Applied Deep Learning

Lecture 7 • March 14th, 2019

Announcements

Videos from DevSummit:

https://www.tensorflow.org/alpha/

Version thingy:

- !pip install -q tensorflow==2.0.0-alpha
- You want this (not nightly)

Administrative stuff

Assignments

Assignment #4 released, due 4/11. **Two parts!**

- Real-time text classification (with TF.js). Posting tonight.
- RNNs: predict color names, generate color names. Posting next week.

Part two will be pretty short, I'll share starter code that will get you most of the way there.

Projects

- Feedback uploaded to CourseWorks (cool ideas!)

TLDR

- Part 1) Real-time sentiment analysis in the browser (train a model in Python, deploy using TF.js)

- Part 2) ColorBot with RNNs. Predict color values (e.g., “Orange -> 255, 165, 0). Generate colors and color names (“TensorFlow Orange 255, 150, 10)”

Administrative stuff

Midterm

- In class next next week. You'll have the full period, but shouldn't take more than 90 mins.
- Closed book / notes. Covers everything up through today.
- Short answer written responses + "circle the bug" (on code similar to assignments 1, 2, 3).

Agenda

- Federated learning
- Assignment #4
- TensorFlow.js **Just a year old. Already making a big splash in the ML community.**
- Toxicity detector

Federated learning

Motivation

Your phone has access to an unprecedented amount of data (much of it sensitive / private).

- You carry it almost everywhere (and use it frequently).
- It has powerful sensors (cameras, microphones, GPS).
- It has access to nearly everything you *write* (email, text), *read* (web), *browse* (photos), everyone you contact, etc.

Can we use this data privately?

Models learned on sensitive data could improve usability. E.g.

- **Language models** (speed up text entry by suggesting the next word).

  Quick discussion: why would a language model trained on a dataset like Wikipedia be helpful?

- **Image models** (automatically select “good” photos).

  Quick discussion: how can we get the labels for this?

Can we use this data privately?

Models learned on sensitive data could improve usability. E.g.

- **Language models** (speed up text entry by suggesting the next word).

  Language in SMS is different.

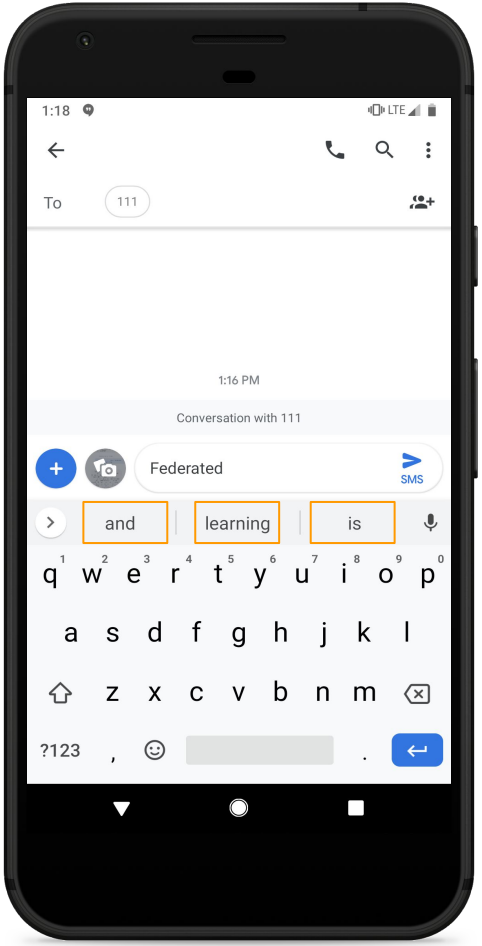
- **Image models** (automatically select “good” photos).

  They can be inferred from user behavior! Can potentially collect a large dataset, with labels that are (on average) accurate, at little cost.

Unfortunately, most ML use cases we’ve seen so far require a centralized training set.

Quick examples

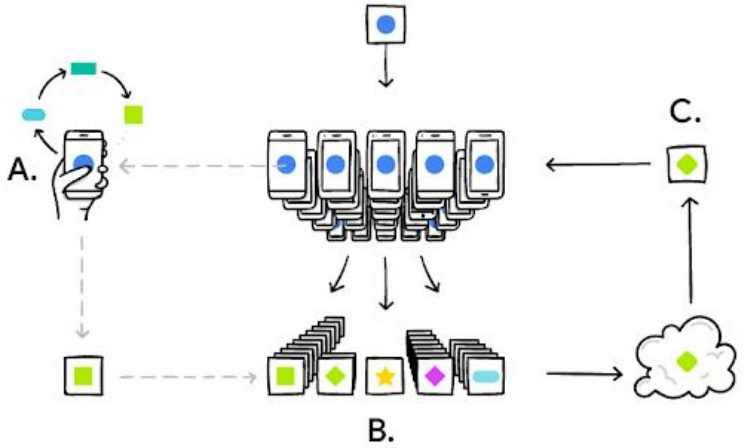
- Actions can indicate whether your model is working well, and/or is useful, and/or is being used!
- Actions can be used to gather more data (e.g., user snaps a bunch of photos, favorites some, and deletes others).
- Next app suggestion.
- [Insert your idea here].



Learning from decentralized data

Goal

- Learn a shared model. Model is updated by a “federation” (collection of participating devices.
- Training data stays **on device**.
- Inference happens locally, **on device**



Communication-Efficient Learning of Deep Networks from Decentralized Data

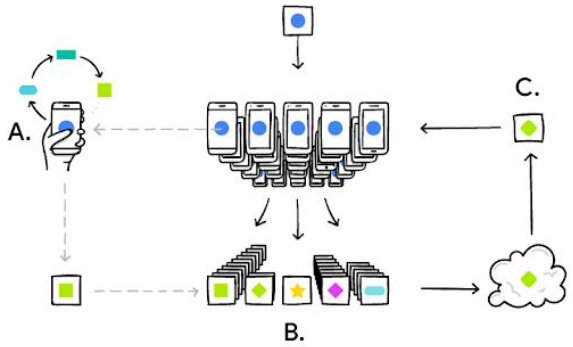
Learning from decentralized data

Server manages a shared model.

1. Server periodically send model (weights) to devices.

2. Devices send **gradient updates** to server.

3. Server averages these gradients, updates the model, and discards them.



Some trust of the server is still required

Greatly improves, but does not guarantee privacy.

Quick discussion: why? Think of both technical (ML) and non ML reasons.

Some trust of the server is still required

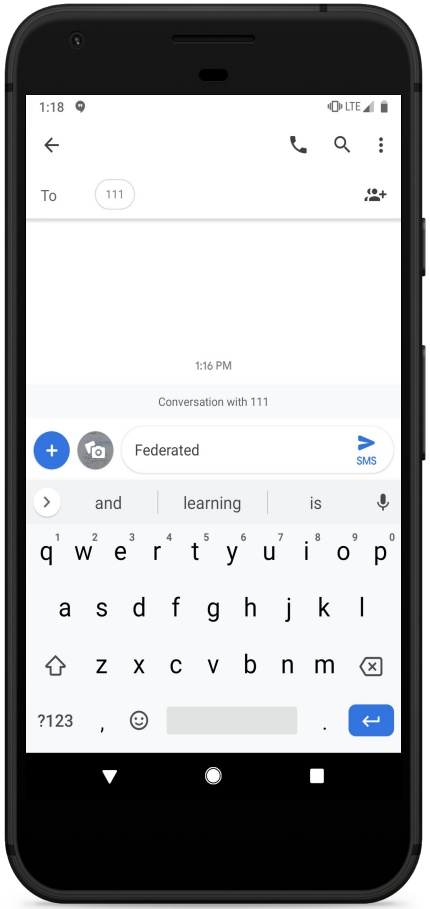
Greatly improves, but does not guarantee privacy.

- Gradient updates are (usually) more difficult to interpret than raw data, but still contain info about the user.
- Server should discard gradient updates after they're applied, but might not.

Leaking info

Quick discussion: can you think of a scenario where sending gradients might leak information?

- Imagine you're building a shared language model.

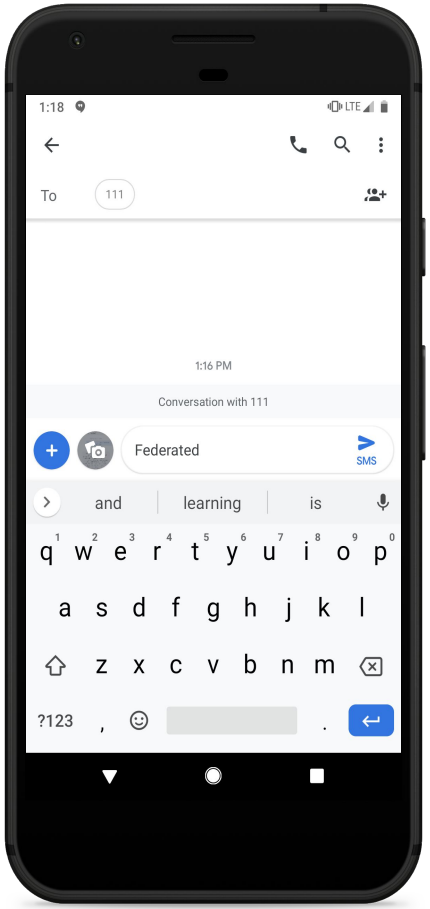


Leaking info

Quick discussion: can you think of a scenario where sending gradients might leak information?

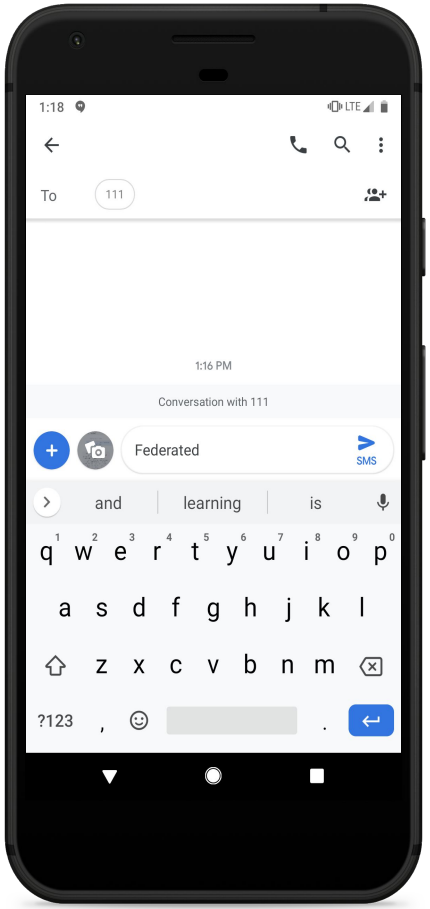
- Imagine you're building a shared language model.

- If the shared language model uses a one-hot encoding at the input layer, the gradient updates sent from the device would reveal the set of words the user wrote!



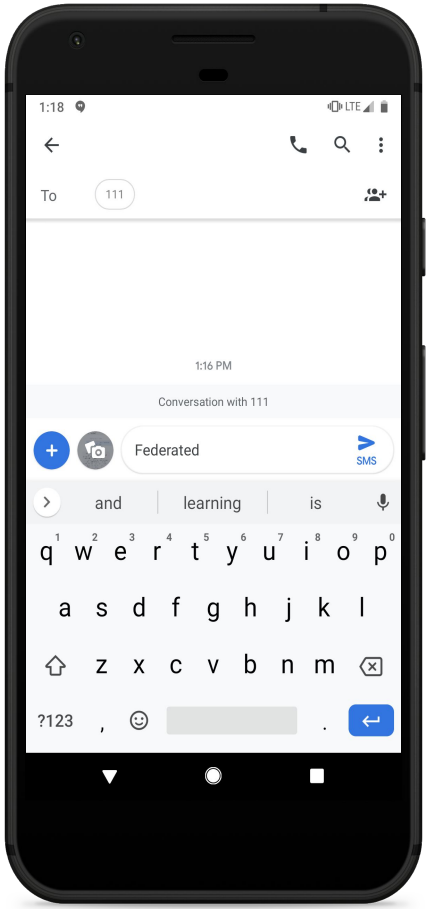
Leaking info

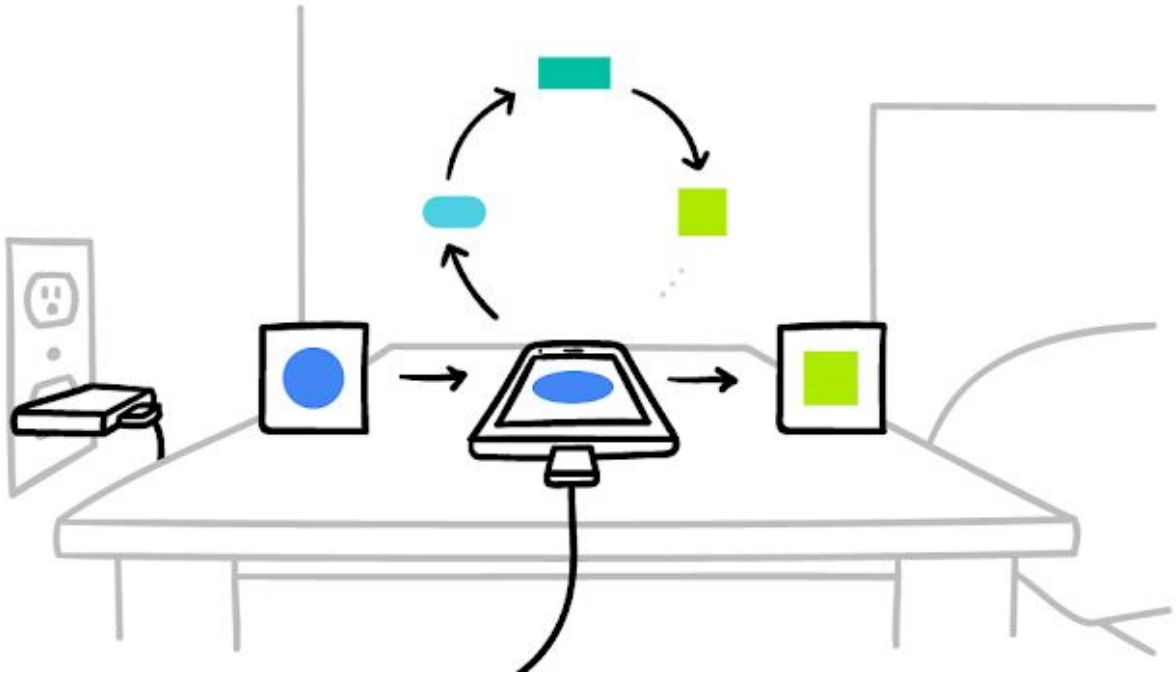
- Less obvious in other scenarios (e.g., the gradients on an image are much less interpretable).



Improving privacy

- Devices add random noise to updates (in aggregate, noise cancels out).
- Device makes a bunch of updates, then sends average gradients.
- Imagine a scheme where gradient updates are encrypted, and cannot be decrypted individually by the server (must be averaged somehow with a large batch of devices).





Notes in case you're new to mobile development. Battery life and heat are key issues, so it's important to perform CPU hungry updates at a time that's convenient for the user.

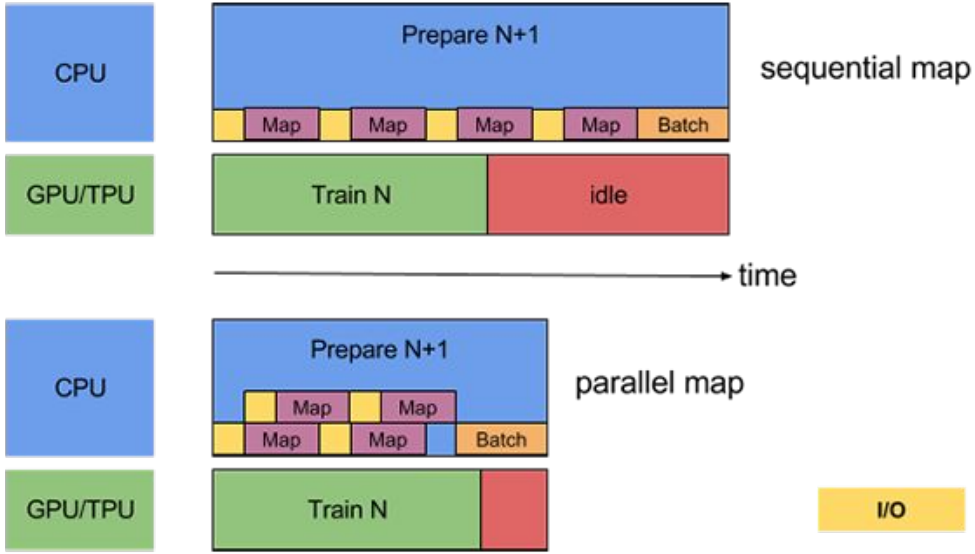
Questions from email

Why use tf.data instead of NumPy for input pipelines? Ex 1: Prefetch.

Without pipelining, the CPU and accelerator sit idle much of the time.

With pipelining, idle time is diminished.

Why use tf.data instead of NumPy for input pipelines? Ex 2: Parallel map.



This one is a little more obvious. Especially helpful when i/o bound (reading images off disk, or over the network), or when data augmentation is compute heavy.

~~Why use tf.data instead of NumPy for input pipelines? Ex 2: Parallel map.~~

- But, most of the time it’s not necessary.
- Start optimizing only when you need to (if at all). Spend that energy elsewhere.

TFRecords? What are they?

tf.Example = a protocol-buffer (think XML, but more efficient) containing a list of ints / floats / strings.

TFRecord = a bunch of tf.Examples serialized to disk

When would you use these?

- Store a large dataset in a machine independent format.
- Do expensive preprocessing once, then save to disk in an efficient format.

Should you use these?

- **Nope** (unless you need to run a large experiment with the same preprocessing many times).

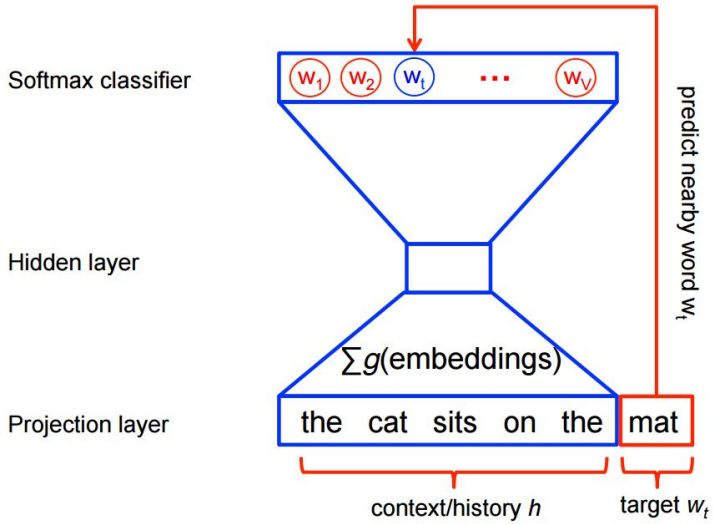
https://www.tensorflow.org/alpha/tutorials/load_data/tf_records

Word2Vec vs example code?

Notes:

- **W2V in a nutshell: train a model to predict the next word in a sentence based on a short context window. In practice, choose the correct word vs. a small number of noisy word candidates for efficiency (rather than having a giant softmax layer where the classifier is forced to select the right word out of all possible choices).**

- **When finished, extract the embedding weights and save / reuse in different tasks down the road, identically to example code.**



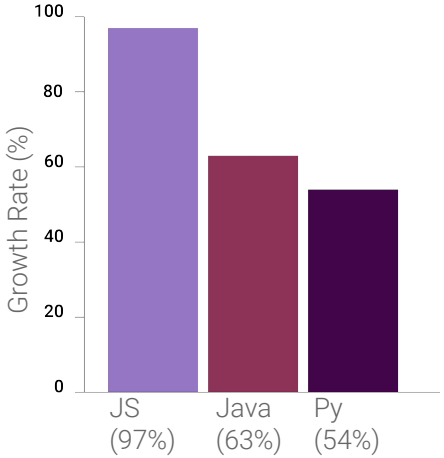
TensorFlow.js

Just a year old, already making a splash

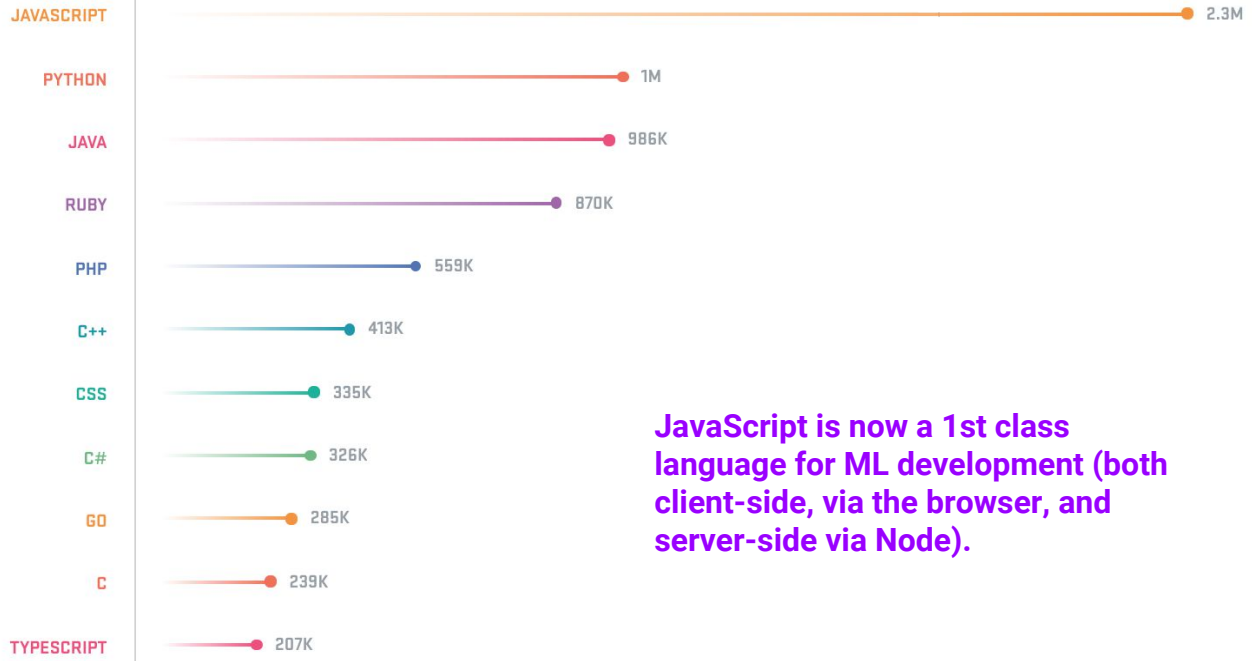
tensorflow.org/js
TensorFlow.js 1.0 (TF Dev Summit '19)
Introducing TensorFlow.js: Machine Learning in Javascript

There's more to life than Python...

JavaScript extends
the deep learning ecosystem
to millions of programmers
and billions of devices.



Number of pull requests in last 12 months on GitHub

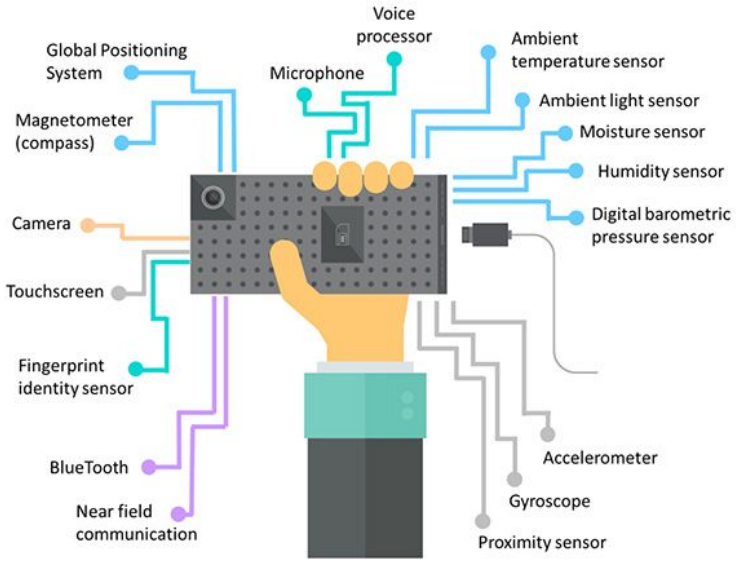


JavaScript is now a 1st class language for ML development (both client-side, via the browser, and server-side via Node).

Source: https://octoverse.github.com/

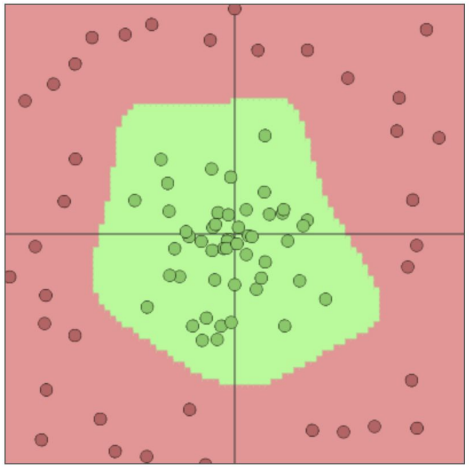
Why in-browser ML

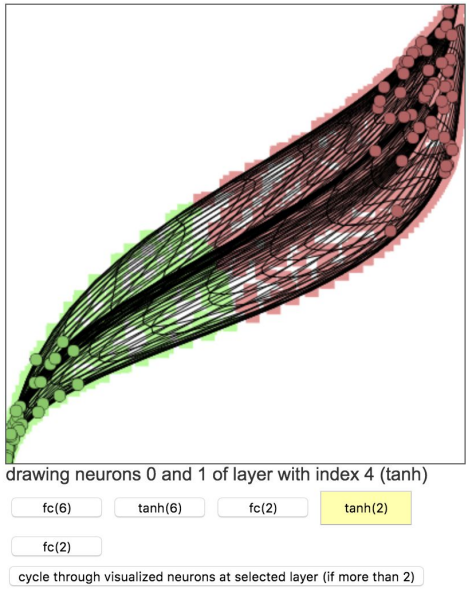
- **No drivers / no installs**
- **Data stays on the client**
- **Interactive**
- **Sensors**
- **Run on desktop, or mobile**
- **GPU acceleration via WebGL**



I bolded my favorites… which ended up being everything. There are a lot of opportunities here.

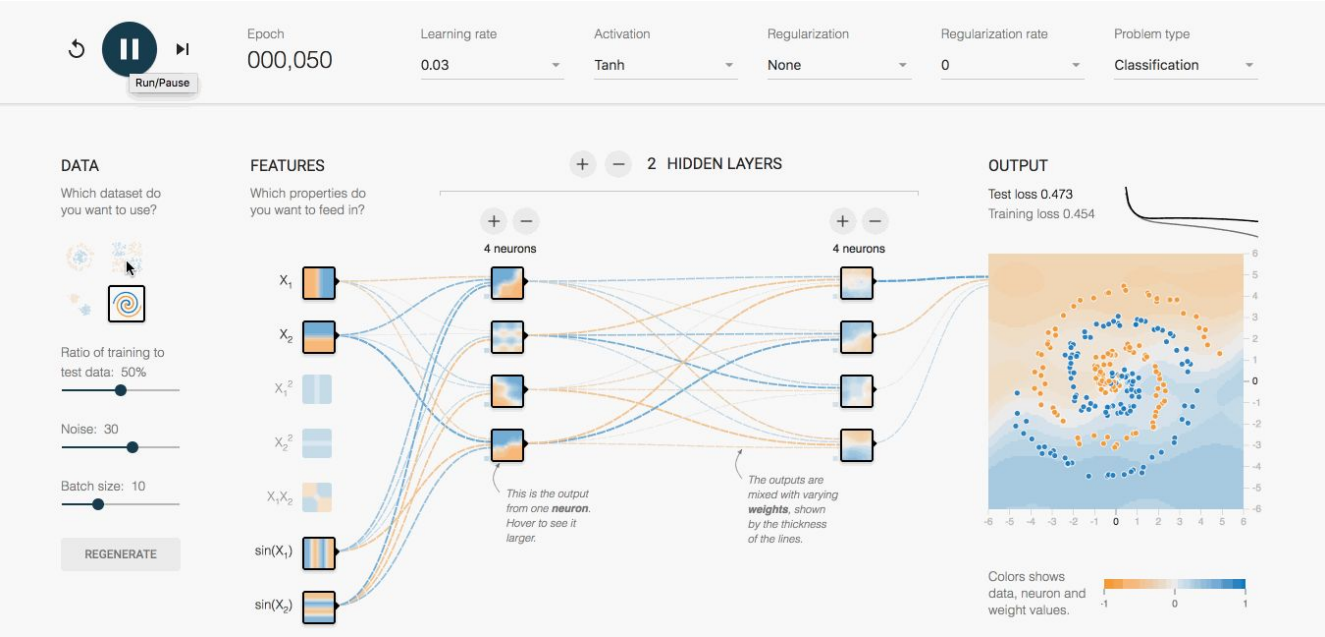
ConvNetJS





ConvNetJS, Demo

TensorFlow Playground

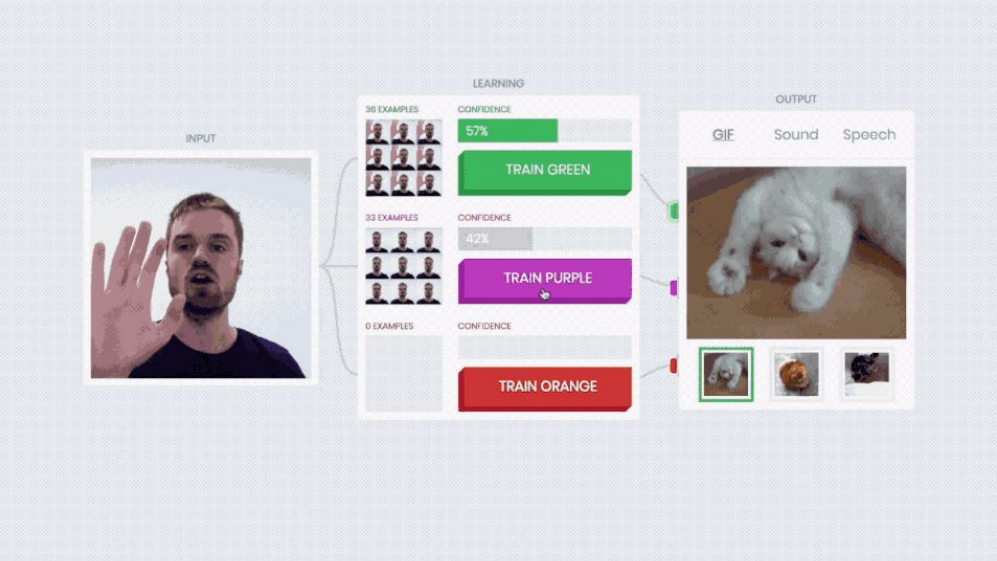


Not powered by tf.js (instead, about 400 lines of JavaScript).

TensorFlow Playground

DeepLearn.js (now TensorFlow.js)

Quick discussion:
why **kNN** on top of a CNN-based model (instead of a Dense layer per usual?)



kNN on top of a pretrained model on ImageNet (haven't checked in a while, probably InceptionV3, or MobileNets if was updated).

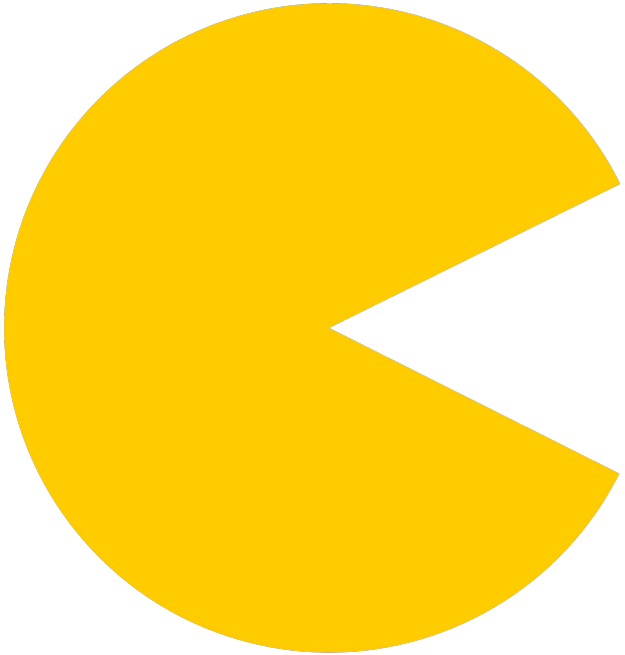
Teachable Machine

PacMan

Transfer learning: trains a dense layer on top of MobileNet activations.

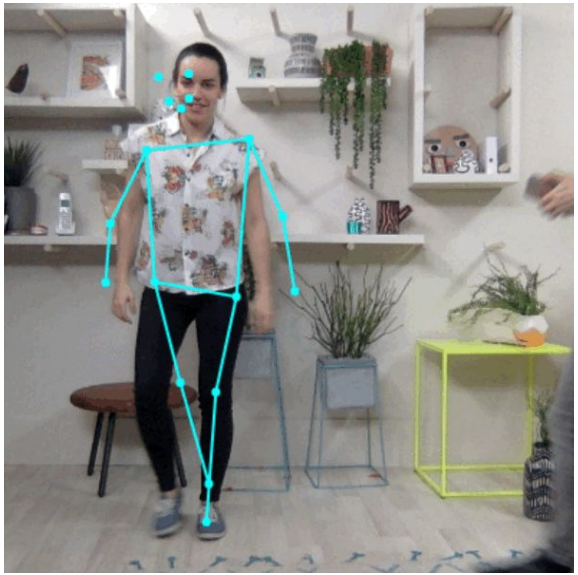
1. **The right MobileNet layer to use was determined experimentally, it's not always the last convolutional one.**

2. **kNN was found to work better with a small amount of data.**



PacMan

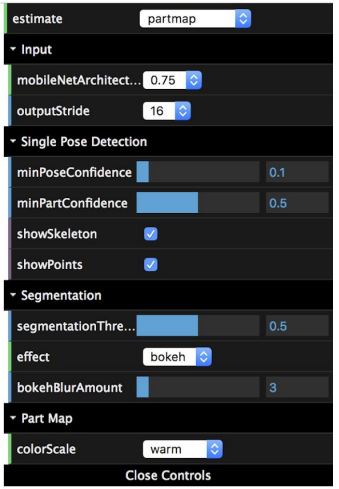
Real-time Human Pose Estimation in the Browser



Naively running at 30 fps! Plenty fast enough to build an application on top of.

PoseNet, Demo

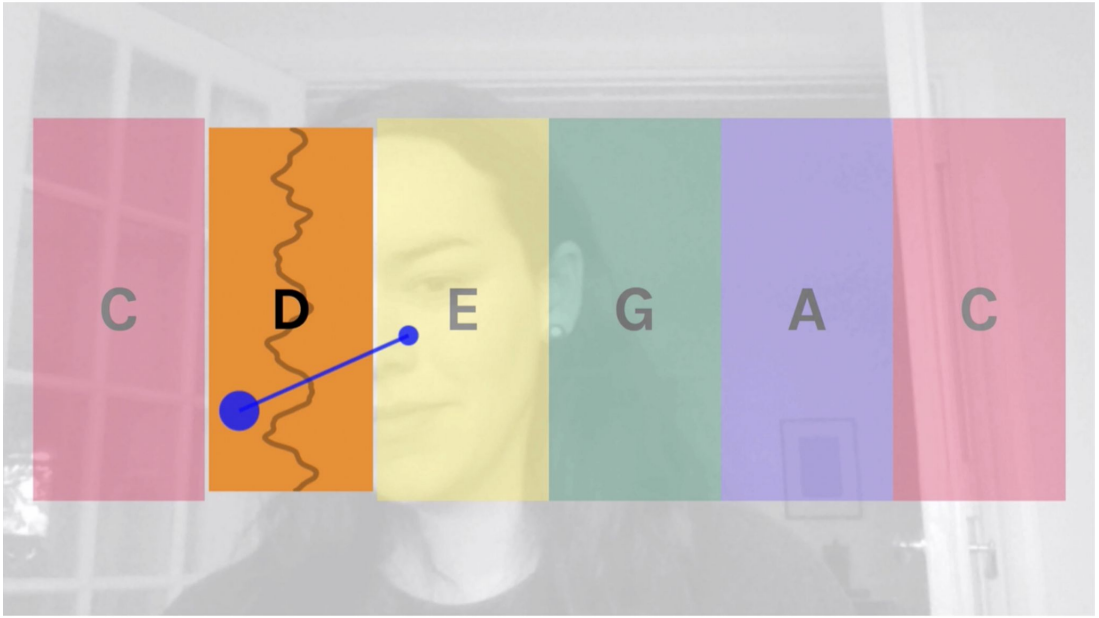
Real-time Image segmentation



[Article / demo]

Naively running at 17 fps! Granted, on an expensive laptop - but hardware prices will drop, and the code will be optimized.

Creativity



This was done with head tilt, but could be done with gaze.

https://experiments.withgoogle.com/collection/creatability

Hello World: start with a simple webpage

```
<html>
<head></head>
<body></body>
</html>
```

Hello World: import tf.js

```
<html>
<head>
 <!-- Load TensorFlow.js -->
 <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs/dist/tf.min.js">
 </script>
```

Hello World: create some training data

```
<!-- y = 2x -1 -->
const xs = tf.tensor2d([-1, 0, 1, 2, 3, 4], [6, 1]);
const ys = tf.tensor2d([-3, -1, 1, 3, 5, 7], [6, 1]);
```

Hello World: define a model

```
const model = tf.sequential();
model.add(tf.layers.dense({units: 1, inputShape: [1]}));

model.compile({
   loss: 'meanSquaredError',
   optimizer: 'sgd'
  });
```

Keras-like API, also eager (imperative) by default.

Hello World: train

```
await model.fit(xs, ys, {epochs: 50});
```

If you're new to JS, this is asynchronous, so we should *await* the return value before proceeding.

Hello World: predict

```
document.getElementById('output_field').innerText =
   model.predict(tf.tensor2d([10], [1, 1]));
```

Complete code

```
<html><head>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs/dist/tf.min.js">
</script>
</head>
<body><div id="output_field"></div></body><script>
 async function learn(){
  const model = tf.sequential();
  model.add(tf.layers.dense({units: 1, inputShape: [1]}));
  model.compile({
   loss: 'meanSquaredError',
   optimizer: 'sgd'
  });
  const xs = tf.tensor2d([-1, 0, 1, 2, 3, 4], [6, 1]);
  const ys = tf.tensor2d([-3, -1, 1, 3, 5, 7], [6, 1]);
  await model.fit(xs, ys, {epochs: 50});
  document.getElementById('output_field').innerText =
   model.predict(tf.tensor2d([10], [1, 1]));
 }
 learn();
</script><html>
```

If you're new to JS, be sure to use the .min versions when you're finished debugging (they're much faster).

Also try https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@latest if you need more debugging info.

New to JS? Setting up an environment to run your code

Three options

1. Open up an HTML file directly in the browser (fine for single files, problems when importing models because of CORS, use #2 instead).

2. Start a simple HTTP Server locally

3. Use a tool like Yarn (see the tf.js demos for instructions)

Start a Simple HTTP Server / serve a directory

```
# If using Python2:
$ python -m SimpleHTTPServer 8000
Serving HTTP on 0.0.0.0 port 8000 ...

# If using Python3:
$ python3 -m http.server
```

https://docs.python.org/2/library/simplehttpserver.html

Quick exercise (5 mins): Run “Hello World”

1. Start a simple HTTP Server
2. Copy the hello world code into a webpage
3. Point your browser at http://localhost:8000

Save and convert a model trained in Python

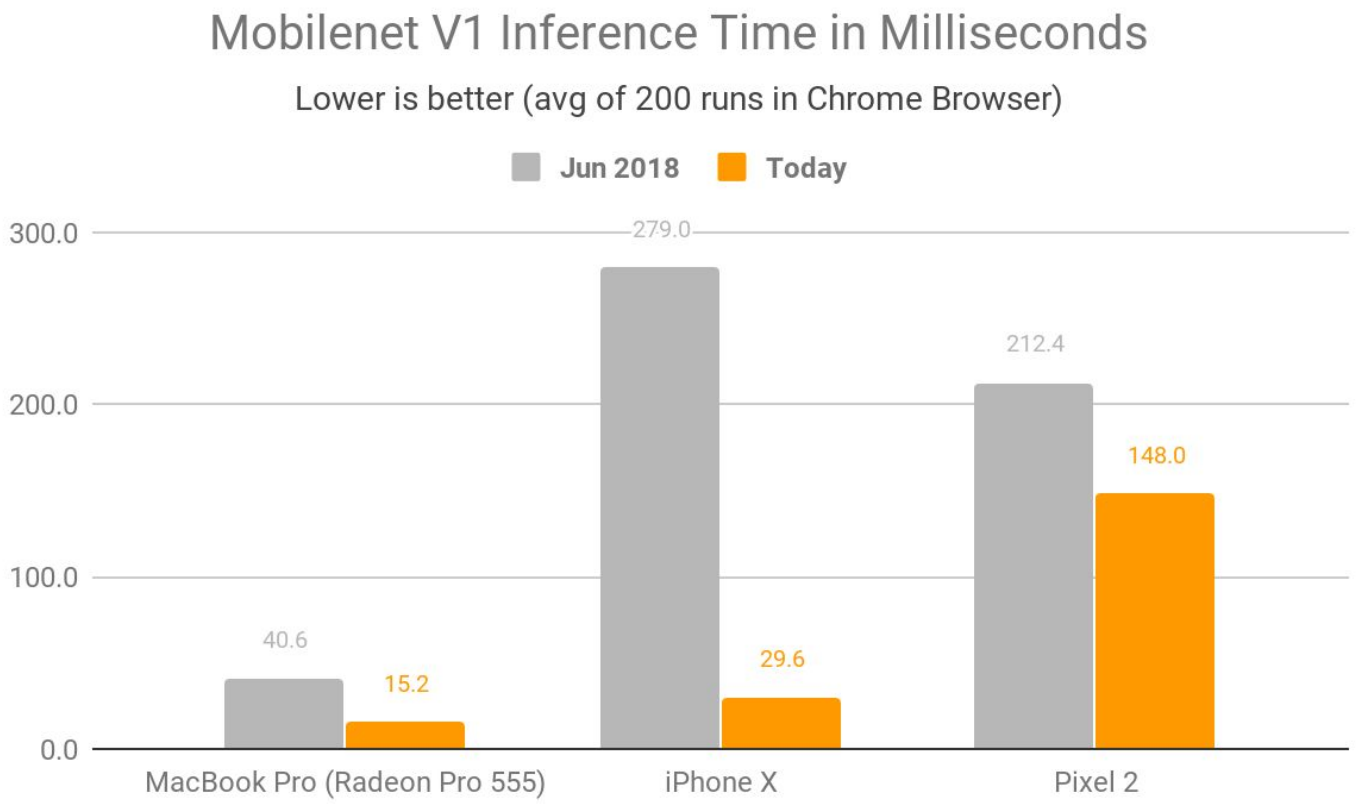
```
# Save in the right format directly
tfjs.converters.save_keras_model(model, FLAGS.artifacts_dir)

# Or, save as usual…
# Then, convert with a script from the command line
model.save('my_model.h5')

$ pip install tensorflowjs
$ tensorflowjs_converter \
    --input_format=keras \
    /tmp/my_model.h5 \
    /tmp/my_tfjs_model
```

Sentiment in the browser

Less exciting (at first glance) than the visual demos, but equally important. Many applications of ML with text in browser.

- **Detecting toxic comments**
- [Your idea here]

Demo

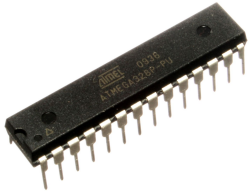


Note: the performance improvement isn't relevant for us - just using this slide for ballpark numbers for inference time in milliseconds.

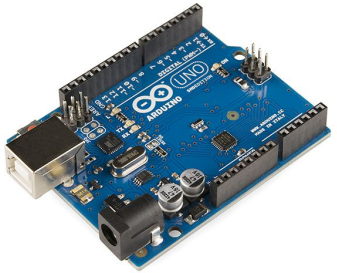
The Future of ML is Tiny

10 MHz, 2KB of ram,
32KB of flash storage.

Doesn’t seem like a lot...
is this enough for useful DL?



ATmega328



Commonly found on an Arduino. Uno

Why the Future of Machine Learning is Tiny

Pretrained models

Imagenet Checkpoints

Classification Checkpoint	MACs (M)	Parameters (M)	Top 1 Accuracy	Top 5 Accuracy
mobilenet_v2_1.4_224	582	6.06	75.0	92.5
mobilenet_v2_1.3_224	509	5.34	74.4	92.1
mobilenet_v2_1.0_224	300	3.47	71.8	91.0
mobilenet_v2_1.0_192	221	3.47	70.7	90.1
mobilenet_v2_1.0_160	154	3.47	68.8	89.0
mobilenet_v2_1.0_128	99	3.47	65.3	86.9
mobilenet_v2_1.0_96	56	3.47	60.3	83.2
mobilenet_v2_0.75_224	209	2.61	69.8	89.6
mobilenet_v2_0.75_192	153	2.61	68.7	88.9
mobilenet_v2_0.75_160	107	2.61	66.4	87.3
mobilenet_v2_0.75_128	69	2.61	63.2	85.3
mobilenet_v2_0.75_96	39	2.61	58.8	81.6
mobilenet_v2_0.5_224	97	1.95	65.4	86.4
mobilenet_v2_0.5_192	71	1.95	63.9	85.4
mobilenet_v2_0.5_160	50	1.95	61.0	83.2
mobilenet_v2_0.5_128	32	1.95	57.7	80.8
mobilenet_v2_0.5_96	18	1.95	51.2	75.8
mobilenet_v2_0.35_224	59	1.66	60.3	82.9
mobilenet_v2_0.35_192	43	1.66	58.2	81.2
mobilenet_v2_0.35_160	30	1.66	55.7	79.1
mobilenet_v2_0.35_128	20	1.66	50.8	75.0
mobilenet_v2_0.35_96	11	1.66	45.5	70.4

Toxicity detector

Curious what you think of this

Lots of interesting issues with releasing a piece of software like this.

Demo (warning: offensive language ahead)

- https://storage.googleapis.com/tfjs-models/demos/toxicity/index.html

Quick discussion: **can anyone think of a good use case for this?**

Curious what you think of this

Lots of interesting issues with releasing a piece of software like this.

Demo (warning: offensive language ahead)

- https://storage.googleapis.com/tfjs-models/demos/toxicity/index.html

Quick discussion: **can anyone think of a good use case for this?**

- Imagine you're a moderator on Wikipedia. One could developed a Chrome or Firefox extension / a tool that helps you quickly flag toxic comments.

- Imagine you run a website where users can comment on articles. Warning users that their comments are inappropriate before they post could be useful.

Curious what you think of this

Lots of interesting issues with releasing a piece of software like this.

Demo (warning: offensive language ahead)

- https://storage.googleapis.com/tfjs-models/demos/toxicity/index.html

Quick discussion: **can anyone think of a good use case for this?**

Quick discussion: **what could go wrong with either of the two ideas?**

Curious what you think of this

Lots of interesting issues with releasing a piece of software like this.

Demo (warning: offensive language ahead)

- https://storage.googleapis.com/tfjs-models/demos/toxicity/index.html

Quick discussion: **can anyone think of a good use case for this?**

Quick discussion: **what could go wrong with either of the two ideas?**

- Bias in training data (could flag non toxic comments as toxic).
- Could focus moderators attention to specific types of comments and away from others)

For next time

Reading

- Deep Learning with Python, Chapter 6 (you'll need RNNs for part two of hw4)
- Communication-Efficient Learning of Deep Networks from Decentralized Data
- Why the Future of Machine Learning is Tiny
- Practical text classification guide

Assignments

- A4 released. Happy to help with questions, especially if you need a hand on the JavaScript side.

Midterm next week

- You'll have the full class period, but shouldn't take more than 90 mins.