Intro to TensorFlow 2.0
MBL, August 2019

Josh Gordon (@random_forests)
Agenda 1 of 2

Exercises

- Fashion MNIST with dense layers
- CIFAR-10 with convolutional layers

Concepts (as many as we can intro in this short time)

- Gradient descent, dense layers, loss, softmax, convolution

Games

- QuickDraw
Agenda 2 of 2

Walkthroughs and new tutorials
- Deep Dream and Style Transfer
- Time series forecasting

Games
- Sketch RNN

Learning more
- Book recommendations
Deep Learning is representation learning
Latest tutorials and guides

tensorflow.org/beta

News and updates

medium.com/tensorflow
twitter.com/tensorflow
Demo

PoseNet and BodyPix
bit.ly/pose-net
bit.ly/body-pix
TensorFlow for JavaScript, Swift, Android, and iOS

tensorflow.org/js
tensorflow.org/swift
tensorflow.org/lite
Minimal MNIST in TF 2.0

A linear model, neural network, and deep neural network - then a short exercise.

bit.ly/mnist-seq
model = Sequential()
model.add(Dense(256, activation='relu', input_shape=(784,)))
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))

Linear model  \[ f(x) = \text{softmax}(W_1 x) \]

Neural network  \[ f(x) = \text{softmax}(W_2(g(W_1 x))) \]

Deep neural network  \[ f(x) = \text{softmax}(W_3(g(W_2(g(W_1 x))))) \]
After training, select all the weights connected to this output.

```python
model.layers[0].get_weights()

# Your code here
# Select the weights for a single output
# ...

img = weights.reshape(28, 28)
plt.imshow(img, cmap = plt.get_cmap('seismic'))
```
After training, select all the weights connected to this output.
Exercise 1 (option #1)

Exercise: bit.ly/mnist-seq

Reference:
tensorflow.org/beta/tutorials/keras/basic_classification

TODO:
Add a validation set. Add code to plot loss vs epochs (next slide).
Exercise 1 (option #2)

bit.ly/ijcav_adv

Answers: next slide.
import matplotlib.pyplot as plt

# Add a validation set
history = model.fit(x_train, y_train, validation_data=(x_test, y_test), ...)

# Get stats from the history object
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
epochs = range(len(acc))

# Plot accuracy vs epochs
plt.title('Training and validation accuracy')
plt.plot(epochs, acc, color='blue', label='Train')
plt.plot(epochs, val_acc, color='orange', label='Val')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
Exercise 1 (option #2)

bit.ly/ijcav_adv

Answers: next slide.

bit.ly/ijcai_adv_answer
About TensorFlow 2.0
Install

# GPU
!pip install tensorflow-gpu==2.0.0-beta1

# CPU
!pip install tensorflow==2.0.0-beta1

In either case, check your installation (in Colab, you may need to use runtime -> restart after installing).

import tensorflow as tf
print(tf.__version__) # 2.0.0-beta1

Nightly is available too, but best bet: stick with a named release for stability.
import tensorflow as tf
print(tf.__version__) # 2.0.0-beta1

x = tf.constant(1)
y = tf.constant(2)
z = x + y

print(z) # tf.Tensor(3, shape=(), dtype=int32)
You can interactive explore layers

```python
from tensorflow.keras.layers import Dense

layer = Dense(units=1, kernel_initializer='ones', use_bias=False)
data = tf.constant([[1.0, 2.0, 3.0]])  # Note: a batch of data
print(data)  # tf.Tensor([[1. 2. 3.]], shape=(1, 3), dtype=float32)

# Call the layer on our data
result = layer(data)

print(result)  # tf.Tensor([[6.]], shape=(1, 1), dtype=float32)
print(result.numpy())  # tf.Tensors have a handy .numpy() method
```
TF1: Build a graph, then run it.

```python
import tensorflow as tf # 1.14.0
print(tf.__version__)

x = tf.constant(1)
y = tf.constant(2)
z = tf.add(x, y)

print(z)
```
TF1: Build a graph, then run it.

```python
import tensorflow as tf  # 1.14.0
print(tf.__version__)

x = tf.constant(1)
y = tf.constant(2)
z = tf.add(x, y)

print(z)  # Tensor("Add:0", shape=(), dtype=int32)

with tf.Session() as sess:
    print(sess.run(x))  # 3
```
Keras is built-in to TF2
How to import tf.keras

If you want to use tf.keras and see the message “Using TensorFlow Backend”, you have accidentally imported Keras (which is installed by default on Colab) from outside of TensorFlow.

Example

```python
# !pip install tensorflow==2.0.0-beta1, then
>>> from tensorflow.keras import layers # Right
>>> from keras import layers # Oops
Using TensorFlow backend. # You shouldn't see this
```

When in doubt, copy the imports from one of the tutorials on tensorflow.org/beta
Notes

A superset of the reference implementation. Built-in to TensorFlow 2.0 (no need to install Keras separately).

Documentation and examples

- Tutorials: tensorflow.org/beta
- Guide: tensorflow.org/beta/guide/keras/

```python
!pip install tensorflow==2.0.0-beta1
from tensorflow import keras
```

I'd recommend the examples you find on tensorflow.org/beta over other resources (they are better maintained and most of them are carefully reviewed).

tf.keras adds a bunch of stuff, including…
model subclassing (Chainer / PyTorch style model building), custom training loops using a GradientTape, a collection of distributed training strategies, support for TensorFlow.js, Android, iOS, etc.
More notes

TF 2.0 is similar to NumPy, with:

- GPU support
- Autodiff
- Distributed training
- JIT compilation
- A portable format (train in Python on Mac, deploy on iOS using Swift, or in a browser using JavaScript)

Write models in Python, JavaScript or Swift (and run anywhere).

API doc: tensorflow.org/versions/r2.0/api_docs/python/tf

Note: make sure you’re looking at version 2.0 (the website still defaults to 1.x)
Three model building styles

Sequential, Functional, Subclassing
Sequential models

```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.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)
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)
Functional models

inputs = keras.Input(shape=(32, 32, 3))

y = layers.Conv2D(3, (3, 3), activation='relu', padding='same')(inputs)
outputs = layers.add([inputs, y])
model = keras.Model(inputs, outputs)

keras.utils.plot_model(model, 'skip_connection.png', show_shapes=True)
Subclassed models

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)
Two training styles

Built-in and custom
Use a built-in training loop

```python
model.fit(x_train, y_train, epochs=5)
```
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))
A few concepts
Calculate the gradient.
Take a step.
Repeat.

Gradient descent

A vector of partial derivatives.

Gradient points in direction of steepest ascent, so we step in reverse direction.

Step size (learning rate).

Parameter

Loss
With more than one variable

The gradient is a vector of partial derivatives (the derivative of a function w.r.t. each variable while the others are held constant).

\[ \nabla_w \text{Loss} = \frac{\partial \text{Loss}}{\partial w_0}, \frac{\partial \text{Loss}}{\partial w_1} \]

The gradient points in the direction of steepest ascent. We usually want to minimize a function (like loss), so we take a step in the opposite direction.
Training models with gradient descent

Forward pass

- Linear regression: $y = mx + b$
- Neural network: $f(x) = \text{softmax}(W_2(g(W_1x)))$

Calculate loss

- Regression: squared error.
- Classification: cross entropy.

Backward pass

- Backprop: efficient method to calculate gradients
- Gradient descent: nudge parameters a bit in the opposite direction
Try it: Linear regression

bit.ly/tf-ws1

Bonus: Deep Dream training loop will be similar.
A neuron

Bias not drawn (you could set $x_1$ to be a constant input of 1).

Linear combination of inputs and weights

$$\hat{y} = g \left( \sum x_i \theta_i \right)$$

Can rewrite as a dot product

$$\hat{y} = g \left( x^T \theta \right)$$
One image and one class

Interpret as “how strongly do you think this image is a plane?”

Multiple inputs; one output
One image and two classes

W is now a matrix

\[ W = \begin{bmatrix} 1.4 & 0.5 & 0.7 & 1.2 \\ -2.0 & 0.1 & 0.2 & -0.7 \end{bmatrix}, \quad x = \begin{bmatrix} 12 \\ 48 \\ 96 \\ 18 \end{bmatrix}, \quad b = \begin{bmatrix} 0.5 \\ 1.2 \end{bmatrix} \]

\[ \text{Output} = W \times x + b = \begin{bmatrix} 130.1 \\ -11.4 \end{bmatrix} \]

Multiple inputs; multiple outputs
Two images and two classes

Image 1

12  48
96  18

Image 2

4   18
2   96

N x D

<table>
<thead>
<tr>
<th></th>
<th>1.4</th>
<th>0.5</th>
<th>0.7</th>
<th>1.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-2.0</td>
<td>0.1</td>
<td>0.2</td>
<td>-0.7</td>
</tr>
<tr>
<td>2</td>
<td>0.2</td>
<td>0.9</td>
<td>-0.2</td>
<td>0.5</td>
</tr>
</tbody>
</table>

D x batch_size

<table>
<thead>
<tr>
<th></th>
<th>12</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>48</td>
<td>18</td>
</tr>
<tr>
<td>2</td>
<td>96</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>18</td>
<td>96</td>
</tr>
</tbody>
</table>

N x 1

<table>
<thead>
<tr>
<th></th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.2</td>
</tr>
<tr>
<td>2</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Output

Image 1 | Image 2
---|---
130.1 | 131.7 | Plane
-11.4 | -71.7 | Car
12.8  | 64.8  | Truck

N x batch_size

W
Weights

x
Inputs

b
Bias

Scores
After training, select all the weights connected to this output.

```python
model.layers[0].get_weights()

# Your code here
# Select the weights for a single output
# ...

img = weights.reshape(28,28)
plt.imshow(img, cmap = plt.get_cmap('seismic'))
```
After training, select all the weights connected to this output.
A neural network

\[ f = W_2 g(Wx) \]

**Sigmoid Function**

\[ g(z) = \frac{1}{1 + e^{-z}} \]
\[ g'(z) = g(z)(1 - g(z)) \]

**Hyperbolic Tangent**

\[ g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \]
\[ g'(z) = 1 - g(z)^2 \]

**Rectified Linear Unit (ReLU)**

\[ g(z) = \max(0, z) \]
\[ g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases} \]
### ReLU

<table>
<thead>
<tr>
<th>130.1</th>
<th>Plane</th>
</tr>
</thead>
<tbody>
<tr>
<td>-11.4</td>
<td>Car</td>
</tr>
<tr>
<td>12.8</td>
<td>Truck</td>
</tr>
</tbody>
</table>

**Output**

Scores

\[
g(z) = \max(0, z)
\]

\[
g'(z) = \begin{cases} 
1, & z > 0 \\
0, & \text{otherwise}
\end{cases}
\]

\[
f = W_2 \cdot g(Wx)
\]
### Applied piecewise

<table>
<thead>
<tr>
<th>Score</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>130.1</td>
<td>Plane</td>
</tr>
<tr>
<td>-11.4</td>
<td>Car</td>
</tr>
<tr>
<td>12.8</td>
<td>Truck</td>
</tr>
</tbody>
</table>

Output Scores

$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 
1, & z > 0 \\
0, & \text{otherwise}
\end{cases}$$

Rectified Linear Unit (ReLU)

$$f = W_2 g(Wx)$$

![ReLU](image)

$$g(130.1)$$ Plane

$$g(-11.4)$$ Car

$$g(12.8)$$ Truck

$$= 130.1$$ Plane

$$= 0$$ Car

$$= 12.8$$ Truck
**Notes**
- You can make similar plots (and more) with this [example](https://www.tensorflow.org). Note: from an older version of TF, but should work out of the box in Colab.
- Each of our convolutional layers used an activation as well (not shown in previous slides).
- You can make a demo of this in [TensorFlow Playground](https://www.tensorflow.org) by setting activation = Linear (or none)
Without activation, many layers are equivalent to one

```python
# If you replace 'relu' with 'None', this model ...
model = Sequential(
    [Dense(256, activation='relu', input_shape=(2,)),
     Dense(256, activation='relu'),
     Dense(256, activation='relu'),
     Dense(1, activation='sigmoid')
])

# ... has the same representation power as this one
model = Sequential([Dense(1, activation='sigmoid', input_shape=(2,))])
```
Softmax converts scores to probabilities

Scores | Probabilities
--- | ---
130.1 | Plane | softmax([130.1, -11.4, 12.8])
-11.4 | Car | \[0.999, 0.001, 0.001\]
12.8 | Truck | 

Note: these are ‘probability like’ numbers (do not go to vegas and bet in this ratio).
Cross entropy compares two distributions

Each example has a label in a one-hot format

This is a bird

Cross entropy loss for a batch of examples

\[ L = - \sum \hat{y} \ln(y_i) \]

True prob (either 1 or 0) in our case!

Sum over all examples

True probabilities

Predicted prob (between 0-1)

Predicted probabilities

Rounded! Softmax output is always \(0 < x < 1\)
Exercise

bit.ly/ijcai_1-a

Complete the notebook for Fashion MNIST

Answers: next slide.
Exercise

bit.ly/ijcai_1-a

Complete the notebook for Fashion MNIST

Answers: bit.ly/ijcai_1-a_answers
TensorFlow RFP

jbgoron@google.com

goo.gle/tensorflow-rfp
Convolution
Not a Deep Learning concept

```python
import scipy
from skimage import color, data
import matplotlib.pyplot as plt
img = data.astronaut()
img = color.rgb2gray(img)
plt.axis('off')
plt.imshow(img, cmap=plt.cm.gray)
```
Convolution example

Notes
Edge detection intuition: dot product of the filter with a region of the image will be zero if all the pixels around the border have the same value as the center.

Does anyone know who this is?
Convolution example

Notes
Edge detection intuition: dot product of the filter with a region of the image will be zero if all the pixels around the border have the same value as the center.
A simple edge detector

```python
kernel = np.array([[-1,-1,-1],
                   [-1,8,-1],
                   [-1,-1,-1]])

result = scipy.signal.convolve2d(img, kernel, 'same')

plt.axis('off')

plt.imshow(result, cmap=plt.cm.gray)
```
Easier to see with seismic

Notes
Edge detection intuition: dot product of the filter with a region of the image will be zero if all the pixels around the border have the same value as the center.

Eileen Collins
Example

An input image (no padding)

<table>
<thead>
<tr>
<th>2</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

A filter (3x3)

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Output image (after convolving with stride 1)
Example

An input image (no padding)

A filter (3x3)

Output image (after convolving with stride 1)

2*1 + 0*0 + 1*1 + 0*0 + 1*0 + 0*0 + 0*0 + 0*1 + 1*0
Example

An input image (no padding)

A filter (3x3)

Output image (after convolving with stride 1)
### Example

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

An input image (no padding)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

A filter (3x3)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Output image (after convolving with stride 1)
Example

An input image (no padding)

A filter (3x3)

Output image (after convolving with stride 1)
In 3d

```python
model = Sequential()

model.add(Conv2D(filters=4,
                kernel_size=(4,4),
                input_shape=(10,10,3))
```
A RGB image as a 3d **volume**. Each color (or channel) is a layer.
In 3d, our filters have width, height, and depth.
Applied in the same way as 2d (sum of weight * pixel value as they slide across the image).
Applying the convolution over the rest of the input image.
More filters, more output channels.
model = Sequential()

model.add(Conv2D(filters=4,
                   kernel_size=(4,4),
                   input_shape=(10,10,3))

model.add(Conv2D(filters=8,
                   kernel_size=(3,3))
Edges
Shapes
Textures
Exercise

bit.ly/ijcai_1_b

Write a CNN from scratch for CIFAR-10.

Answers: next slide.

Ref: tensorflow.org/beta/tutorials/images/intro_to_cnns
Exercise

bit.ly/ijcai_1b

Write a CNN from scratch for CIFAR-10.

Answers: bit.ly/ijcai_1_b_answers
Would you like to volunteer?
quickdraw.withgoogle.com
Example: transfer learning

bit.ly/ijcai_2

Transfer learning using a pretrained MobileNet and a Dense layer.

Ref: tensorflow.org/beta/tutorials/images/transfer_learning

Ref: tensorflow.org/beta/tutorials/images/hub_with_keras
Example: transfer learning

bit.ly/ijcai_2

Transfer learning using a pretrained MobileNet and a Dense layer.

Answers: bit.ly/ijcai_2_answers
Deep Dream

New tutorial

bit.ly/dream-wip
Image segmentation

Recent tutorial

bit.ly/im-seg
Timeseries forecasting

Recent tutorial
Game 2

Who would like to volunteer?

magenta.tensorflow.org/assets/sketch_rnn_demo/index.html
CycleGAN

Recent tutorial
Under the hood
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
@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
```
@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
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'])
```
Learning more

Latest tutorials and guides

- tensorflow.org/beta

Books

- Hands-on ML with Scikit-Learn, Keras and TensorFlow (2nd edition)
- Deep Learning with Python

For details

- deeplearningbook.org