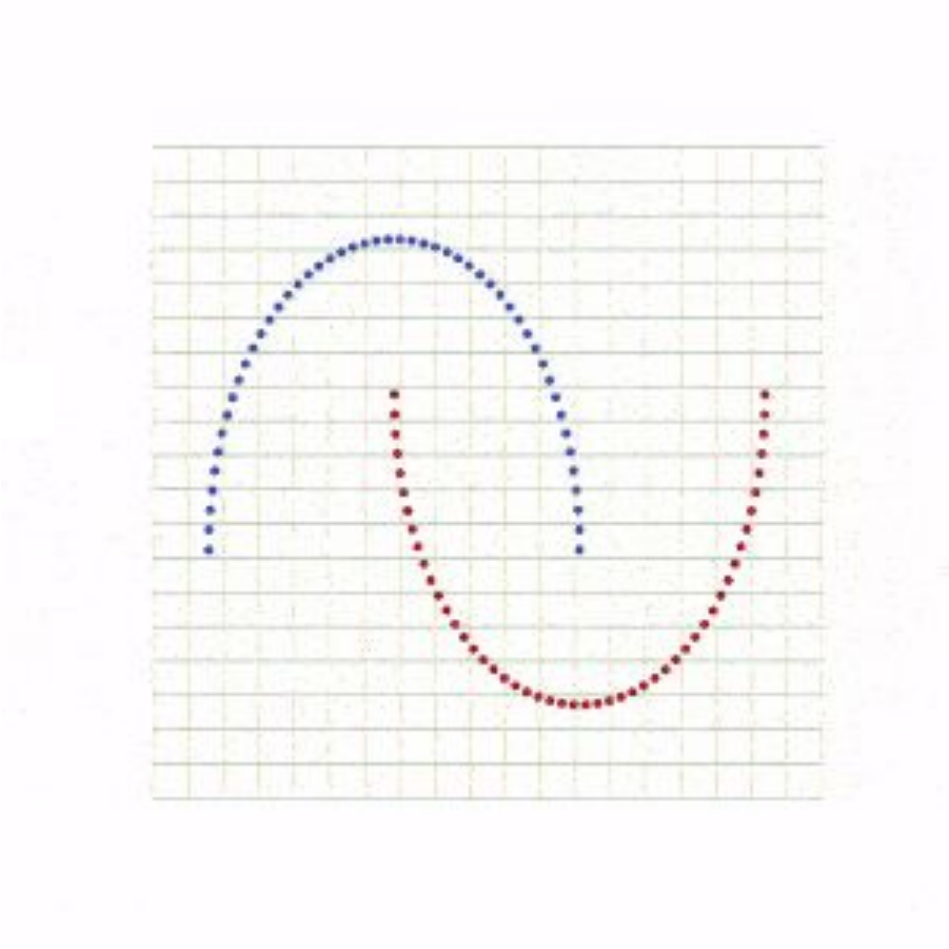

1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0


1.6
1.4
1.2
1.0
0.8
z
0.6
0.4
0.2
0.0
1.0
0.5
0.0
−0.5
−1.0
y
−1.0
−0.5
0.0
0.5
1.0
x


DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
X1
X2
X1²
X2²
X1X2
sin(X1)
sin(X2)
+
−
0 HIDDEN LAYERS
OUTPUT
Test loss 0.486
Training loss 0.543
Colors shows data, neuron and weight values.
-1
0
1
Show test data
Discretize output


DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
X1
X2
X1²
X2²
X1X2
sin(X1)
sin(X2)
1 HIDDEN LAYER
3 neurons
This is the output from one neuron. Hover to see it larger.
OUTPUT
Test loss 0.547
Training loss 0.600
Colors shows data, neuron and weight values.
-1
0
1
Show test data
Discretize output


DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
X1
X2
X1²
X2²
X1X2
sin(X1)
sin(X2)
4 HIDDEN LAYERS
6 neurons
6 neurons
6 neurons
6 neurons
This is the output from one neuron. Hover to see it larger.
The outputs are mixed with varying weights, shown by the thickness of the lines.
OUTPUT
Test loss 0.511
Training loss 0.502
Colors shows data, neuron and weight values.
-1
0
1
Show test data
Discretize output



Exercise 1

Linear regression in TensorFlow 2 (low-level, so you can see every piece involved).

Exercise 1

Goals

- Install TensorFlow 2.0
- Introduce Colaboratory
- Introduce the ingredients needed to train a neural network (training and testing data, **prediction**, **loss**, and **optimization**)

bit.ly/tf-ws1

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.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 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)
```


K

Keras and tf.keras

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`

Exercise 2

Fashion MNIST in TensorFlow 2.0

Exercise 2

Goals

- Learn about the Sequential API
- Train a simple image classifier

Visit

bit.ly/tf-ws4

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)
```

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

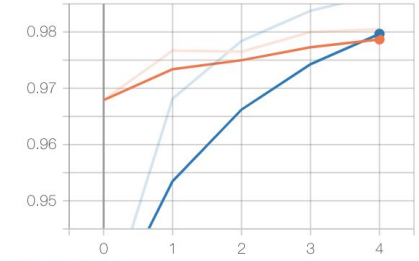
Write a regex to filter runs

20190227-033014/test
20190227-033014/train

Filter tags (regular expressions supported)

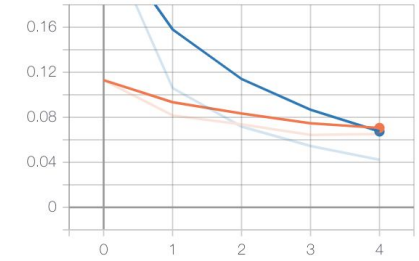
accuracy

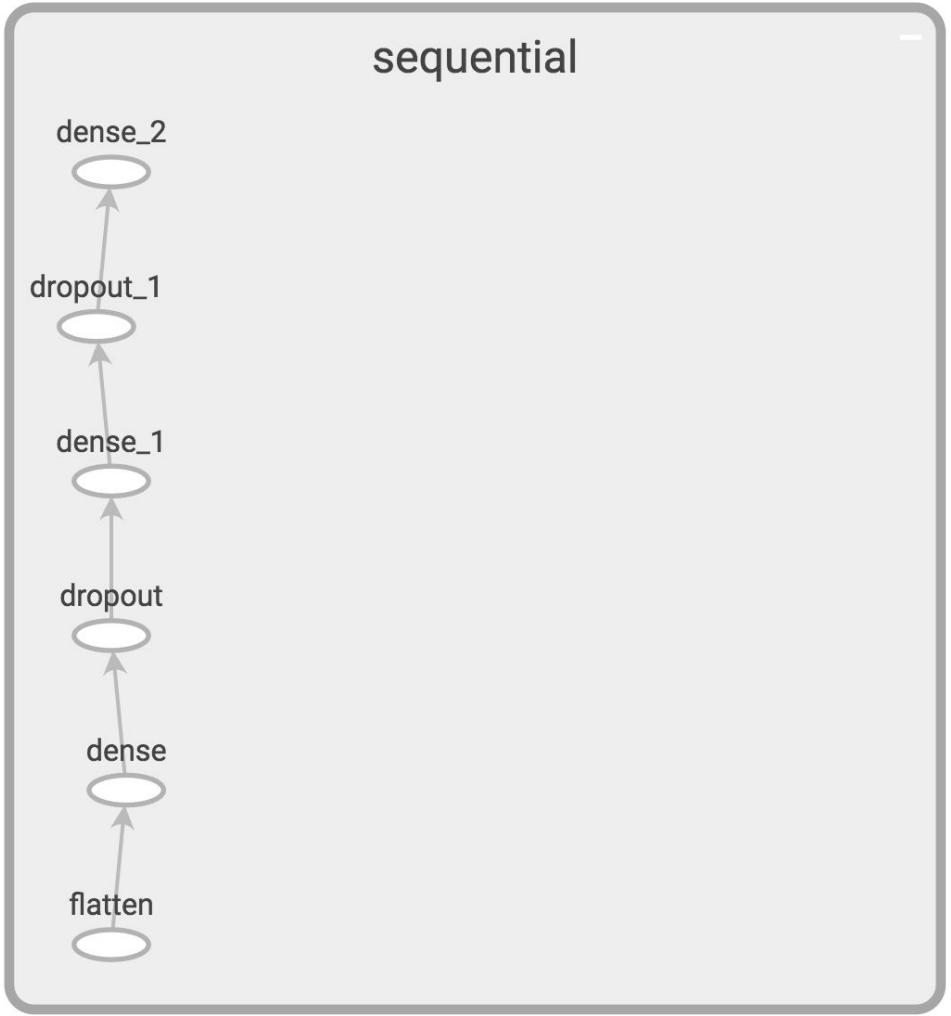
accuracy
tag: accuracy



loss

loss
tag: loss





Exercise 3

Keras model subclassing and TensorBoard

Exercise 3

Goals

- Learn about the Subclassing API
- Run TensorBoard in the browser

[bit.ly/tf-ws3](bit.ly/tf-ws3)

Code corrections

!pip install -q tensorflow==2.0.0-alpha0

!pip install -q tensorboard==1.13.0

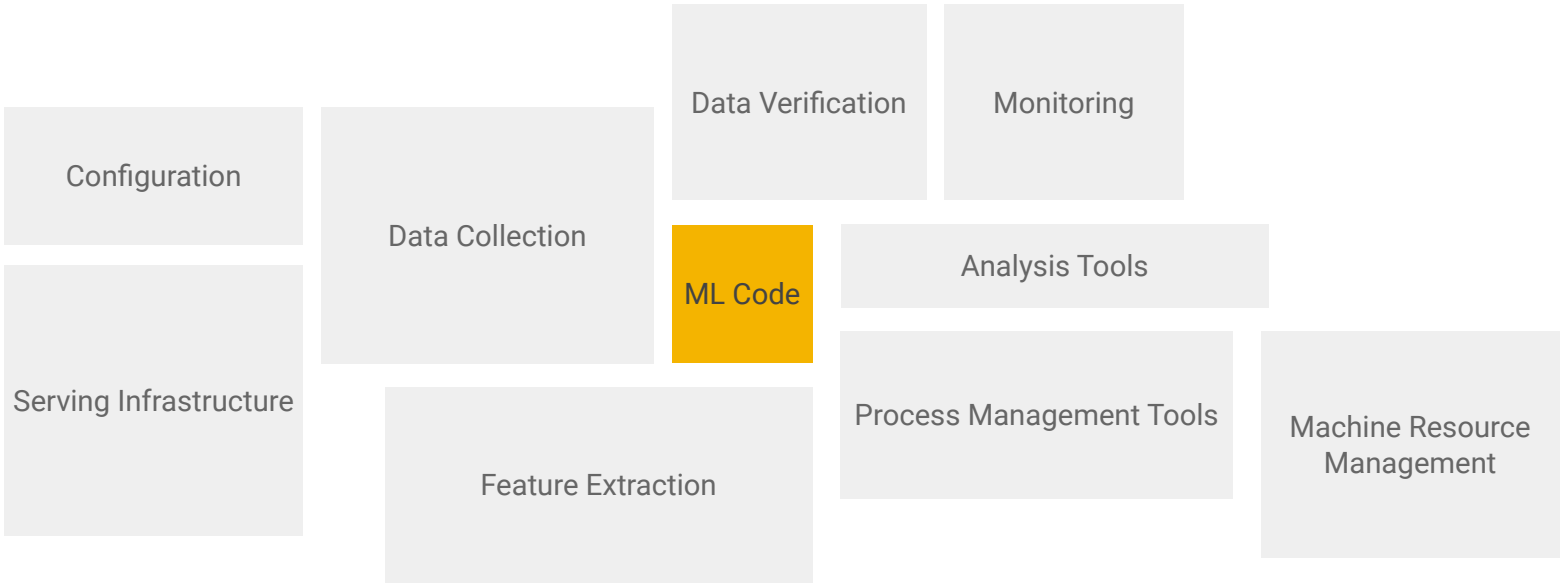
%load_ext tensorboard.notebook (thank you!)

TensorFlow Extended (TFX)

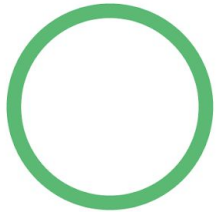
Real World Machine Learning in Production

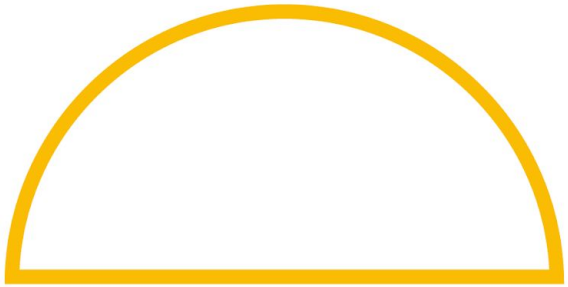
In addition to training a model ...

… a production solution requires so much more



Tensorflow Extended (TFX)





TensorFlow Extended (TFX)

Powers our most important bets and products

And some of our most important partners

airbnb

PayPal

Production Machine Learning

Machine Learning Development

- Labeled data
- Feature space coverage
- Minimal dimensionality
- Maximum predictive data
- Fairness
- Rare conditions
- Data lifecycle management

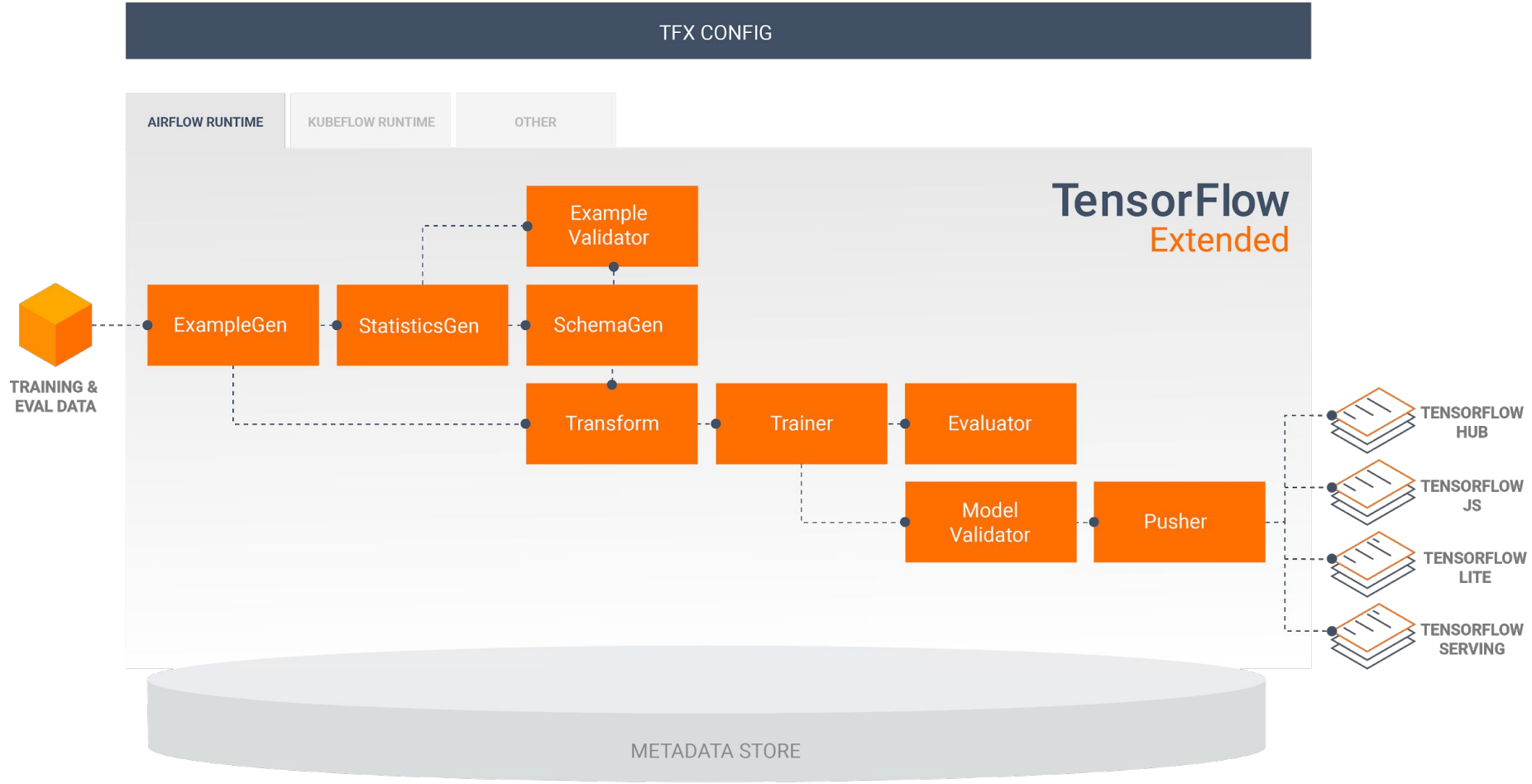
Production Machine Learning

Machine Learning Development

- Labeled data
- Feature space coverage
- Minimal dimensionality
- Maximum predictive data
- Fairness
- Rare conditions
- Data lifecycle management

Modern Software Development

- Scalability
- Extensibility
- Configuration
- Consistency & Reproducibility
- Modularity
- Best Practices
- Testability
- Monitoring
- Safety & Security


TFX CONFIG
AIRFLOW RUNTIME
KUBEFLOW RUNTIME
OTHER
TensorFlow
Extended
Example Validator
ExampleGen
StatisticsGen
SchemaGen
TRAINING & EVAL DATA
Transform
Trainer
Evaluator
Model Validator
Pusher
TENSORFLOW HUB
TENSORFLOW JS
TENSORFLOW LITE
TENSORFLOW SERVING
METADATA STORE

TensorFlow Serving

Running your trained model in a server/cluster

Key Concepts

- **Servables** are the central abstraction in TensorFlow Serving. Servables are the underlying objects that clients use to perform computation (for example, a lookup or inference)

Key Concepts

Servable Versions

TensorFlow Serving can handle one or more versions of a servable over the lifetime of a single server instance.

Servable Streams

A servable stream is the sequence of versions of a servable, sorted by increasing version numbers.

Key Concepts

Models

TensorFlow Serving represents a model as one or more servables.

Loaders

Loaders manage a servable's life cycle. The Loader API enables common infrastructure independent from specific servables.

Key Concepts

Sources

Sources are plugin modules that find and provide servables. Each Source provides zero or more servable streams. For each servable stream, a Source supplies one Loader instance for each version it makes available to be loaded.

Key Concepts

Aspired Versions

Aspired versions represent the set of servable versions that should be loaded and ready.

Key Concepts

Version Policy

Version Policies specify the sequence of version loading and unloading within a single servable stream.

- **Availability Preserving Policy** - *Avoid leaving zero versions* loaded; typically load a new version before unloading an old one
- **Resource Preserving Policy** - *Avoid having two versions* loaded simultaneously, thus requiring double the resources (unload an old version before loading a new one)
- **Custom -** Whatever you need

Key Concepts

Managers

Managers handle the full lifecycle of Servables, including:

- loading Servables
- serving Servables
- unloading Servables

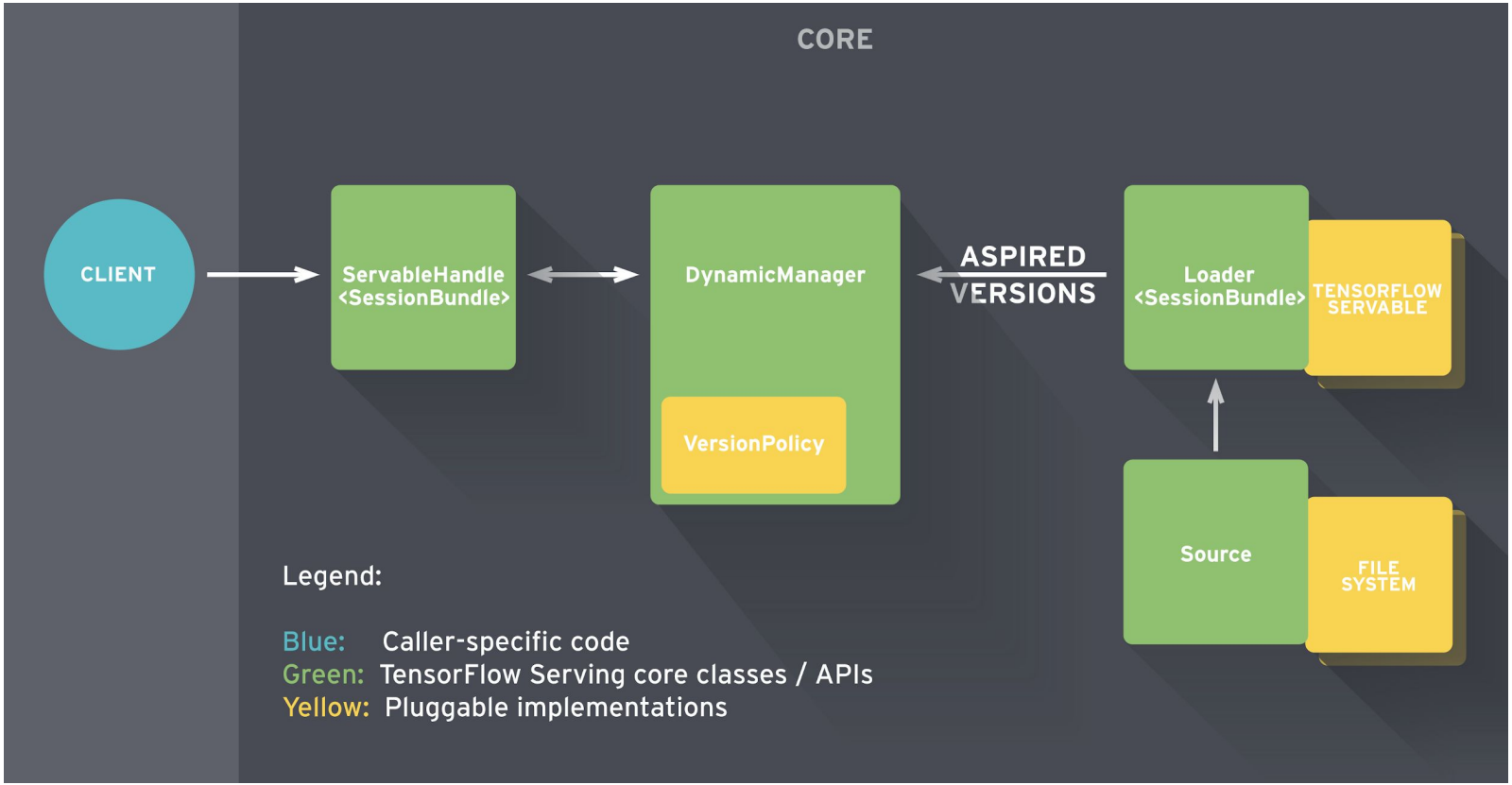
Key Concepts

ServableHandle

Managers provide a simple, narrow interface -- GetServableHandle() -- for clients to access loaded servable instances.

Clients ask the Manager for the Servable, either specifying a version explicitly or just requesting the latest version. The Manager returns a handle for the Servable.

Architecture



Exercise

- Train a model
- Save a SavedModel
- Serve it

http://bit.ly/serving-colab

TensorFlow

Beyond Hello World

A few of my favorites

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

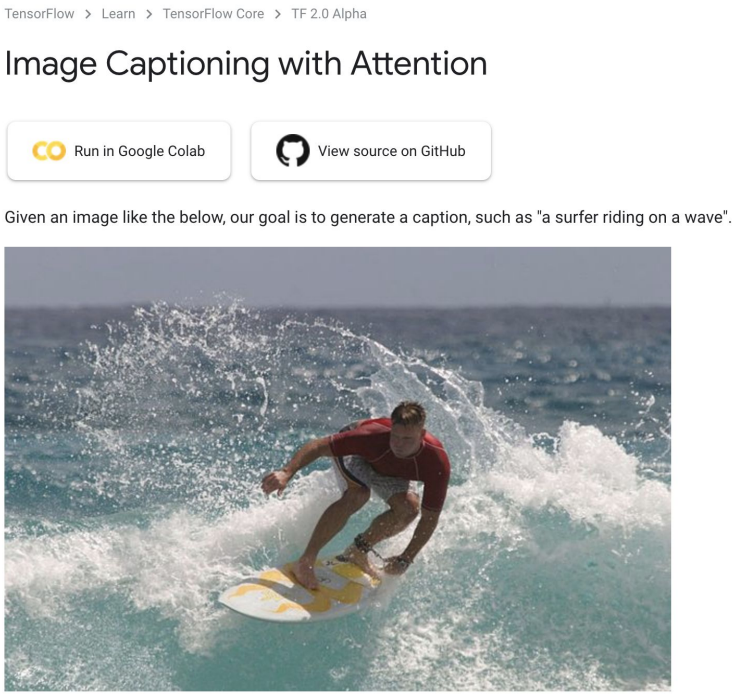
The docs are runnable

Tutorials on tf.org/alpha are

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

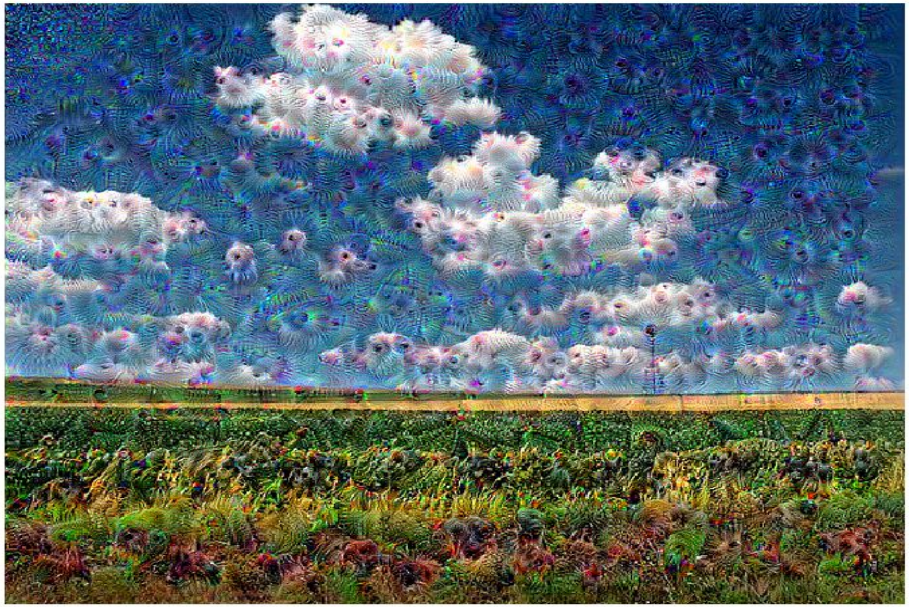
They automatically

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



tensorflow.org/alpha/tutorials/text/image_captioning

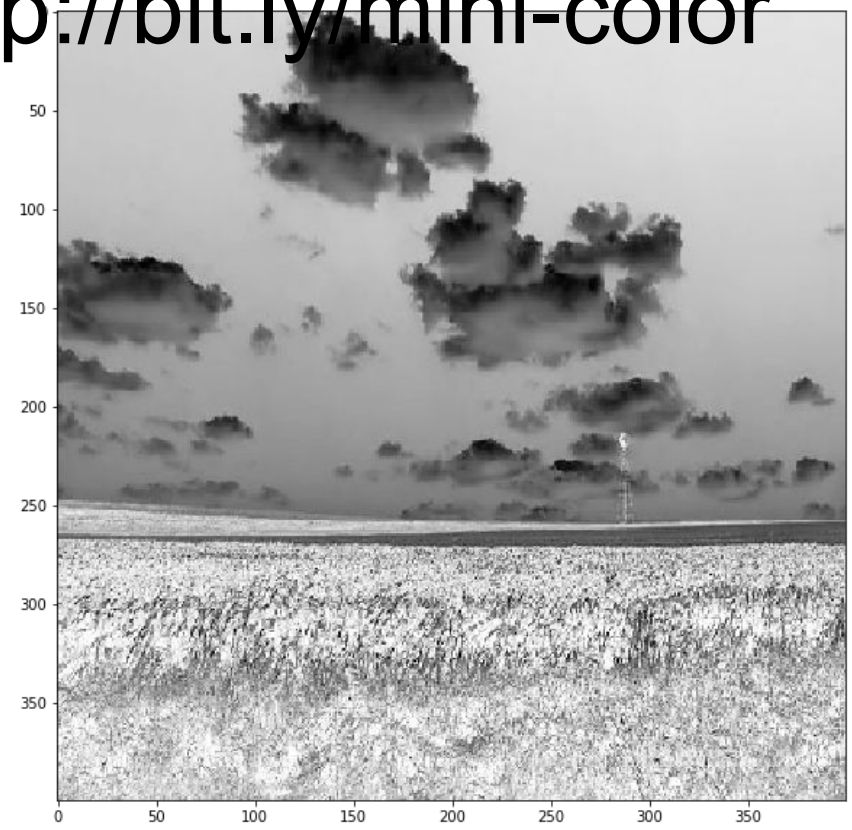
http://bit.ly/mini-dream



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

Code walkthrough

http://bit.ly/mini-color



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

Exercise 6

Deep Dream

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.

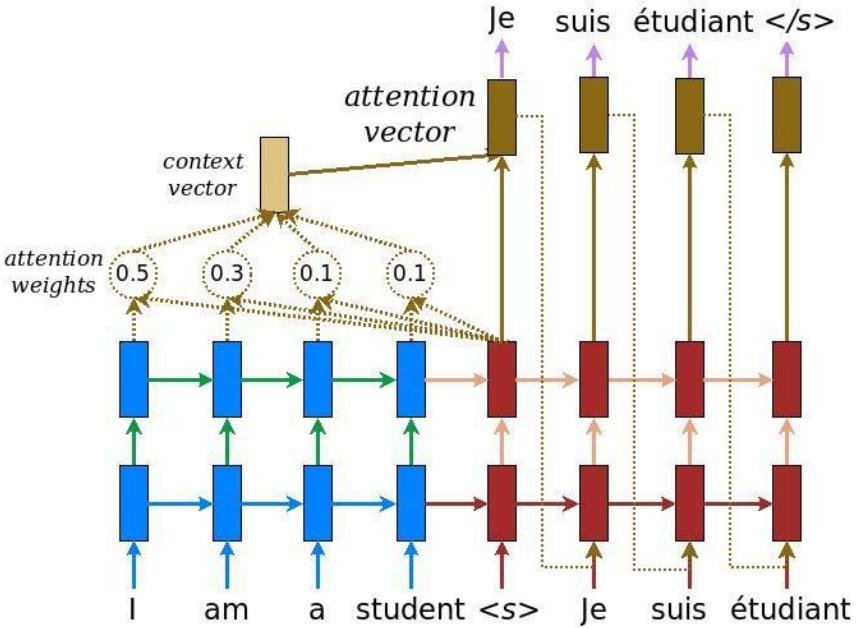
Exercise 7

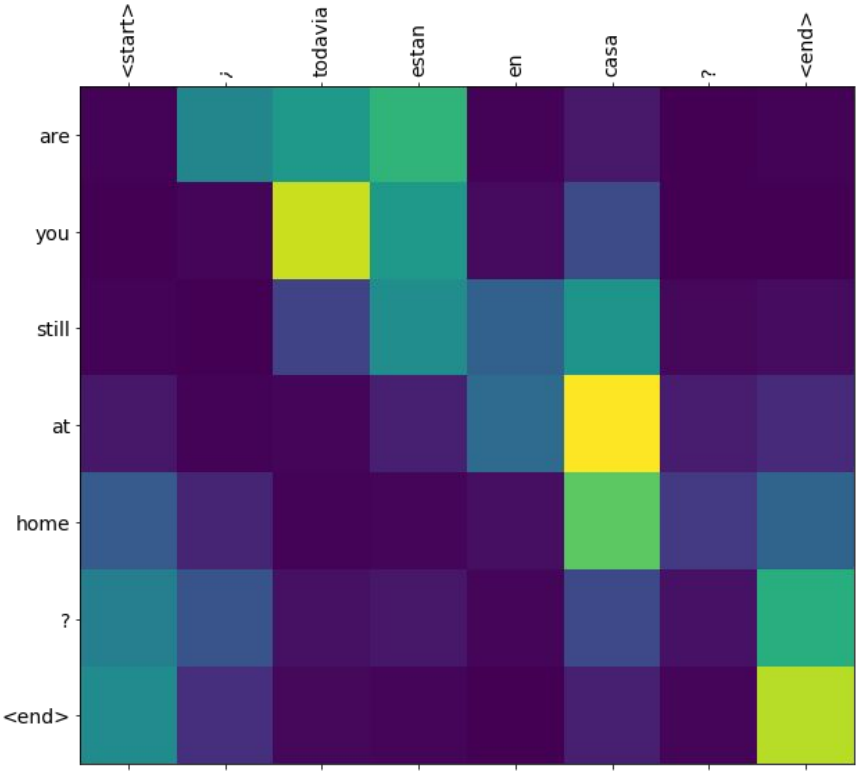
Goals

- Train an English to Spanish model, just for fun
- Learn about encoder / decoders

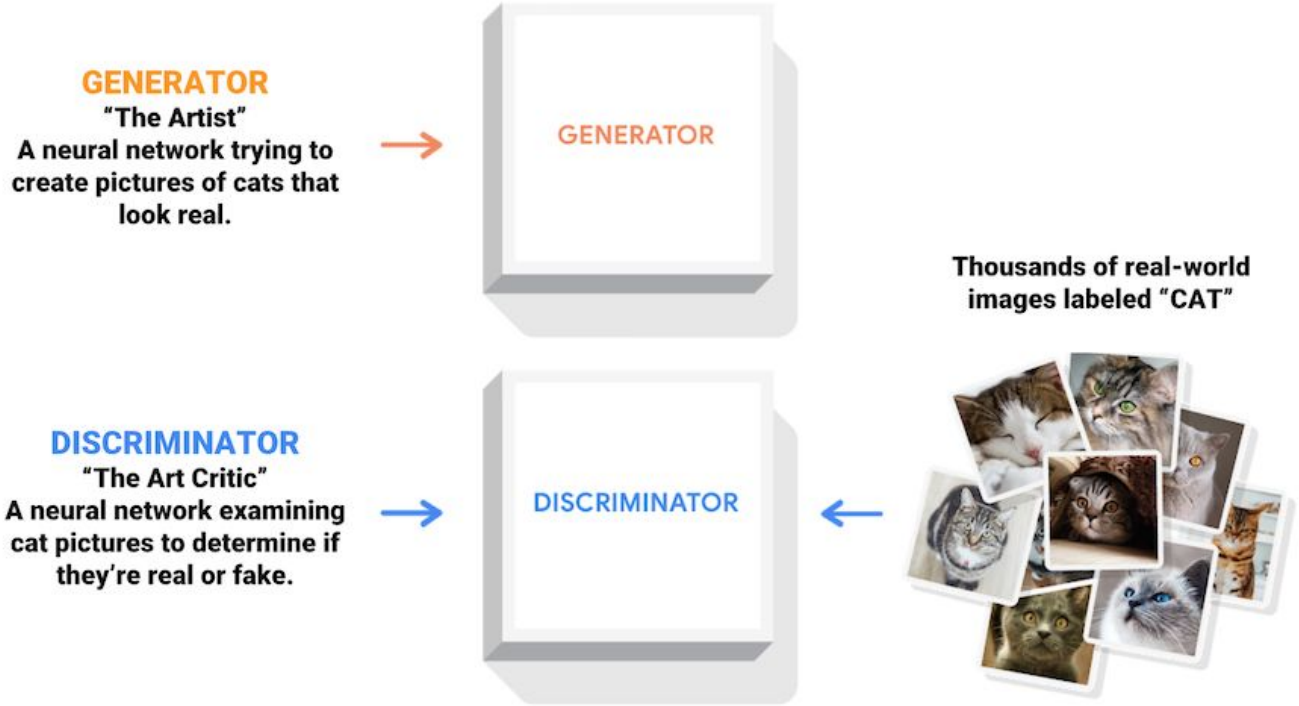
Visit

bit.ly/minimal-nmt

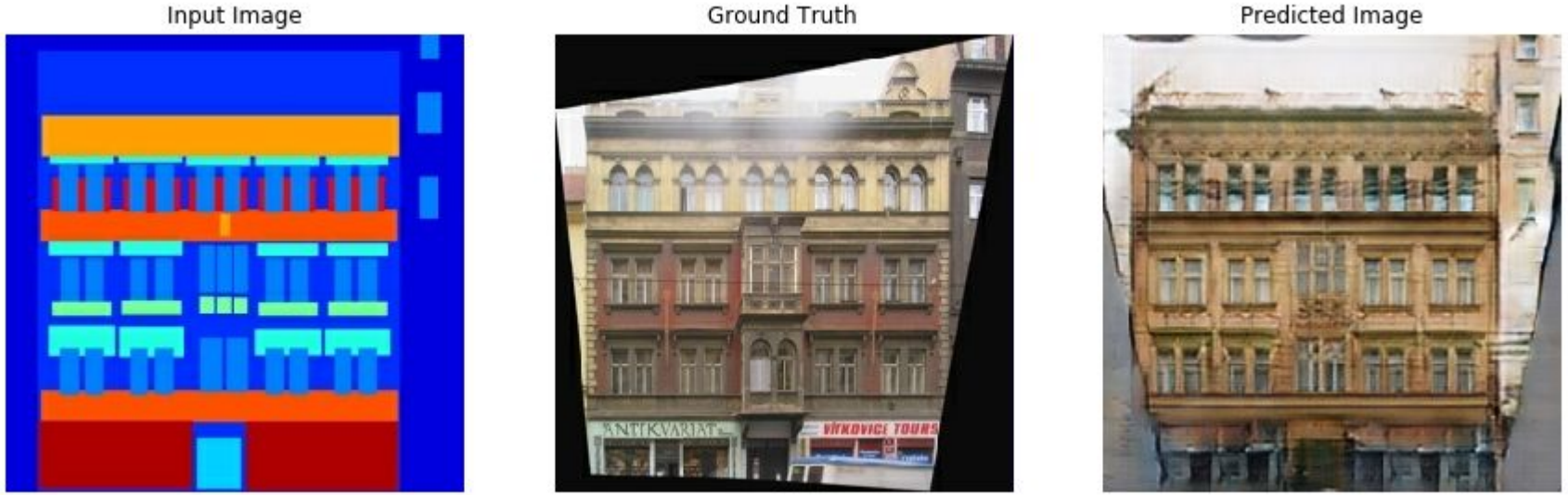




https://www.tensorflow.org/alpha/tutorials/sequences/nmt_with_attention



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

Exercise 6

Goals

- Use a pretrained CNN
- Extract intermediate activations
- Compute gradients w.r.t. an image

Visit

bit.ly/tf-ws6

TensorFlow

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

```
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

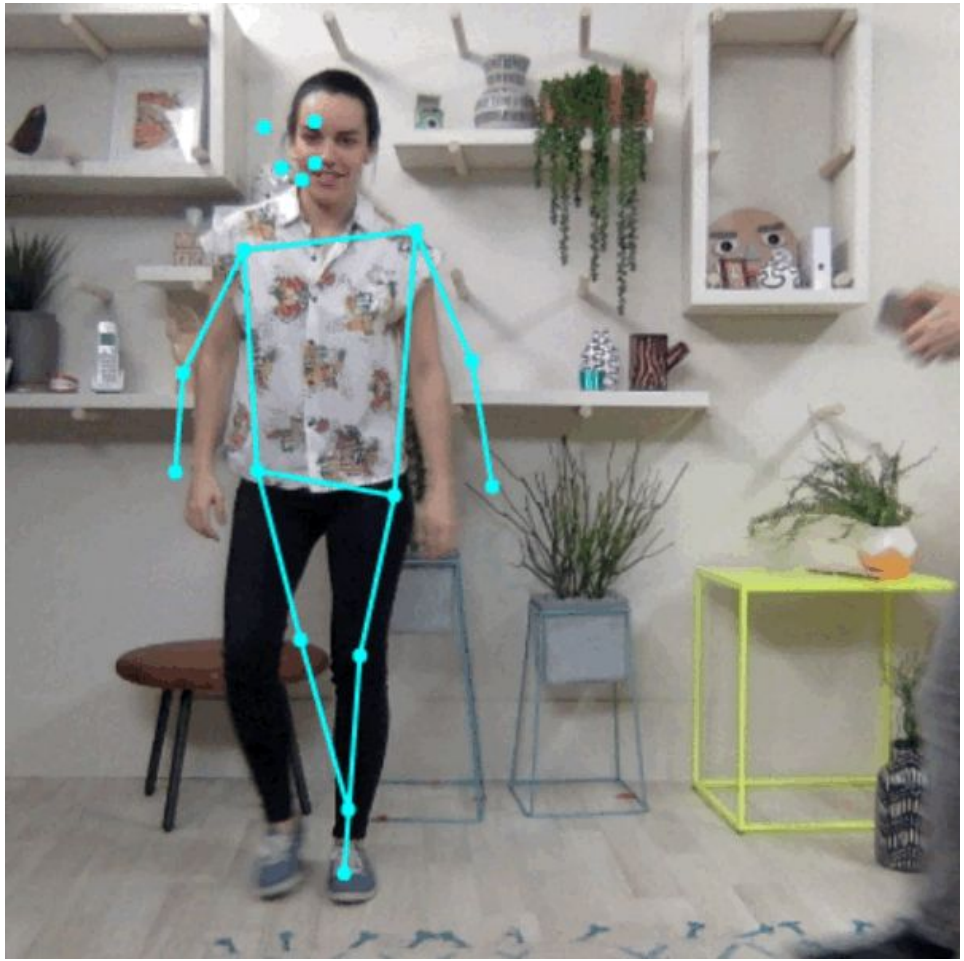
TensorFlow

TensorFlow.js

Demo #1

PoseNet

PoseNet



bit.ly/pose-net

Demo #2

BodyPix

BodyPix

bit.ly/body-pix

Exercise 10

Deploy a model in the browser

Exercise 10

Goals

- Train a model in Python and run it in a webpage, without leaving Colab

Visit

bit.ly/c-to-web

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 only, in +/- a month or two**)

Courses

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

tf.thanks!

Josh Gordon (twitter.com/random_forests)
Robert Crowe (twitter.com/robert_crowe)

Extras

Exercise 4

Structured data

Exercise 4

Goals

- Start from raw data
- Create a model

Visit

bit.ly/tf-ws4a

Facets pair-code.github.io/facets/