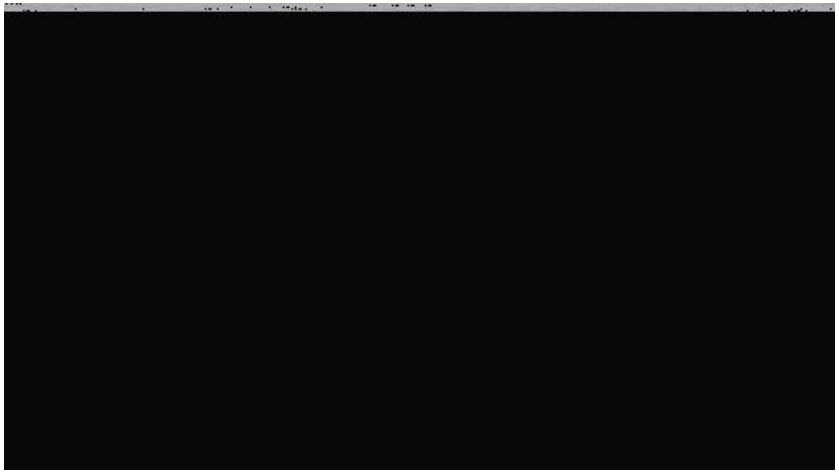
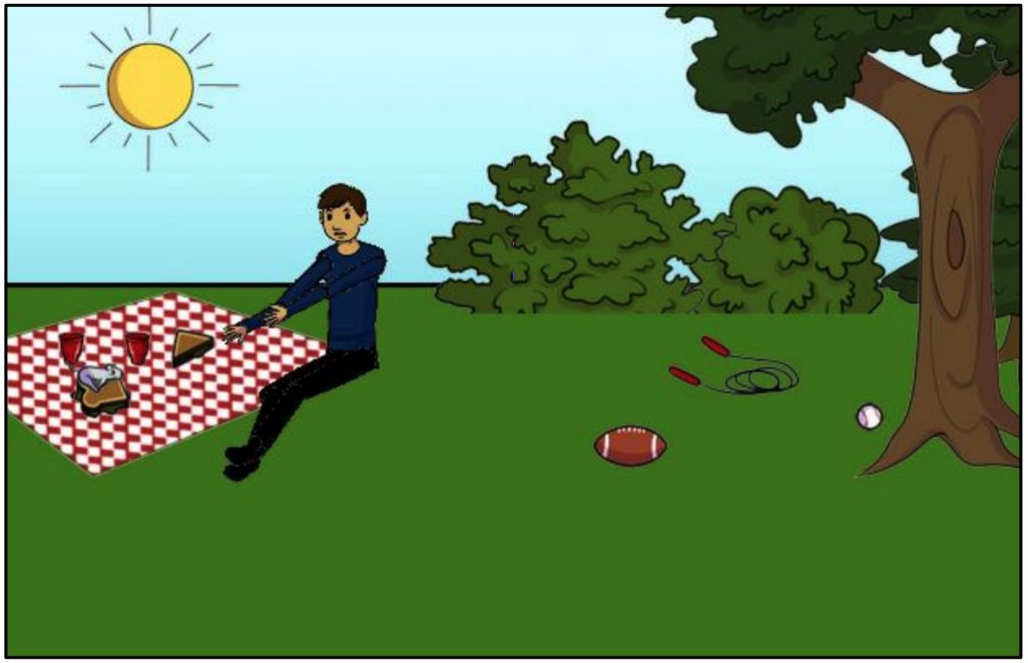




Is this person expecting company?
What is just under the tree?

Applied Deep Learning

Lecture 7 • Oct 17th, 2019

Announcements

Assignments

- Assignment #4 will be Visual Question Answering. I think you'll enjoy it, it's a really cool topic, and a chance to worked with multi-input models with mixed data types. This will be a bit involved, start early and come to office hours if you need a hand.

Projects

- Uploaded to CourseWorks (see the assignments section). If you'd like to do a custom project, please submit a proposal by next week (required, but ungraded - it's just so we can give feedback on your idea).

Announcements

Midterm

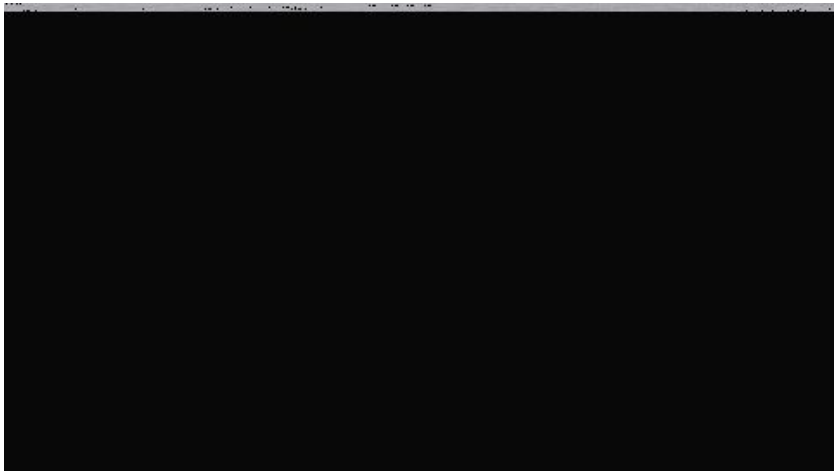
- On 11/7. You'll have the full class period (expect it to take around 90 mins, give or take).

- Review next week. Short answer written responses and "circle the bug" problems (on code similar to assignments 1, 2, 3).

News: Arduino / TinyML

Arduino announced support for TensorFlow Lite Micro. Currently works with the Nano 33 BLE Sense (256KB RAM, 1Mhz, 1MB flash storage), around $30.

Run models on device



Built-in sensors

- Voice: digital microphone
- Motion: accelerometer, gyroscope
- Environmental: temp, humidity, pressure
- Light: brightness, color, object proximity

Examples

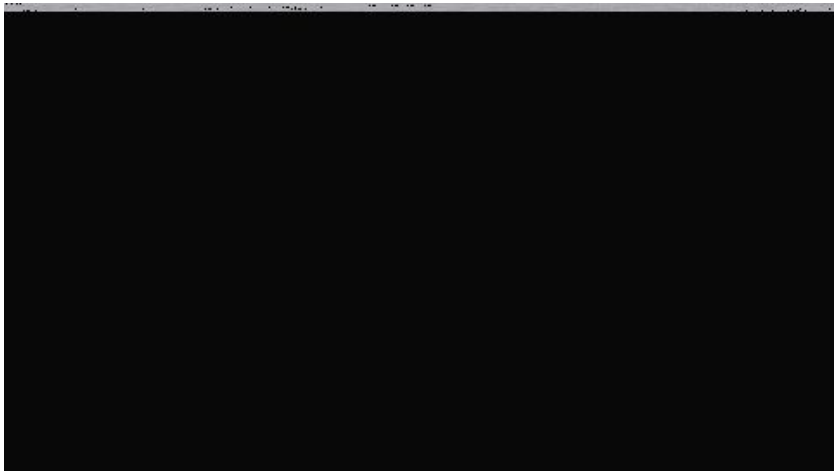
- speech_rec: simple commands
- magic_wand: gesture recognition

You can also attach external cameras.

News: Arduino / TinyML

Arduino announced support for TensorFlow Lite Micro. Currently works with the Nano 33 BLE Sense (256KB RAM, 1Mhz, 1MB flash storage), around $30.

Run models on device



Notes

Early days. Expect **lots** of rough edges (but a promising area for a hackathon or a project).

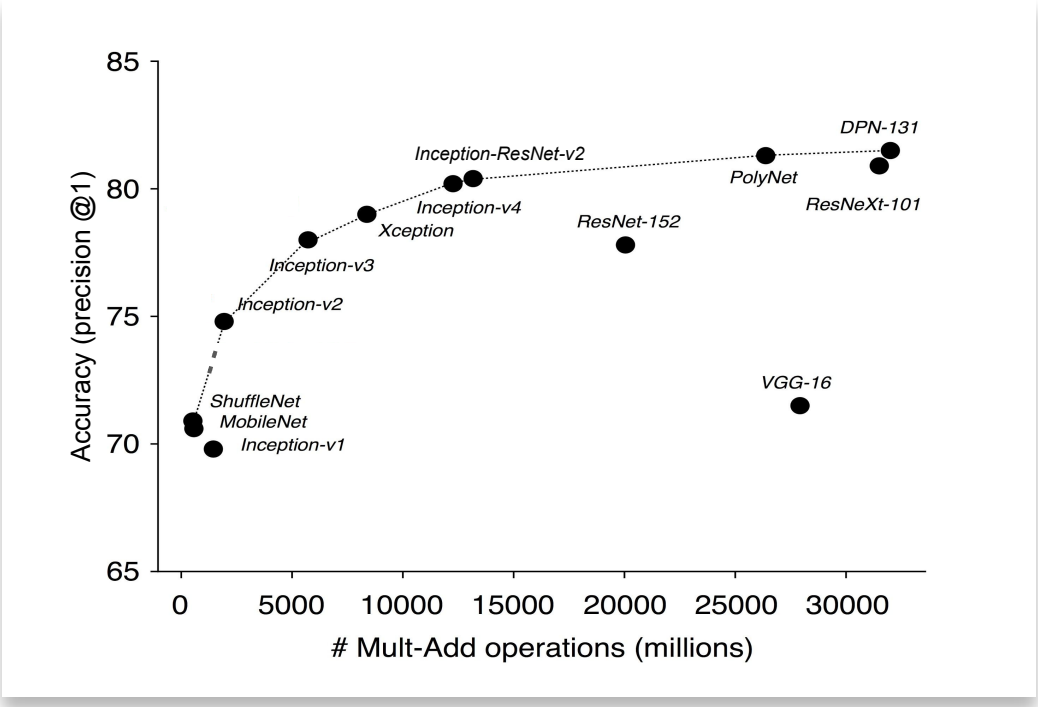
If you'd like some inspiration, check out Why the Future of ML is Tiny, by Pete Warden.

And a new book: TinyML.

News: Neural Architecture Search for video

- From today. Similar to Neural Architecture Search from about a year back.
- Summary: instead of defining architectures by hand, search. Previous work resulted in both more accurate (and smaller / faster) image classifier. This time, for video.
- If you're feeling overwhelmed by the array of famous vision models, this may be good news for the future. Instead of messing with layers, we can spend more time on experimental design (and/or use this technology to discover new layer types and arrangements).
- Slides from last semester:

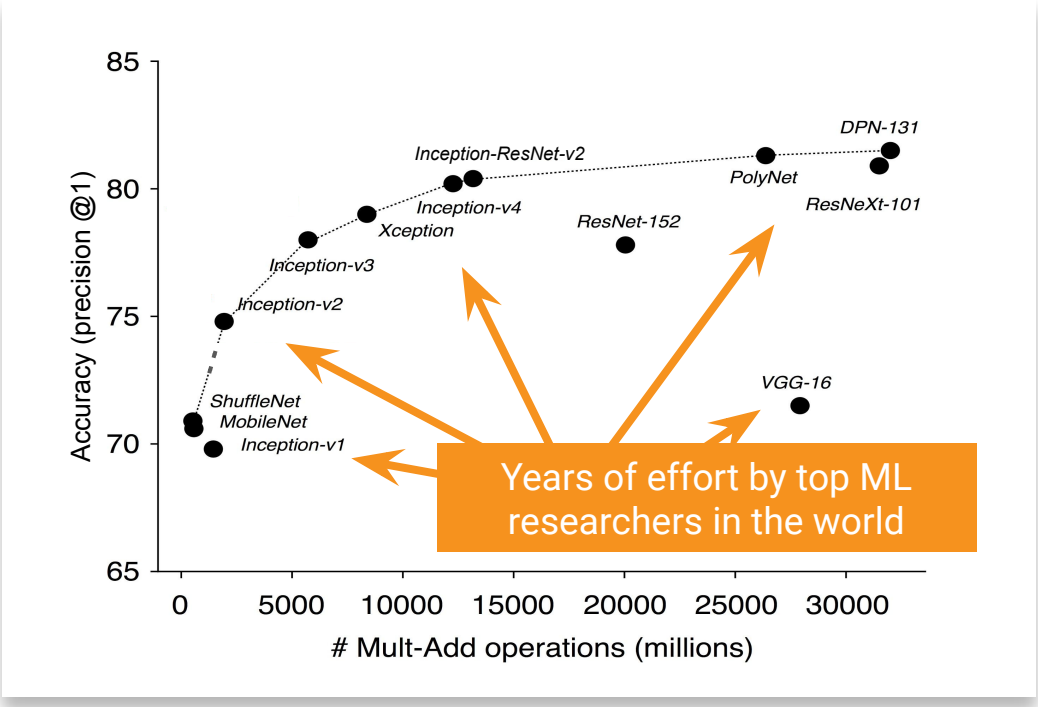
Video Architecture Search



Top performing CNN-based architectures on ImageNet ILSVRC.

[Learning Transferable Architectures for Scalable Image Recognition]

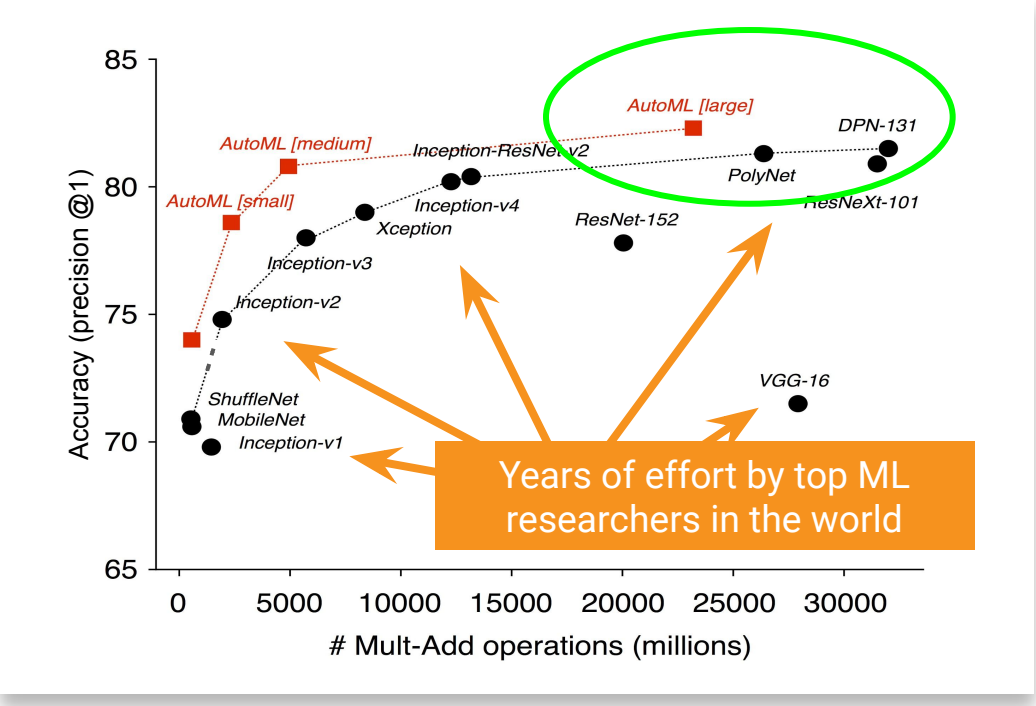
[Keynote (TensorFlow Dev Summit 2018)]



Top performing CNN-based architectures on ImageNet ILSVRC.

[Learning Transferable Architectures for Scalable Image Recognition]

[Keynote (TensorFlow Dev Summit 2018)]

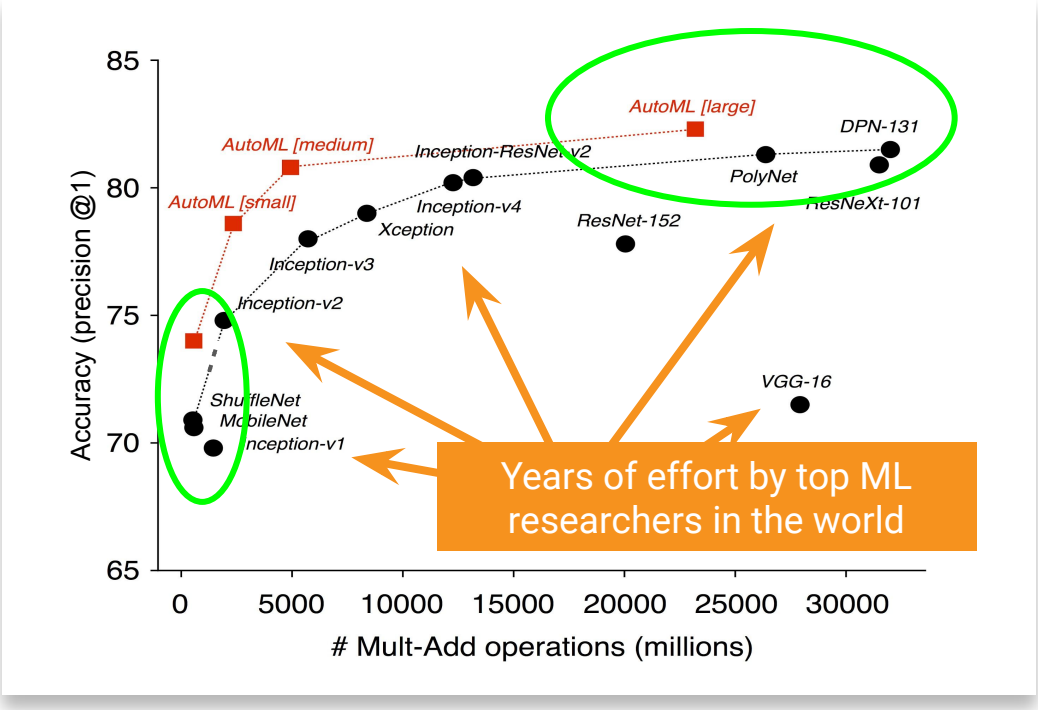


Top performing CNN-based architectures on ImageNet ILSVRC.

Note: AutoML is a brand. What's important are the ideas expressed in the paper re: architecture search.

[Learning Transferable Architectures for Scalable Image Recognition]

[Keynote (TensorFlow Dev Summit 2018)]



Top performing CNN-based architectures on ImageNet ILSVRC.

Note: AutoML is a brand. What's important are the ideas expressed in the paper re: architecture search.

[Learning Transferable Architectures for Scalable Image Recognition]

[Keynote (TensorFlow Dev Summit 2018)]

Agenda

- Visual Question Answering
- DenseNet (thanks Pratik!)
- Break
- Federated learning
- TensorFlow.js

Motivation for multi-input models

- They're important, in addition to being really cool. E.g. in Using Deep Learning to Inform Differential Diagnoses of Skin Diseases, we saw a model that takes both images **and** metadata as input (in that case structured data)

- Of course, more realistic (and powerful) than a model than returns a diagnosis based on an image alone.

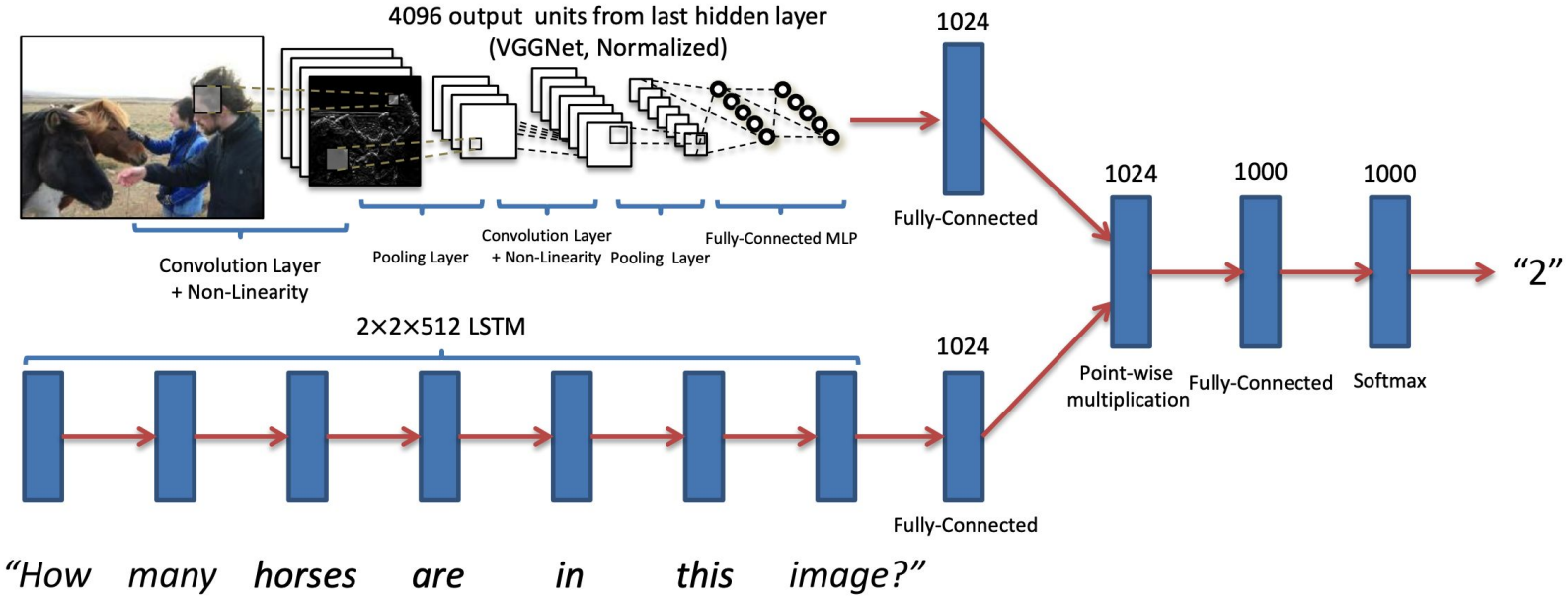
- This is one of the strengths of DL (ability to mix structured and unstructured data in the same model).

How multi-input models work

Dense layers take vectors as input.

- It does not matter if the first half of a vector describes features from an image, and the second half describes structured data (or features from text).

- Features from different data types can be concatenated together, and processed by downstream layers as usual.



VQA: Visual Question Answering, 2016

Visual Question Answering

Visual Question Answering

Is this plane in the air? -> No

Is the bus white? -> No

VQA: Visual Question Answering (2016)

For HW4, you'll work with only **yes/no** questions

Data format

Question, Answer, Image

Is the sky blue?, yes, COCO_train2014_000000393221.jpg

Is there snow on the mountains?, yes, COCO_train2014_000000393221.jpg

Is the window open?, yes, COCO_train2014_000000393223.jpg

Is she brushing?, yes, COCO_train2014_000000393223.jpg

Is the man smiling?, no, COCO_train2014_000000393224.jpg

...

1,000 possible answers (simplified to yes/no for hw4).

What’s the first thing you do with a new dataset?

Look at the data: Images drawn from COCO

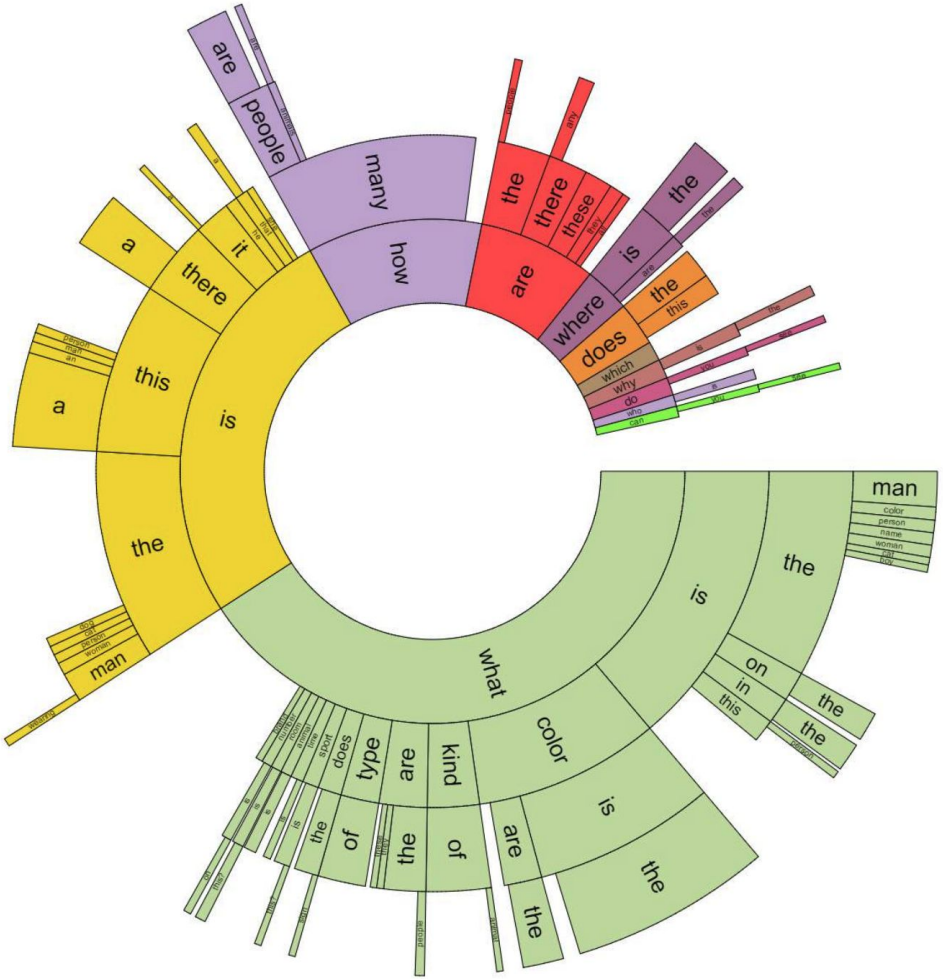
Everyday scenes containing common objects in their natural context.

vs

cocodataset.org

Look at the data: Questions

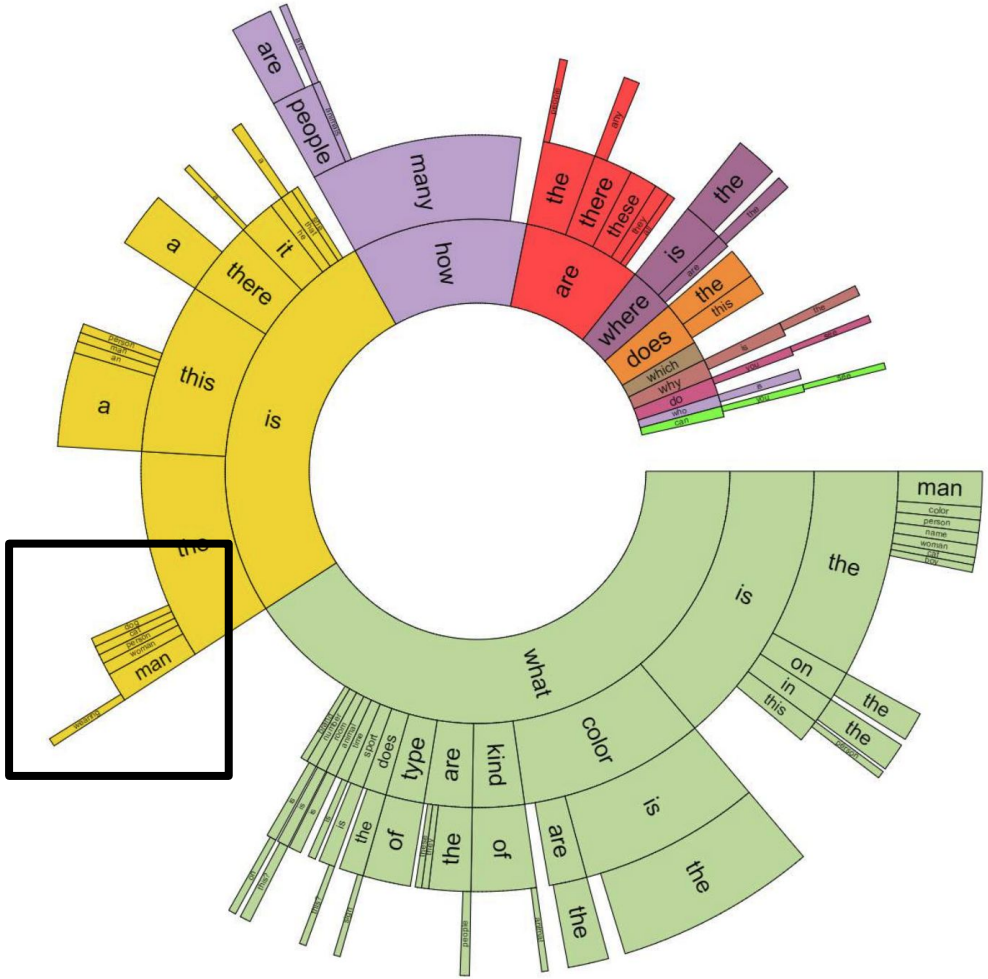
Notice any fairness problems here?



VQA: Visual Question Answering, 2016

Look at the data: Questions

Notice any fairness problems here?



VQA: Visual Question Answering, 2016

Look at the data: Questions

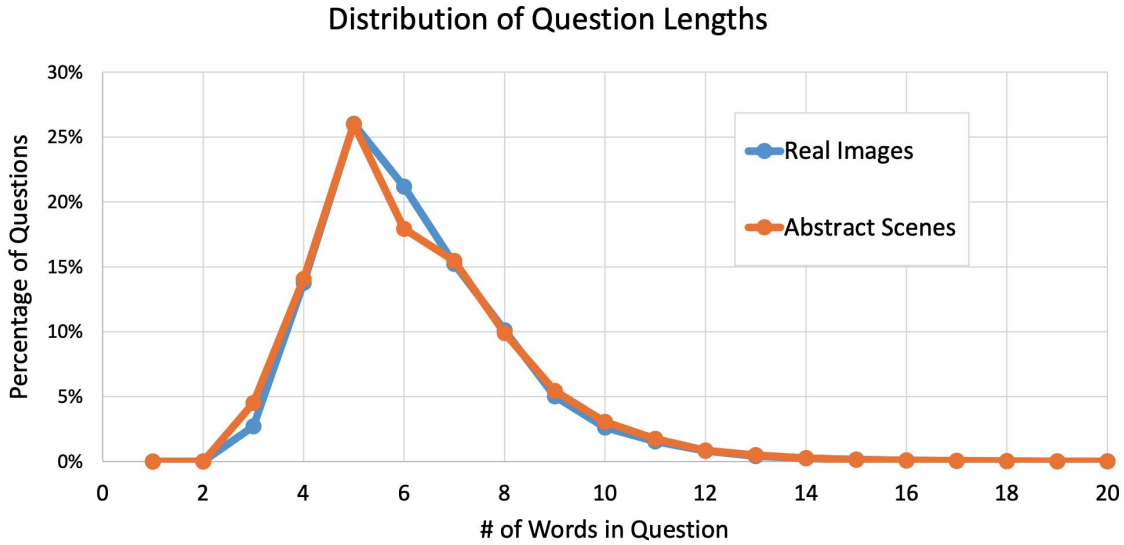
Some unusual questions

Is this cat planning to travel? -> No.

VQA: Visual Question Answering, 2016

Look at the data: Questions

When training your LSTM, you will need to choose a maximum sequence length, balancing between capturing all of the words in each, and speed.



VQA: Visual Question Answering, 2016

Dataset

(2016) v1: +/- 0.25M images, +/- 0.76M questions, and +/- 10M answers

(2017) v2: About twice as large. More importantly, adjusted to increase the importance of the images.

V2 training images are about 13GB

Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering, 2017

Sensible baselines

Quick discussion: Say you're starting on VQA for yes/no questions. What are reasonable baseline models to try, before training a model on images & text?

Sensible baselines

Quick discussion: Say you're starting on VQA for yes/no questions. What are reasonable baseline models to try, before training a model on images & text?

- Predict majority class
- Train a model on images only
- Train a model on text only

Cleaning up the answers

Quick discussion: Say on average each question has about 10 answers. How do you choose the ground truth?

Is the picture blurry?

[{'answer': 'no', 'answer_confidence': 'yes', 'answer_id': 1}, {'answer': 'no', 'answer_confidence': 'yes', 'answer_id': 2}, {'answer': 'no', 'answer_confidence': 'yes', 'answer_id': 3}, {'answer': 'no', 'answer_confidence': 'yes', 'answer_id': 4}, {'answer': 'yes', 'answer_confidence': 'yes', 'answer_id': 5}, {'answer': 'no', 'answer_confidence': 'yes', 'answer_id': 6}, {'answer': 'no', 'answer_confidence': 'yes', 'answer_id': 7}, {'answer': 'yes', 'answer_confidence': 'maybe', 'answer_id': 8}, {'answer': 'no', 'answer_confidence': 'yes', 'answer_id': 9}, {'answer': 'no', 'answer_confidence': 'maybe', 'answer_id': 10}]

Cleaning up the answers

Quick discussion: Say on average each question has about 10 answers. How do you choose the ground truth?

- Same ideas as lecture 4 (best answer depends on the domain, here - majority vote is sensible).

- If this was medical data (or something else important), you'd want to do adjudication, or collect more answers, or discard noise, etc.

Other nice properties of this dataset

Automatic evaluation

- Softmax output / can phrase the correct answer of 1 of 1,000 choices
- Much simpler than evaluating free text output (NMT, or image captioning).

How good are people at this task?

Dataset	Input	All	Yes/No	Number	Other
Real	Question	40.81	67.60	25.77	21.22
	Question + Caption*	57.47	78.97	39.68	44.41
	Question + Image	83.30	95.77	83.39	72.67
Abstract	Question	43.27	66.65	28.52	23.66
	Question + Caption*	54.34	74.70	41.19	40.18
	Question + Image	87.49	95.96	95.04	75.33

VQA: Visual Question Answering, 2016

Algorithms

Approach	All	Yes/No	Number	Other
Prior	25.98	61.20	00.36	01.17
Language-only	44.26	67.01	31.55	27.37
d-LSTM+n-I [24]	54.22	73.46	35.18	41.83
MCB [9]	62.27	78.82	38.28	53.36

Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering, 2017

Tips and demo

Tips (performance)

- Use Checkpoints (save weights after each epoch to Drive, in case Colab disconnects)

- Use Transfer learning (and **cache** the result to disk!). It will be too slow to train your model if you need to forward each image through a model like Inception again, and again, and again. Instead (before training) forward all your images once, and cache the activations to disk (in case they are too large to keep in memory).

Notes: The Image Captioning tutorial has an older (but working) example. We would write this a bit differently today (on our list), but fine to use for HW4 - will save a bunch of time.

Tips (correctness)

Fit a single batch first (the tips from this article are great)

- Do not try to train a large model for one epoch and hope that it works =D Instead, train on a single batch of data repeatedly, and verify you can memorize the data (loss goes to zero). Use model.train_on_batch()

After fitting on a batch, make sure your model's predictions are sensible (display an image, the ground truth, and your models output).

Small models are fine.

- The example on keras.org is unnecessarily large for HW4. Use a much smaller model.

Tips (cont'd)

- You are better off **starting with a small, simple model that trains fast**, so you can verify loss is decreasing, and you're able to making increasingly sensible predictions.

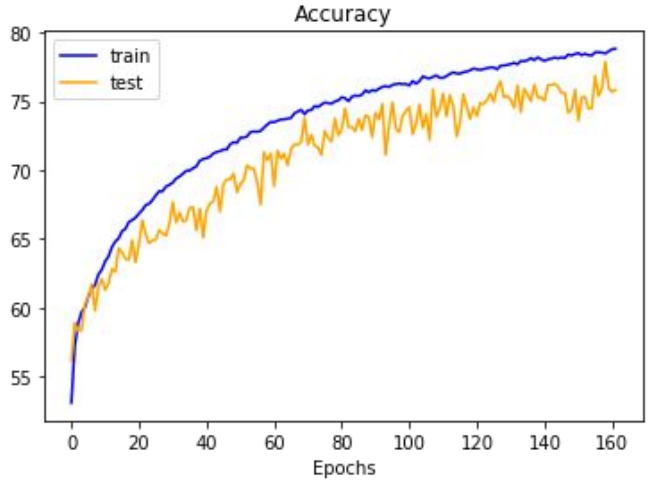
- It will probably take you longer to set up your infrastructure (code to download and preprocess data, save activations to disk, produce training / loss plots, display predictions on images, etc) than it will to define your model.

- Use a small vocabulary size to start (say, 2000 words) and a small sequence length (say, 10).

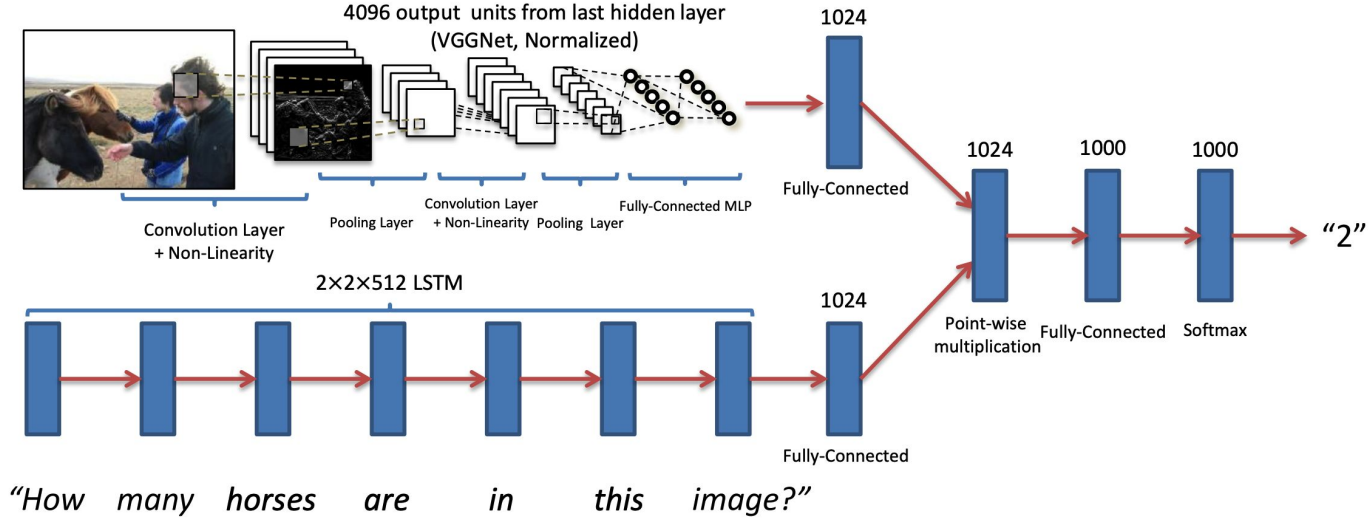
- Use Global average pooling (instead of a flatten), after retrieving the activations from a pretrained CNN. Remember that Dense layers are expensive, reduce the number of weights that must be learned.

Tips (cont'd)

- If your validation set is large, it will be slow to evaluate on it every epoch. Instead, evaluate either every +/- 5, or - select a random, smaller validation set to validation on as you go. If you do, you may see plots like this (bumpiness results from different validation images at each step).



Demo, Functional API



VQA: Visual Question Answering, 2016

DenseNet

What will you learn today?

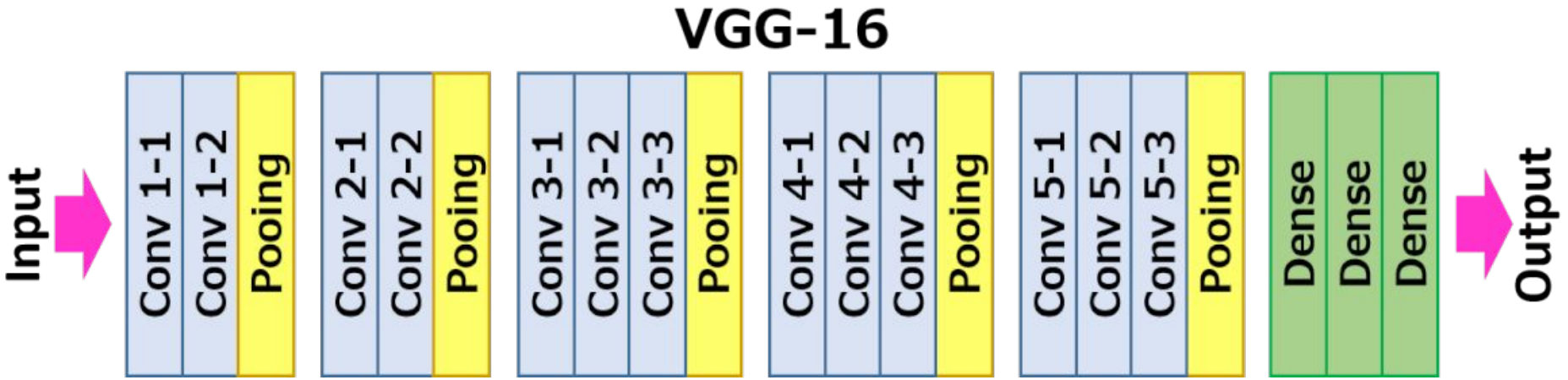
- Prehistoric architecture (circa 2016)
- DenseNet
- Implementation
- Results
- Tutorial!

DenseNet

- DenseNet or ‘Densely Connected Convolutional Networks’ is a convolutional neural network architecture,
- Connects *each* layer to *each other* layer in a feed-forward manner.
- Won the ‘Best Paper Award’ at the Computer Vision and Pattern Recognition (CVPR) Conference in 2017.

Densely Connected Convolutional Networks.

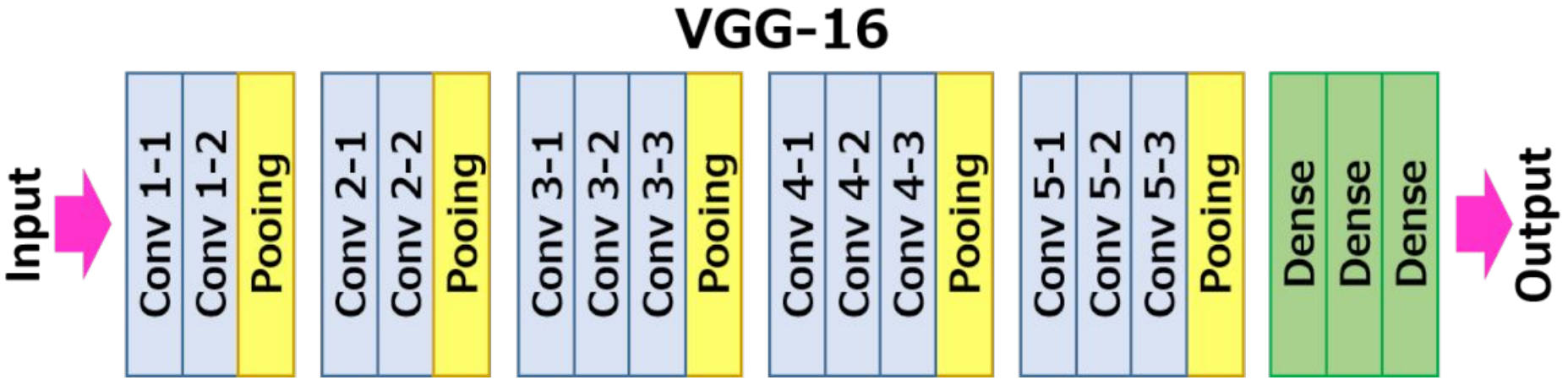
Looking Back at VGG



What are the number of parameters in VGG-16?

[Very Deep Convolutional Networks for Large-Scale Image Recognition]

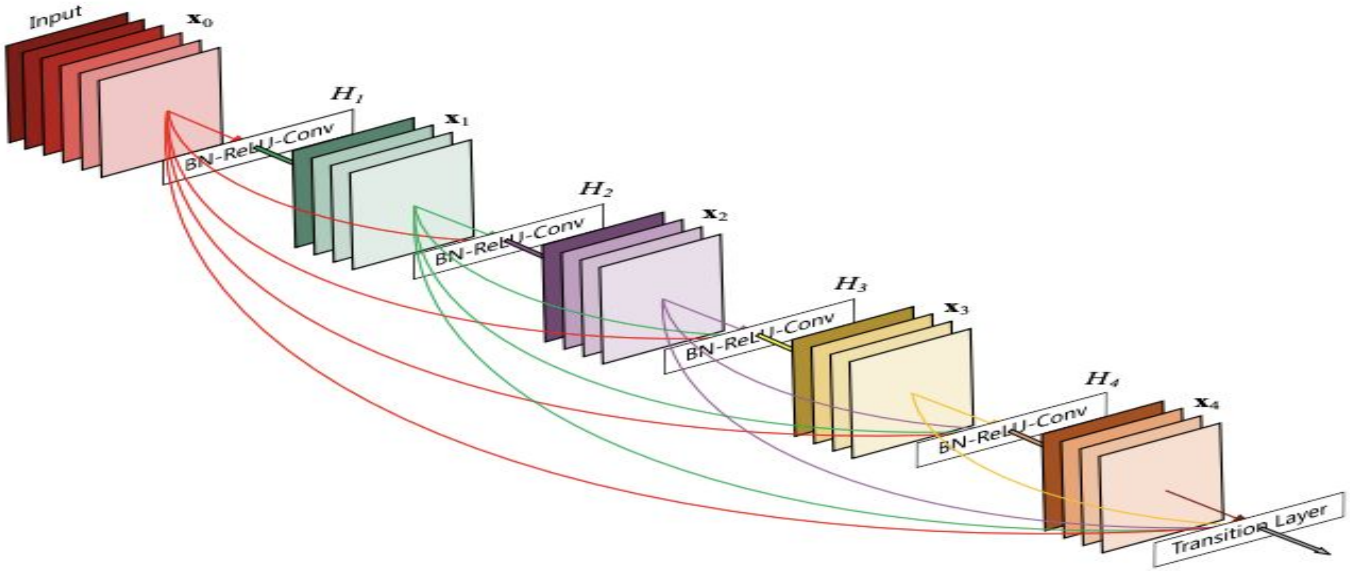
Looking Back at VGG



What are the number of parameters in VGG-16? **~138 Million.**

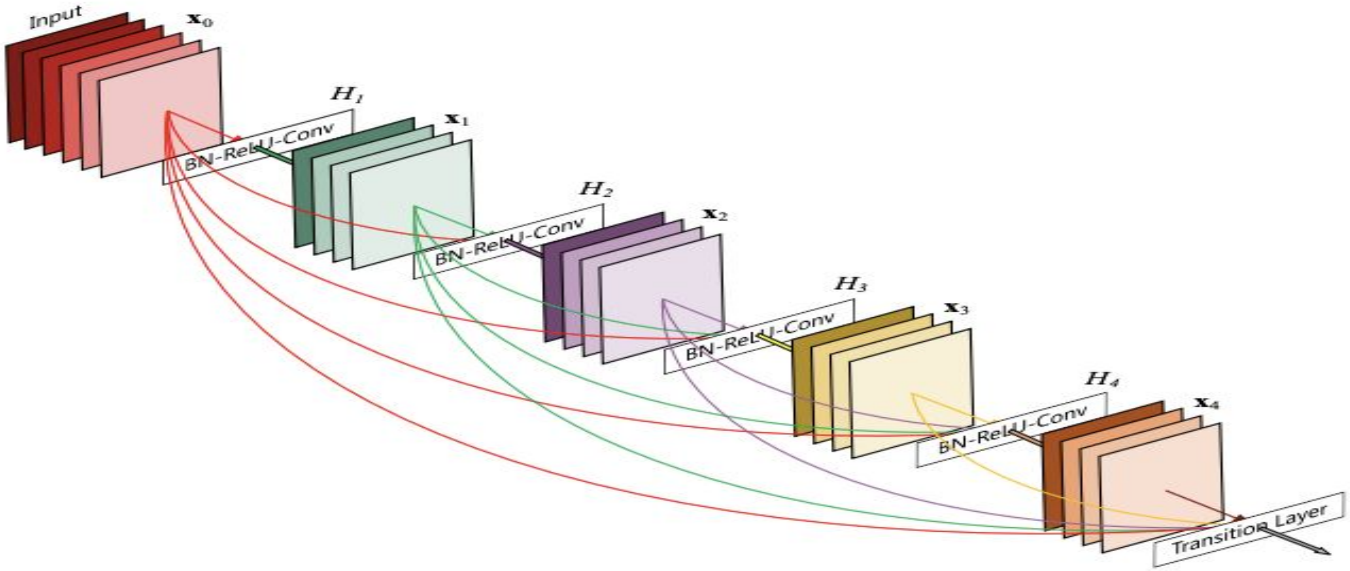
[Very Deep Convolutional Networks for Large-Scale Image Recognition]

Back to DenseNet



Densely Connected Convolutional Networks.

Back to DenseNet



The number of parameters in a 40 layer DenseNet is ~ **1 million**.

Densely Connected Convolutional Networks.

DenseNet Architecture

- Does *not* have a fully connected layer!
  - At the end, there's a GlobalAveragePooling followed by a softmax
  - Josh has a demo in the VQA code
- Architecture consists of three main components:
  - Convolution Block
  - Dense Block
  - Transition Block

Architecture

- Global Average Pooling (GAP) layer is used before classification layer.
- GAP layer transforms input of dimensions *(h, w, c)* to *(1, 1, c)*.
- Using a GAP layer in place of a FC layer *significantly* reduces the number of parameters.

Architecture - Convolution Block

```
def conv_block(x, input_shape):
  x = tf.keras.layers.BatchNormalization(input_shape=input_shape)(x)
  x = tf.keras.layers.Activation('relu')(x)
  x = tf.keras.layers.Conv2D(filters=growth_rate, kernel_size=3, strides=1,
                             padding='same', kernel_initializer='he_normal')(x)
  return x
```

Architecture - Dense Block

```
def dense_block(x, num_conv_layers):
  concat_feat = x
  for ctr in range(num_conv_layers):
    x = conv_block(concat_feat, concat_feat.shape[1:])
    concat_feat = tf.keras.layers.concatenate([concat_feat, x])
  return concat_feat
```

Architecture - Transition Block

```
def transition_block(x):
  x = tf.keras.layers.BatchNormalization()(x)
  x = tf.keras.layers.Activation('relu')(x)
  x = tf.keras.layers.Conv2D(filters=x.shape[-1], kernel_size=1, strides=1,
                             kernel_initializer='he_normal')(x)
  x = tf.keras.layers.AveragePooling2D(pool_size=2, strides=2)(x)
  return x
```

Advantages

- It alleviates the vanishing gradient problem.
- Strengthens feature propagation.
- Encourage feature reuse.
- *Substantially* reduce the number of parameters.

Densely Connected Convolutional Networks.

Results

DenseNet obtained state-of-the-art results in 2017 on:

- CIFAR-10 - 3.46% error rate
- CIFAR-100 - 17.18% error rate
- SVHN - 1.59% error rate

Colab Tutorial
Notebook is on CourseWorks

Federated learning

Motivation

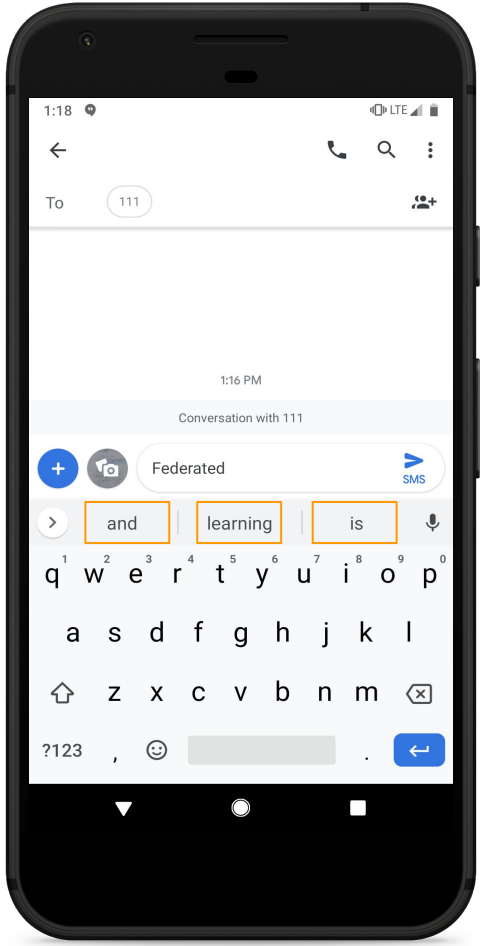
Your phone has access to an unprecedented amount of data (much of it sensitive / private).

- 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...

Models learned on sensitive data could be useful

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

- Quick discussion: why not use a language model trained on Wikipedia?



What else could you do?

Next app suggestion (easy)

Automatically suggest “good” photos (harder)

- Quick discussion: how could you infer labels for this at scale?

Problem

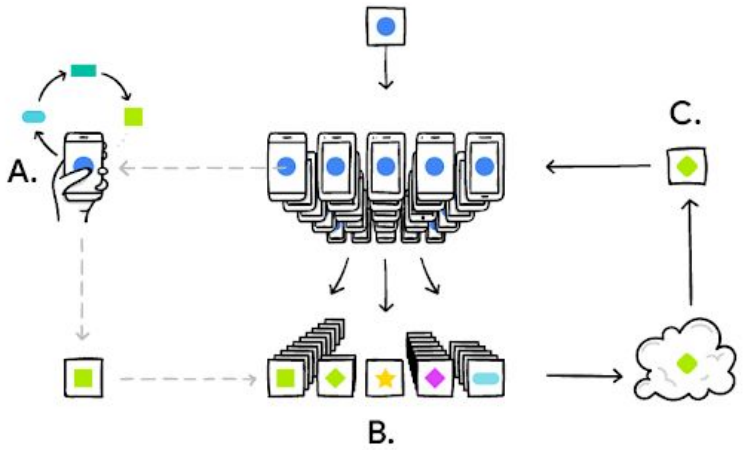
Training usually requires a centralized training set.

- We’re not interested in uploading a bunch of sensitive data from our phones to a server.

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, 2017

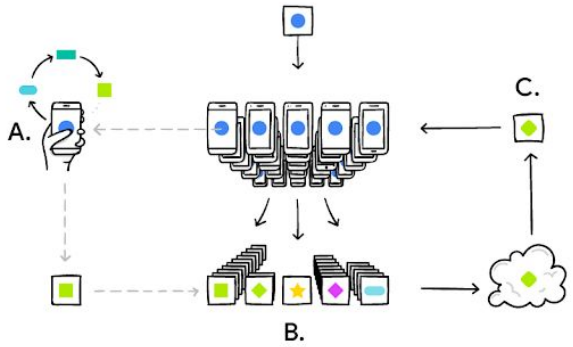
Learning from decentralized data

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.

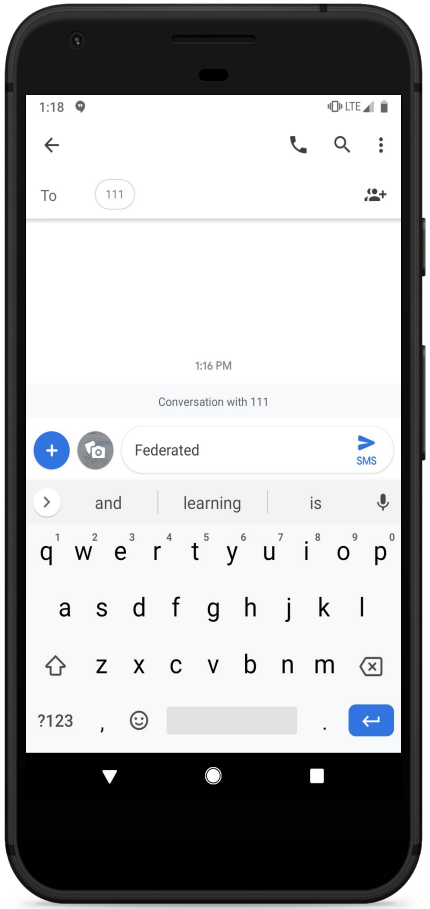


Communication-Efficient Learning of Deep Networks from Decentralized Data, 2017

Leaking info

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

- Imagine you're building a shared language model.

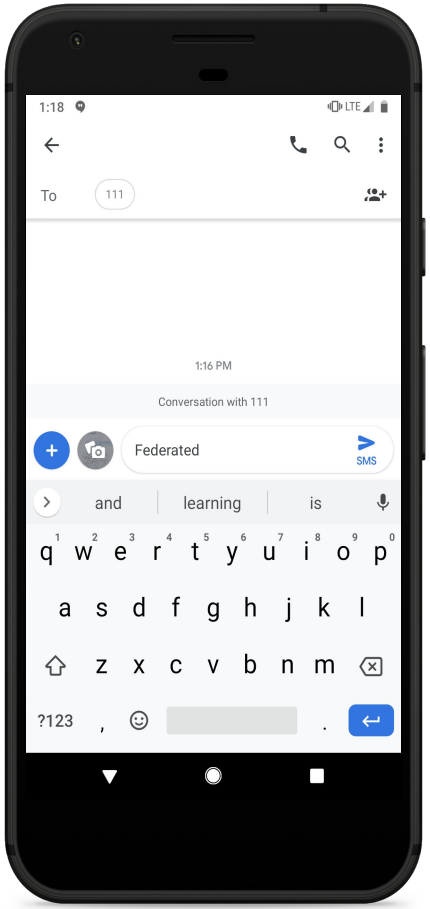


Leaking info

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.



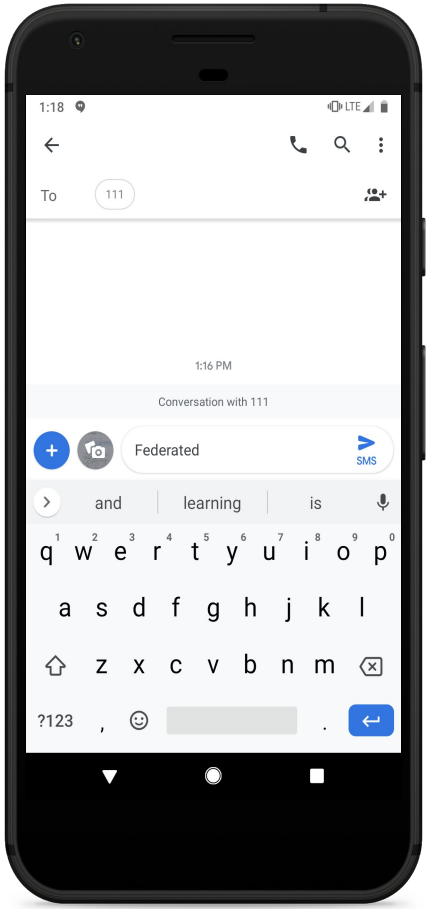
Some trust of the server is still required

Greatly improves, but does not guarantee privacy.

- Gradient updates are more difficult to interpret than raw data, but still contain info about the user.
- Server should discard gradient updates after they’re applied.

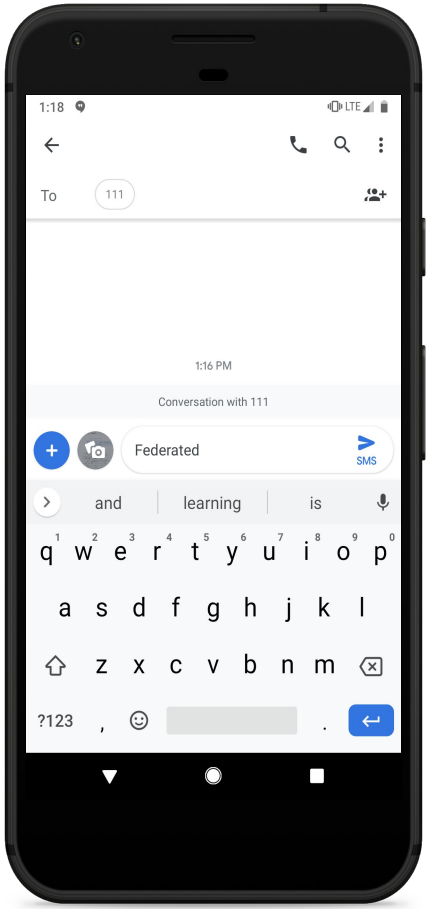
Improving privacy further

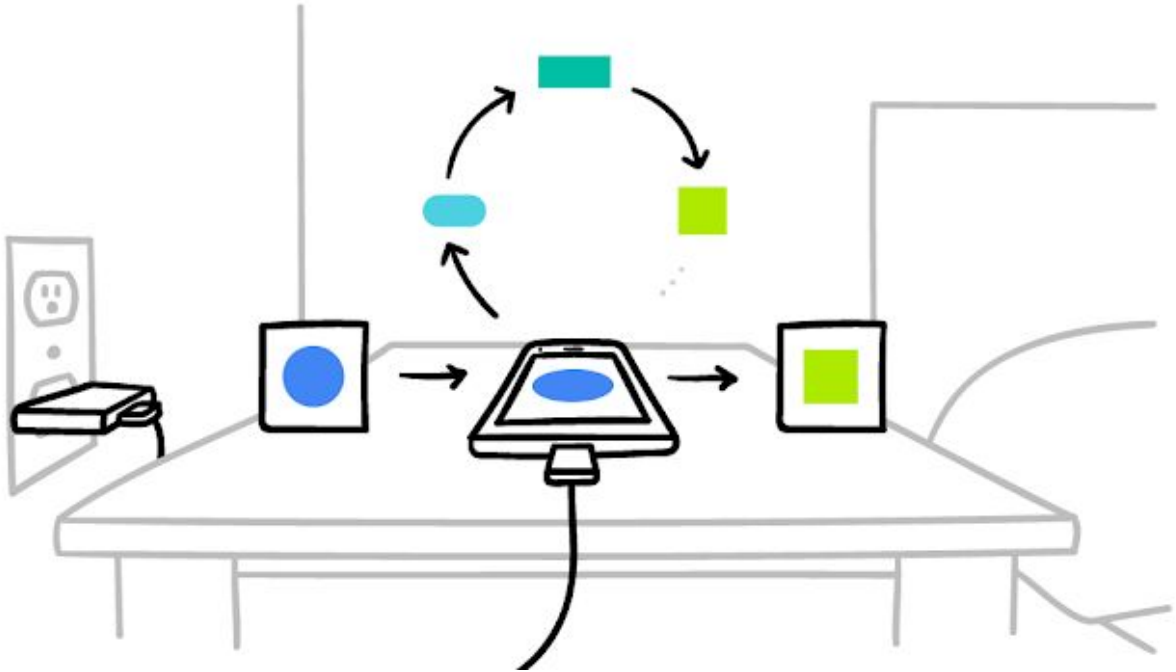
- Devices add random noise to updates (in aggregate, noise cancels out).
- Device makes a bunch of updates, then sends average gradients to server.
- Imagine a scheme where gradient updates are encrypted, and cannot be decrypted individually by the server (must be averaged somehow with a large batch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



Simple example

- We can calculate our average age without anyone revealing their age to the class.





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. This is also one of the reason why we usually don't train models on-device.

Questions from email

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.
- Reading large, contigious files is faster than random access to a bunch of small ones

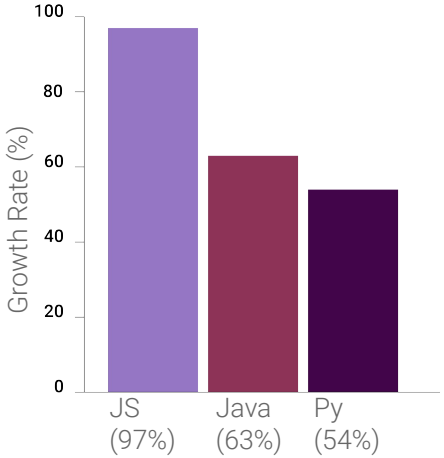
Should you use these?

- **Only if you're working on a large project, or research where you can invest extra engineering time**

TensorFlow.js

There's more to life than Python

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



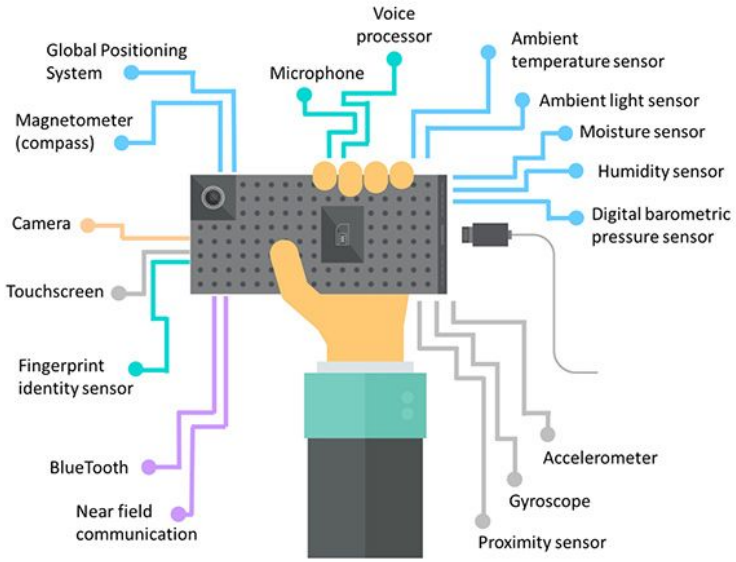
Number of pull requests in last 12 months on GitHub

TensorFlow.js now also supports Node.js

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

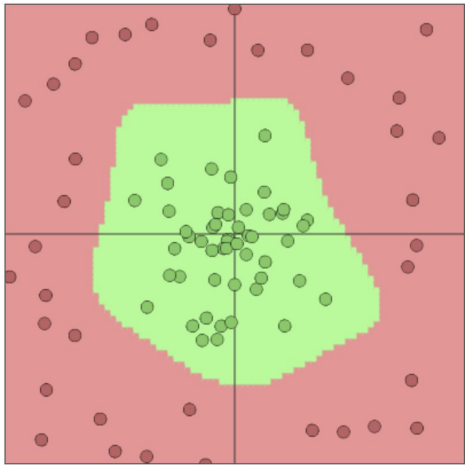
Why in-browser ML

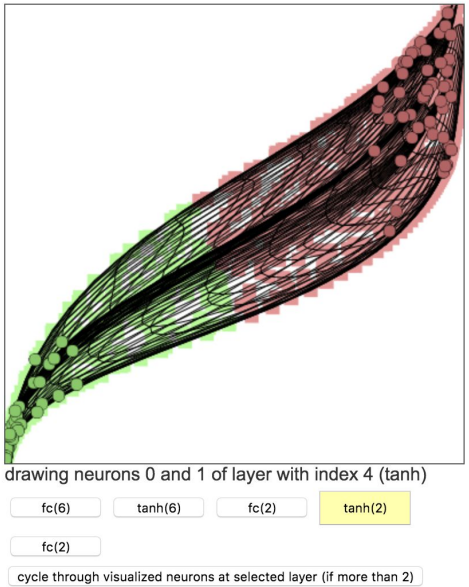
- **No drivers / no installs**
- **Data stays on the client**
- **Interactive**
- **Sensors**
- **Run on desktop, or mobile**
- **GPU acceleration via WebGL**



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

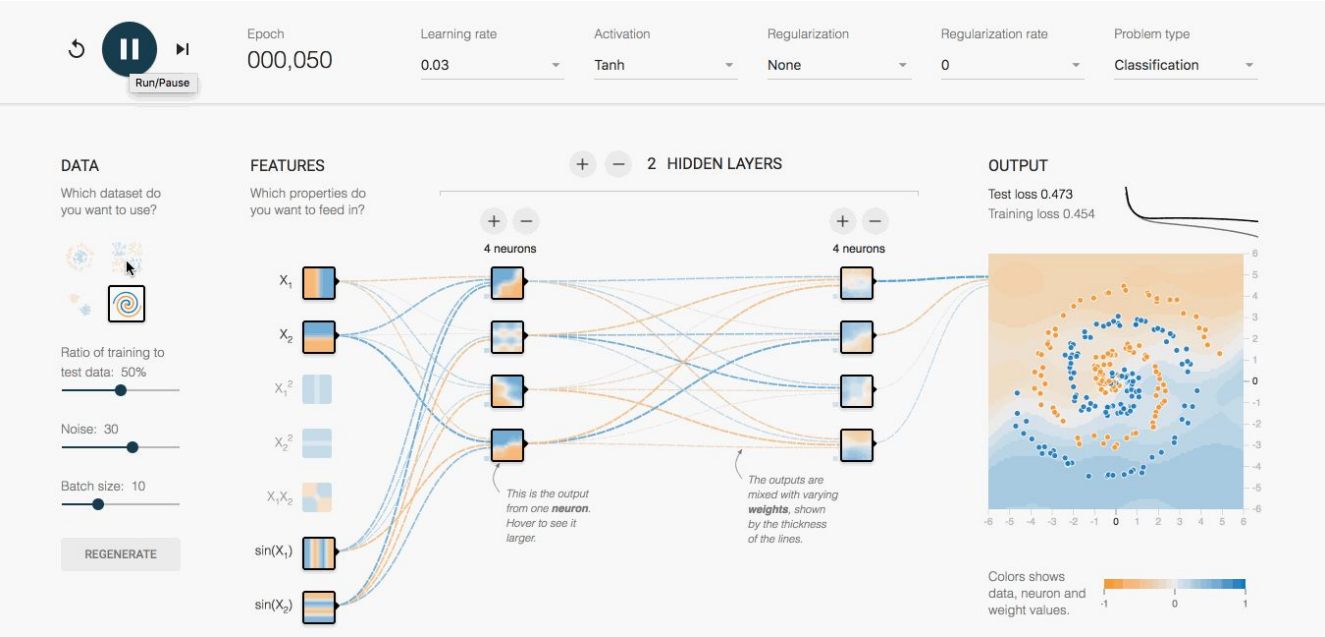
Early days: ConvNetJS





ConvNetJS, Demo

Early days: TensorFlow Playground

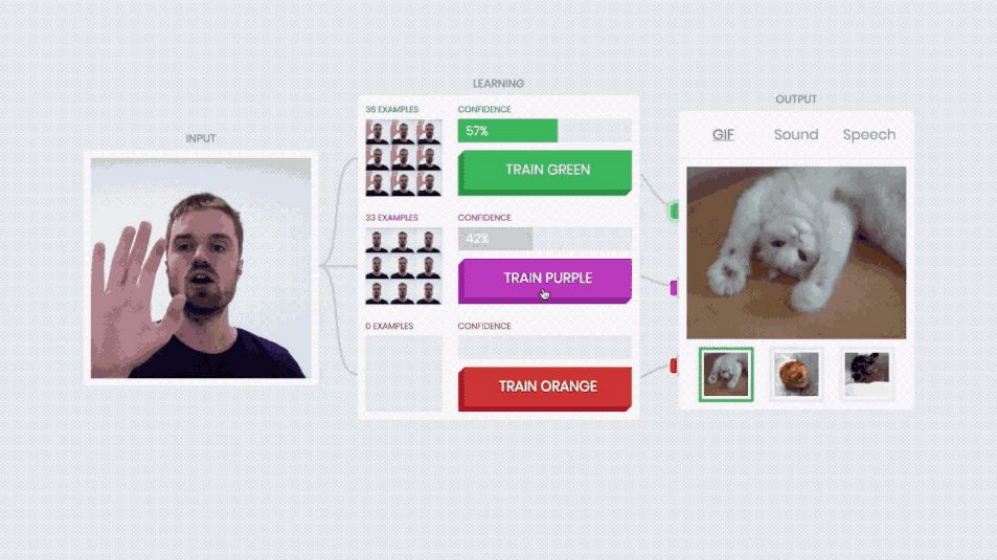


Not powered by tf.js (instead, about 400 lines of JavaScript).

TensorFlow Playground

Early days: 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).

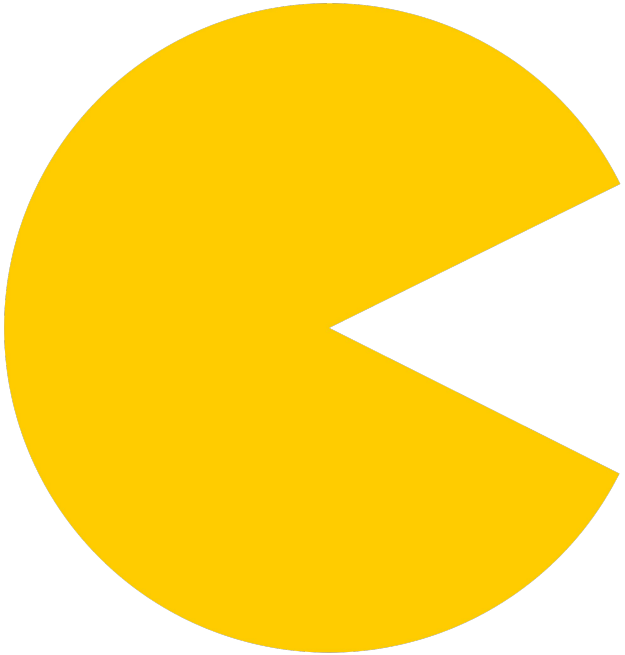
Teachable Machine

PacMan

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

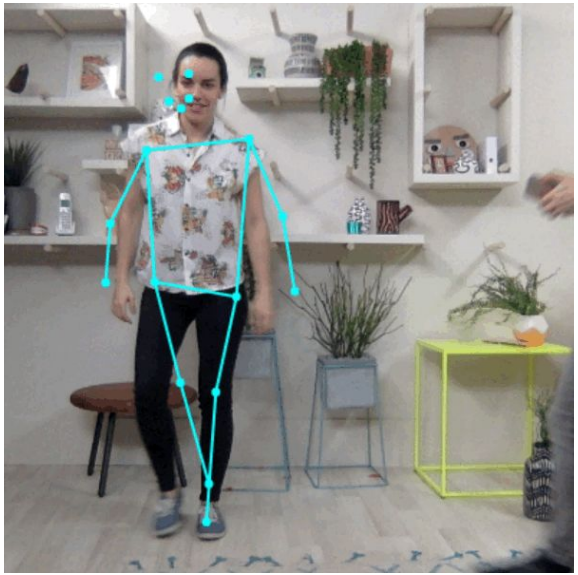
1. **The right MobileNet layer to use was determined experimentally, it's not always the last convolutional one.**

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



PacMan

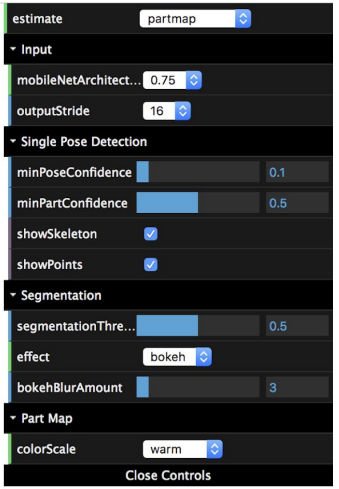
Real-time Human Pose Estimation in the Browser



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

PoseNet, Demo

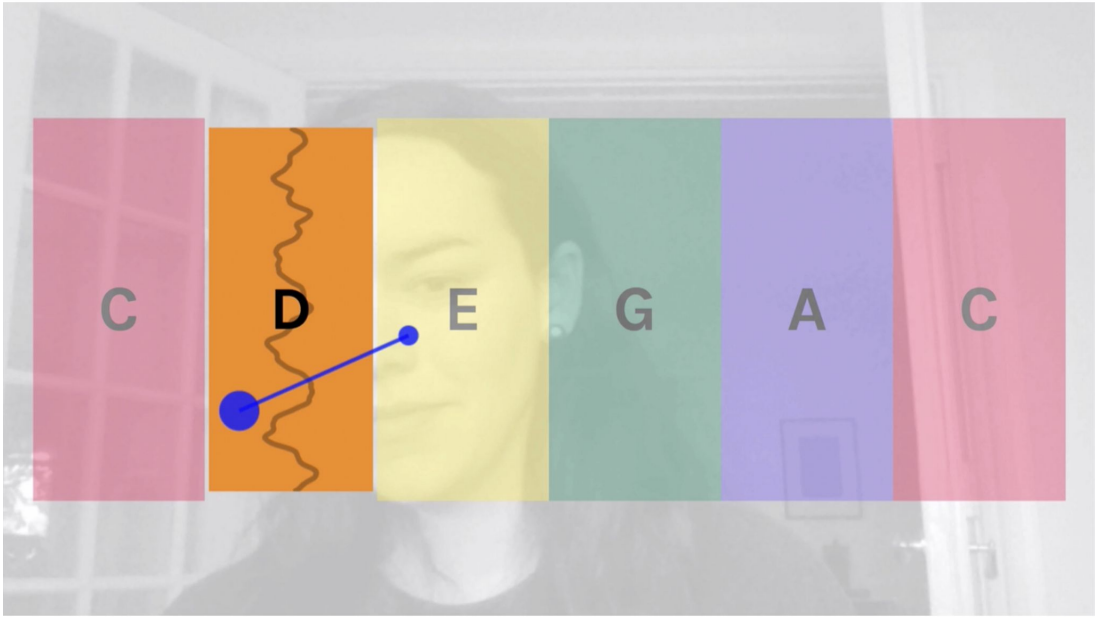
Real-time Image segmentation



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

Article / demo

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

How to debug / chrome dev tools

MobileNets

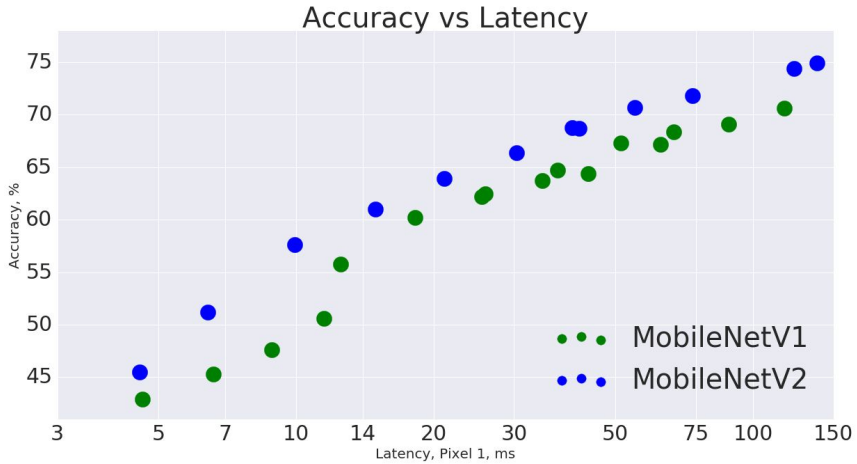
- **Ballpark**, how long does it take to classify an image with high accuracy on a modern phone? (In milliseconds).

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

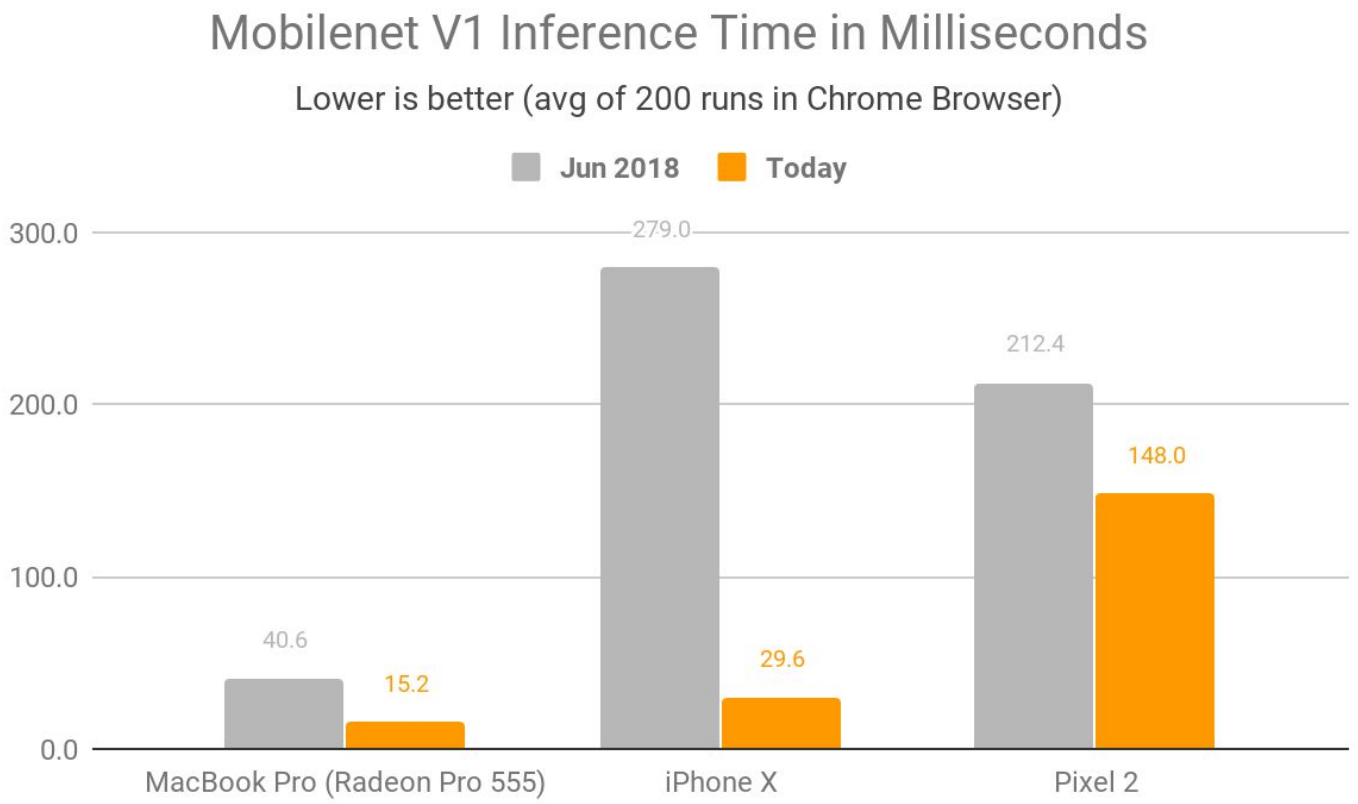
MobileNets



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



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

MobileNets

- **Ballpark**, how long does it take to classify an image with high accuracy on a modern phone? (In milliseconds).

- How about on a **microcontroller**? (Think a $10 Arduino, or something that someday could be embedded in a door and always on).

Pretrained models

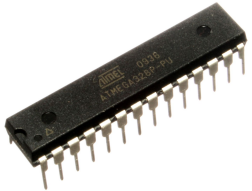
Imagenet Checkpoints

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

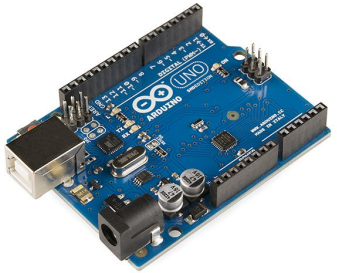
The Future of ML is Tiny

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

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



ATmega328

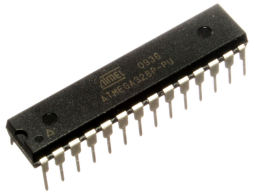


Commonly found on an Arduino. Uno

Why the Future of Machine Learning is Tiny

CNNs run surprisingly well on tiny devices

- MobileNetV2 takes about **22M** Multiply-Adds to classify an image in its smallest configuration.
- This microcontroller can achieve **~1 MIPS / MHz**, and can clock at ~20 MHz.



MobileNets, MobileNetV2

Toxicity detector

Curious what you think of this

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

Demo

- 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

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

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

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

- Idea: 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

- 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

- 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

Papers

- VQA: Visual Question Answering
- Communication-Efficient Learning of Deep Networks from Decentralized Data

Notable blog posts

- A Recipe for Training Neural Networks
- Why the Future of Machine Learning is Tiny