



AI for JavaScript Developers with tensorflow.js



@chrisachard

chrisachard.com

<https://github.com/chrisachard/tfjs-demo>

Overview

- Waaaay too much info for one session
- General idea of deep learning
- Tensorflow.js apis and capabilities

Keras



TensorFlow



PyTorch



<https://www.quora.com/What-are-the-major-differences-between-TensorFlow-Keras-and-PyTorch>



Browser

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

Node.js

```
import * as tf from '@tensorflow/tfjs';
```

Why use TensorFlow.js?

- AI in JS
 - Works in the browser, server, mobile, IoT...
 - Privacy, inference speed (no server roundtrip)
 - Can be as fast (or faster!) as the python counterparts
-

Why not use TensorFlow.js?

- Can be slower
- Extensive supporting libraries already in Python
- Majority of AI work is still done in Python (tutorials, papers, existing algorithms)



LipSync by YouTube

See how well you synchronize to the lyrics of the popular hit "Dance Monkey." This in-browser experience uses the Facemesh model for estimating key points around the lips to score lip-syncing accuracy.

[Explore demo ↗](#)

[View code](#) 

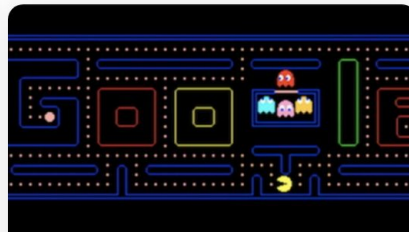


Emoji Scavenger Hunt

Use your phone's camera to identify emojis in the real world. Can you find all the emojis before time expires?

[Explore demo ↗](#)

[View code](#) 



Webcam Controller

Play Pac-Man using images trained in your browser.

[Explore demo ↗](#)

[View code](#) 



Image classification

Classify images with labels from the ImageNet database (MobileNet).

View code 



Object detection

Localize and identify multiple objects in a single image (Coco SSD).

View code 



Body segmentation

Segment person(s) and body parts in real-time (BodyPix).

View code 



Pose estimation

Estimate human poses in real-time (PoseNet).



Text toxicity detection

Score the perceived impact a comment may have on a conversation, from "Very toxic" to "Very healthy" (Toxicity).



Universal sentence encoder

Encode text into embeddings for NLP tasks such as sentiment classification and textual similarity (Universal Sentence Encoder).

<https://www.tensorflow.org/js/models>

Big Question

When and how can we use AI?

Goal

Work backwards to understand what TensorFlow.js is and what it can do

Explore the API with code and examples

Early artificial intelligence stirs excitement.



1950's

1960's

1970's

Machine learning begins to flourish.



1980's

1990's

2000's

Deep learning breakthroughs drive AI boom.



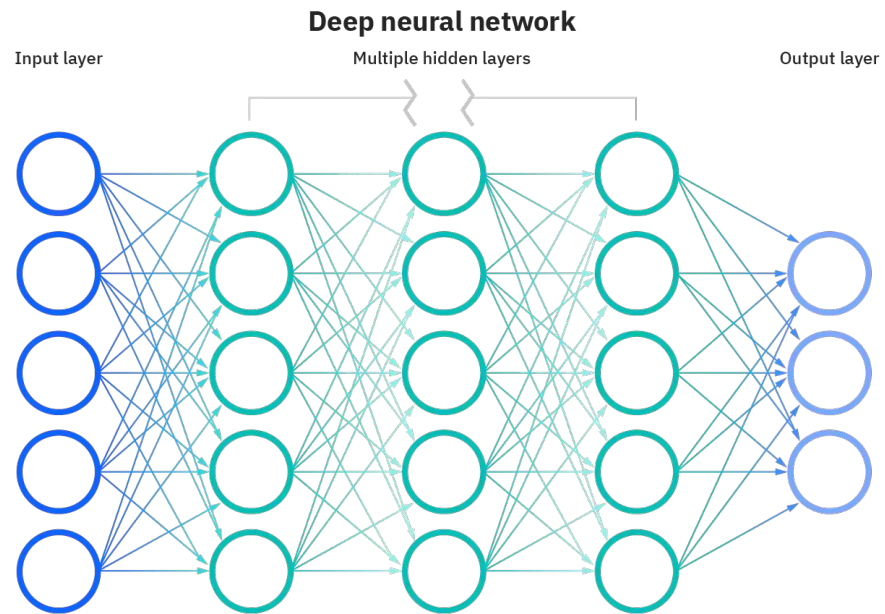
2010's

Deep Learning

Teaching a computer to solve a certain task by giving it examples using a specific structure called a neural network or “deep” neural network

(Can also do machine learning)

What is deep learning?



Conventional

Deep Learning

Conventional

```
if()  
if()  
return  
if()  
.map()  
.filter()  
if()  
return
```

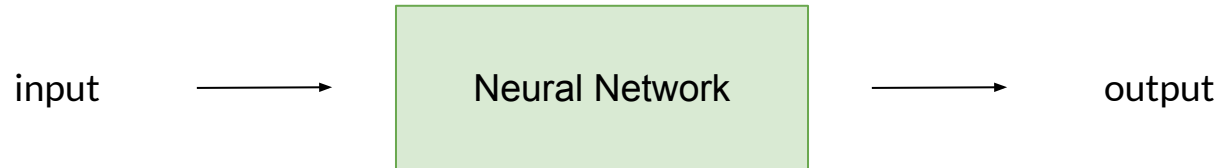
Deep Learning

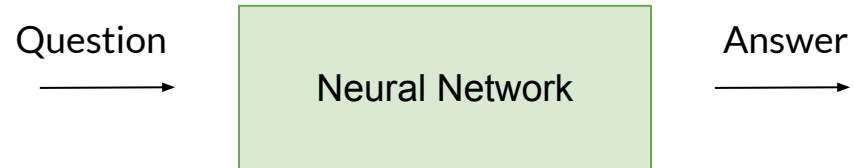
Conventional

```
if()  
if()  
return  
if()  
.map()  
.filter()  
if()  
return
```

Deep Learning

[illegible]





What kind of inputs?

Conventional

```
if()  
if()  
return  
if()  
.map()  
.filter()  
if()  
return
```

Deep Learning

Conventional

Deep Learning

Strings:

“My name is Chris”

Numbers:

0.5, 1, 3.1415

Data Structures:

```
{  
  name: “Chris”,  
  job: “developer”  
}
```

... anything!

Conventional

Strings:

“My name is Chris”

Numbers:

0.5, 1, 3.1415

Data Structures:

```
{
  name: "Chris",
  job: "developer"
}
```

... anything!

Deep Learning

[illegible]

Conventional

Strings:
"My name is Chris"

Numbers:
0.5, 1, 3.1415

Data Structures:
{
 name: "Chris",
 job: "developer"
}

... anything!

Deep Learning

Numbers:
0.5, 1, 3.1415

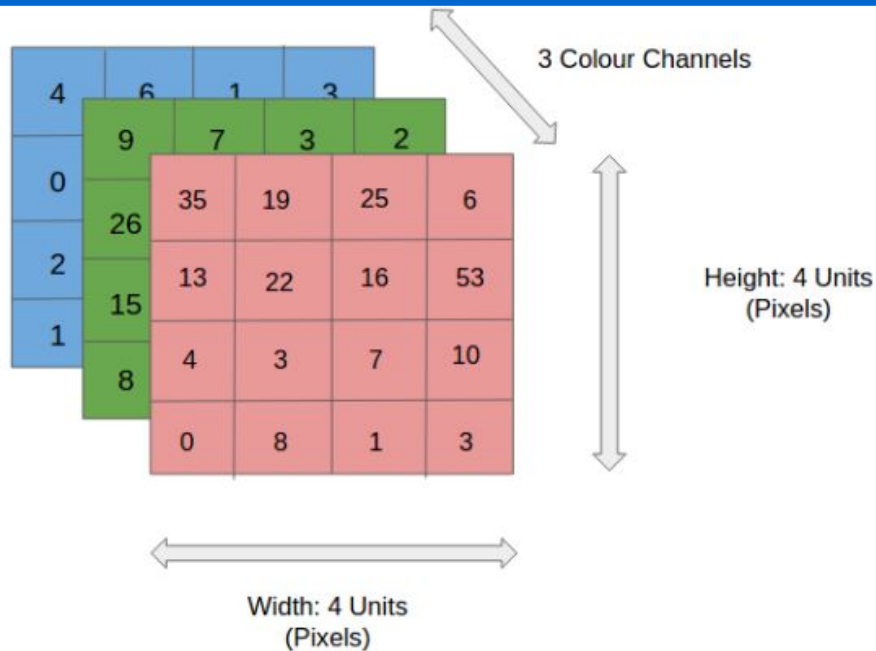
What type of data can we
represent as numbers?

Tabular Data

| Rank ↕ | Country or dependent territory ↕ | Population ↕ | % of world ↕ | Date ↕ |
|--------|-----------------------------------------------------------------------------------------------------------------|---------------|--------------|-------------|
| 1 |  China ^{†[b]} | 1,411,778,724 | 17.9% | 1 Nov 2020 |
| 2 |  India ^{†[c]} | 1,378,250,968 | 17.5% | 16 Jun 2021 |
| 3 |  United States ^{†[d]} | 331,853,982 | 4.21% | 16 Jun 2021 |
| 4 |  Indonesia [†] | 271,350,000 | 3.45% | 31 Dec 2020 |
| 5 |  Pakistan ^{†[e]} | 225,200,000 | 2.86% | 1 Jul 2021 |
| 6 |  Brazil [†] | 213,278,433 | 2.71% | 16 Jun 2021 |
| 7 |  Nigeria [†] | 211,401,000 | 2.68% | 1 Jul 2021 |
| 8 |  Bangladesh [†] | 170,847,360 | 2.17% | 16 Jun 2021 |
| 9 |  Russia ^{†[f]} | 146,171,015 | 1.86% | 1 Jan 2021 |
| 10 |  Mexico [†] | 126,014,024 | 1.60% | 2 Mar 2020 |

https://en.wikipedia.org/wiki/List_of_countries_and_dependencies_by_population

Images



Text

The quick brown fox
jumps over the lazy dog

ASCII or Unicode

84 104 101 32 113 117 105 99 107 32 98 114
111 119 110 32 102 111 120 32 106 117 109
112 115 32 111 118 101 114 32 116 104 101
32 108 97 122 121 32 100 111 103

Word Lookup table

the: 1

quick: 2

brown: 3

fox: 4

jumps: 5

over: 6

lazy: 7

dog: 8

1 2 3 4 5 6 1 7 8

Word Embeddings

the: 0.1, 0.3, 0.4

quick: 1.2, 0.9, 0.1

brown: 0.2, 0.1, 1.1

fox: 0.9, 0.4, 1.4

jumps: 0.4, 0.3, 0.3

over: 0.2, 0.1, 1.2

lazy: 1.1, 0.9, 0.1

dog: 0.9, 2.1, 0.2

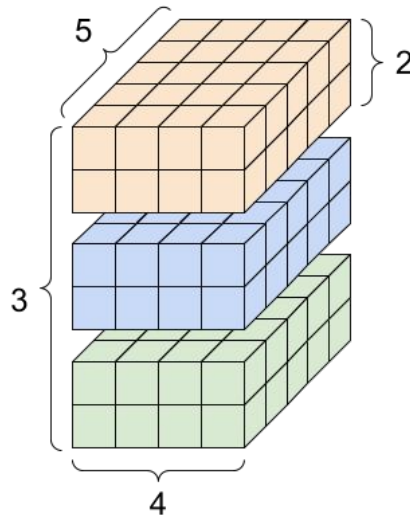
0.1, 0.3, 0.4, 1.2, 0.9, 0.1, 0.2, 0.1, 1.1, 0.9,
0.4, 1.4, 0.4, 0.3, 0.3, 0.2, 0.1, 1.2, 0.1, 0.3,
0.4, 1.1, 0.9, 0.1, 0.9, 2.1, 0.2

Any data can be
represented as numbers

How numbers are handled in TensorFlow.js

Tensors

Multidimensional
Array of Numbers

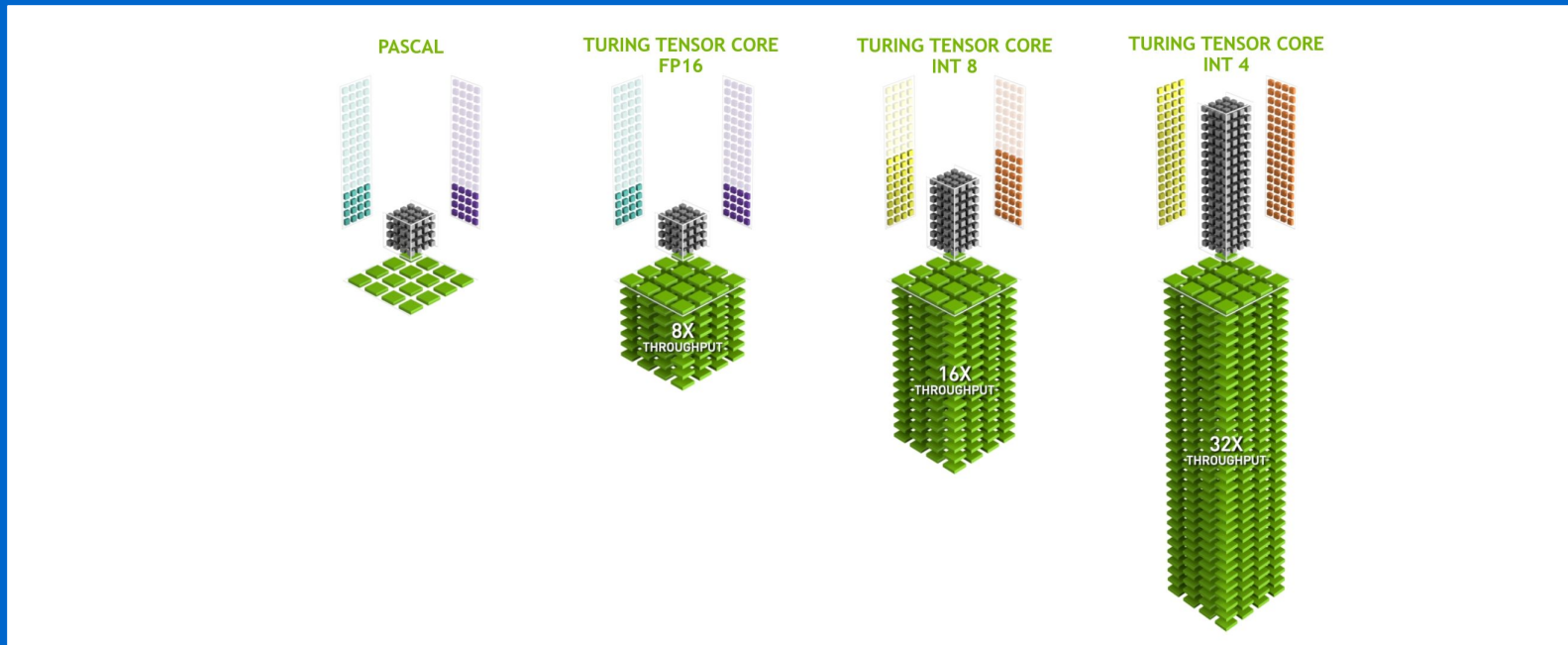


GPUs



<https://www.nvidia.com/en-gb/graphics-cards/>

GPU Tensor Cores



<https://developer.nvidia.com/tensor-cores>

So if we turn data like this:

Artificial intelligence (AI) is intelligence demonstrated by machines, unlike the natural intelligence displayed by humans and animals, which involves consciousness and emotionality. The distinction between the former and the latter categories is often revealed by the acronym chosen. 'Strong' AI is usually labelled as artificial general intelligence (AGI) while attempts to emulate 'natural' intelligence have been called artificial biological intelligence (ABI). Leading AI textbooks define the field as the study of "intelligent agents": any device that perceives its environment and takes actions that maximize its chance of achieving its goals.[3] Colloquially, the term "artificial intelligence" is often used to describe machines that mimic "cognitive" functions that humans associate with the human mind, such as "learning" and "problem solving".[4]

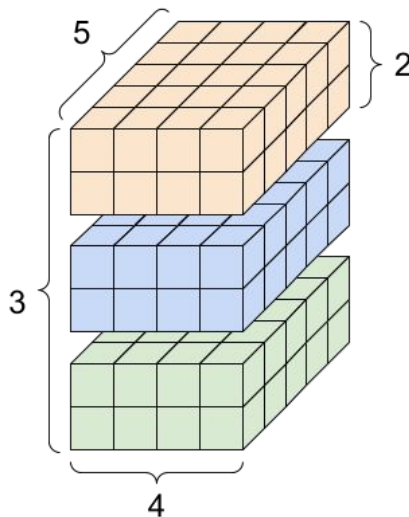
As machines become increasingly capable, tasks considered to require "intelligence" are often removed from the definition of AI, a phenomenon known as the AI effect.[5] A quip in Tesler's Theorem says "AI is whatever hasn't been done yet." [6] For instance, optical character recognition is frequently excluded from things considered to be AI,[7] having become a routine technology.[8] Modern machine capabilities generally classified as AI include successfully understanding human speech,[9] competing at the highest level in strategic game systems (such as chess and Go),[10] and also imperfect-information games like poker,[11] self-driving cars, intelligent routing in content delivery networks, and military simulations.[12]

Artificial intelligence was founded as an academic discipline in 1955, and in the years since has experienced several waves of optimism,[13][14] followed by disappointment and the loss of funding (known as an "AI winter"),[15][16] followed by new approaches, success and renewed funding.[14][17] After AlphaGo defeated a professional Go player in 2015, artificial intelligence once again attracted widespread global attention.[18] For most of its history, AI research has been divided into sub-fields that often fail to communicate with each other.[19] These sub-fields are based on technical considerations, such as particular goals (e.g. "robotics" or "machine learning"),[20] the use of particular tools ("logic" or artificial neural networks), or deep philosophical differences.[23][24][25] Sub-fields have also been based on social factors (particular institutions or the work of particular researchers).[19]

The traditional problems (or goals) of AI research include reasoning, knowledge representation, planning, learning, natural language processing, perception and the ability to move and manipulate objects.[20] AGI is among the field's long-term goals.[26] Approaches include statistical methods, computational intelligence, and traditional symbolic AI. Many tools are used in AI, including versions of search and mathematical optimization, artificial neural networks, and methods based on statistics, probability and economics. The AI field draws upon computer science, information engineering, mathematics, psychology, linguistics, philosophy, and many other fields.

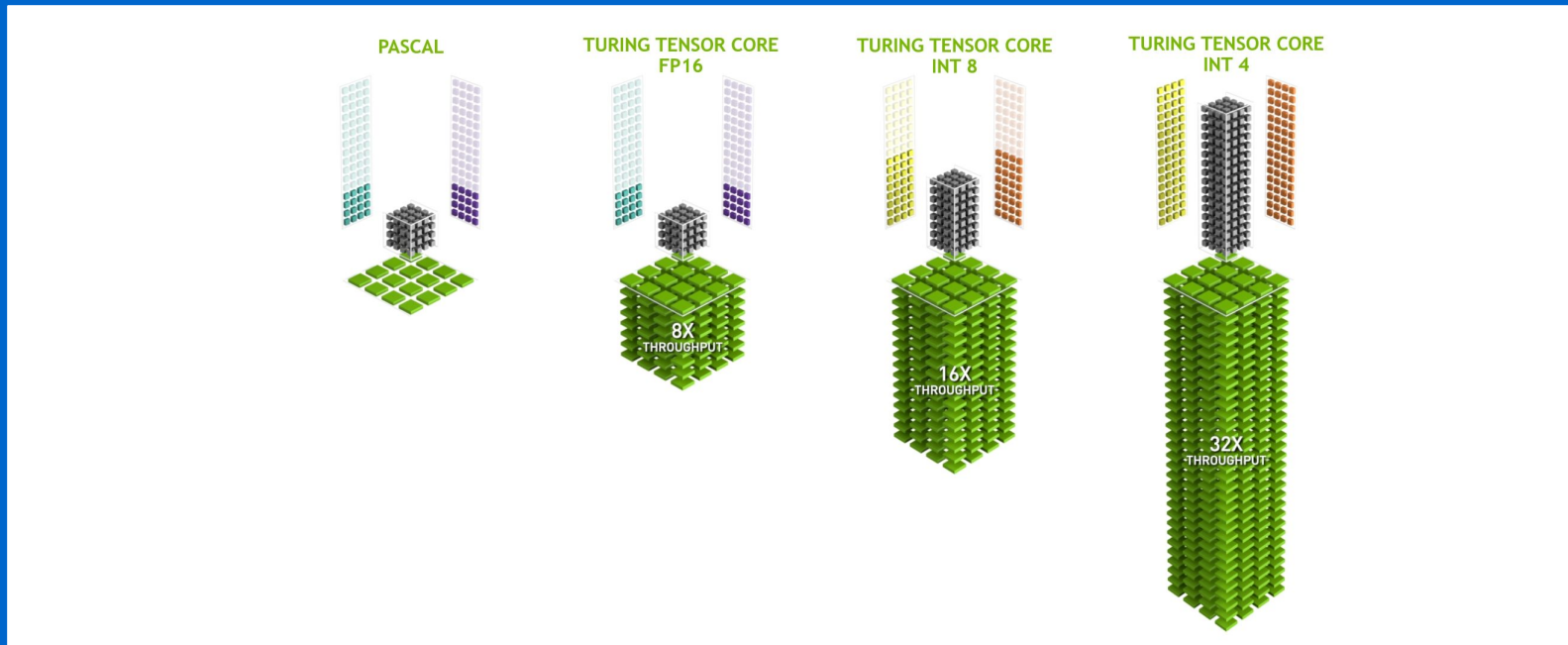
The field was founded on the assumption that human intelligence "can be so precisely described that a machine can be made to simulate it".[27] This raises philosophical arguments about the mind and the ethics of creating artificial beings endowed with human-like intelligence. These issues have been explored by myth, fiction and philosophy since antiquity.[32] Some people also consider AI to be a danger to humanity if it progresses unabated.[33][34] Others believe that AI, unlike previous technological revolutions, will create a risk of mass unemployment.[35]

Into this:



<https://www.tensorflow.org/guide/tensor>

Then we take advantage of this:



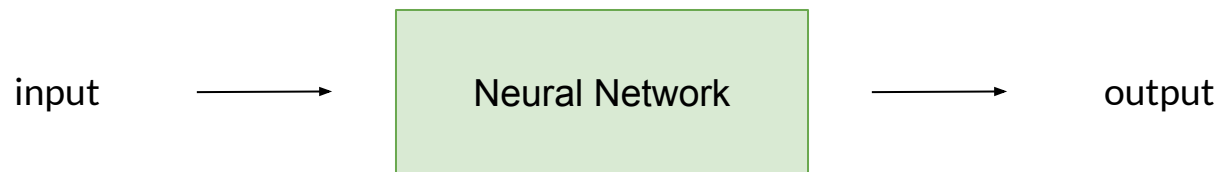
<https://developer.nvidia.com/tensor-cores>

Tensorflow.js handles that for us

```
// Create a rank-2 tensor (matrix) matrix tensor from a multidimensional array.  
const a = tf.tensor([[1, 2], [3, 4]]);  
console.log('shape:', a.shape);  
a.print();
```

https://www.tensorflow.org/js/guide/tensors_operations

Numbers as inputs and outputs



How does the neural network
know what to do?

Conventional

```
if()  
if()  
return  
if()  
.map()  
.filter()  
if()  
return
```

Deep Learning

[illegible]

Deep Learning

input1 * w1 + b1
input2 * w2 + b2
input3 * w3 + b3
input4 * w4 + b4

...

activation (if)

...

...

...

...

...

...

...

...

input1 * w5 + b5
input2 * w6 + b6
input3 * w7 + b7
input4 * w8 + b8

...

activation (if)

...

...

...

...

...

...

...

...

input1 * w9 + b9
input2 * w10 + b10
input3 * w11 + b11
input4 * w12 + b12

...

activation (if)

...

...

...

...

...

...

...

...

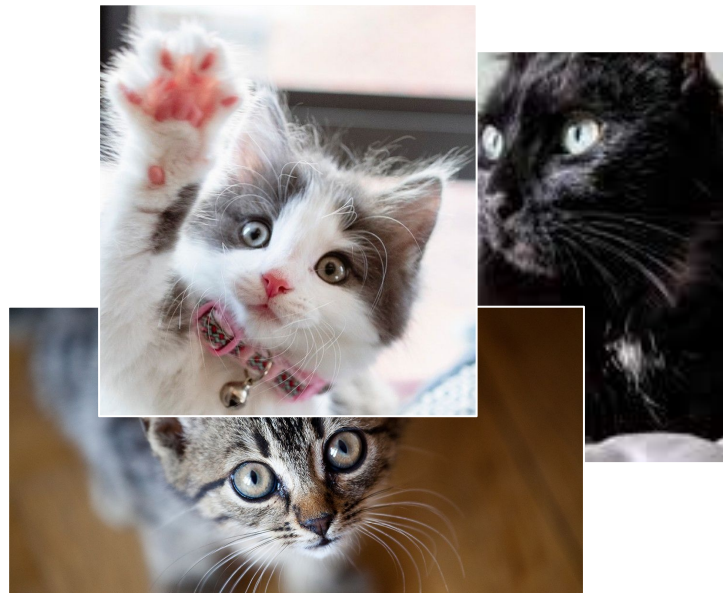
.....

Training

Learn by example



Training data



input

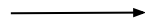


Neural Network



first guess

input



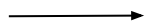
Neural Network



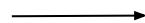
first guess

UPDATE

input



Neural Network



first guess

UPDATE

input

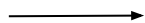


Neural Network

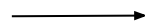


next guess

input



Neural Network



first guess

UPDATE

input



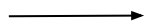
Neural Network



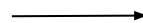
next guess

UPDATE

input



Neural Network



first guess

UPDATE

input



Neural Network



next guess

UPDATE

input



Neural Network

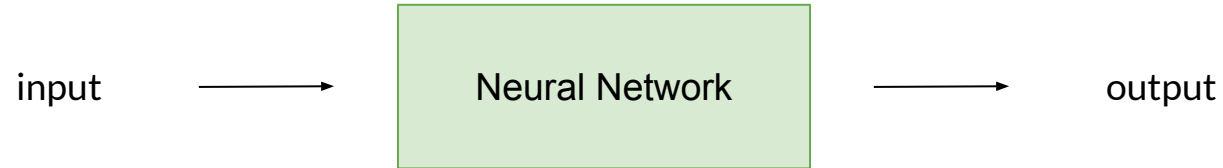


next guess

Update Cycle

1. Run training data through the network
2. Compare nn output to training values
3. Calculate “loss”
4. Tell the nn to update (SGD)
5. Repeat until loss is low enough

Function



Model



Image classification

Classify images with labels from the ImageNet database (MobileNet).

View code 



Object detection

Localize and identify multiple objects in a single image (Coco SSD).

View code 



Body segmentation

Segment person(s) and body parts in real-time (BodyPix).

View code 



Pose estimation

Estimate human poses in real-time (PoseNet).



Text toxicity detection

Score the perceived impact a comment may have on a conversation, from "Very toxic" to "Very healthy" (Toxicity).



Universal sentence encoder

Encode text into embeddings for NLP tasks such as sentiment classification and textual similarity (Universal Sentence Encoder).

<https://www.tensorflow.org/js/models>

Model demo

<https://github.com/chrisachard/tfjs-demo>

What if you can't find a
pretrained model?

Just want cat or dog

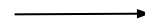


Finetune / transfer learning

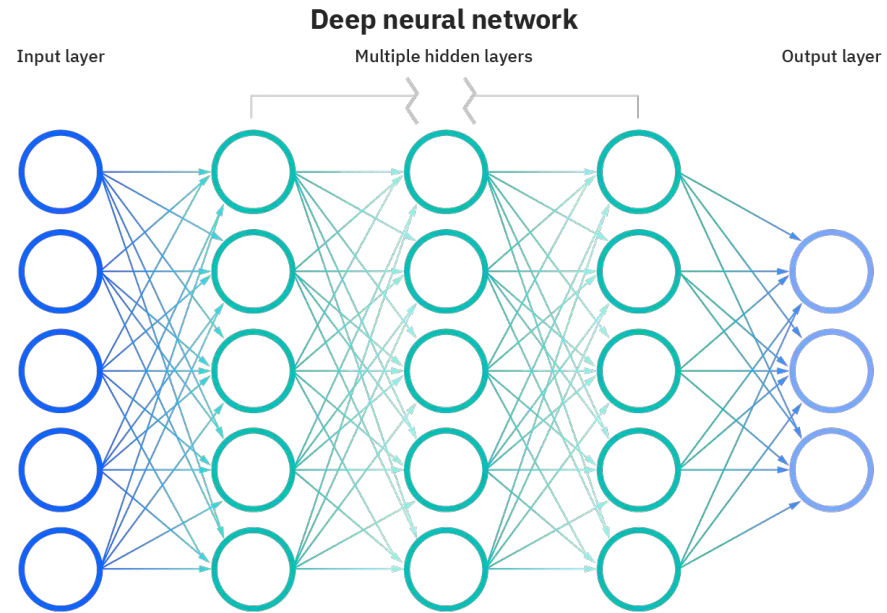
input



Neural Network



output



```
const activation = net.infer(imgEl, embedding=false); // 1x1000  
const activation = net.infer(imgEl, embedding=true); // 1x1024  
const activation = net.infer(imgEl, 'conv_preds'); // 1x1024
```

Embeddings

the: 0.1, 0.3, 0.4

quick: 1.2, 0.9, 0.1

brown: 0.2, 0.1, 1.1

fox: 0.9, 0.4, 1.4

jumps: 0.4, 0.3, 0.3

over: 0.2, 0.1, 1.2

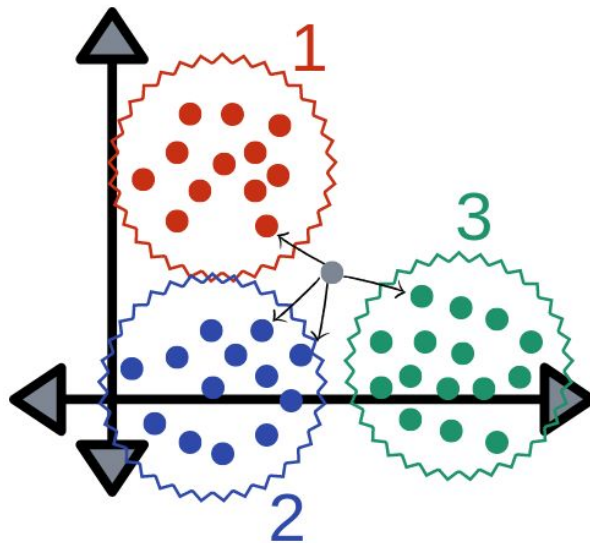
lazy: 1.1, 0.9, 0.1

dog: 0.9, 2.1, 0.2

0.1, 0.3, 0.4, 1.2, 0.9, 0.1, 0.2, 0.1, 1.1, 0.9,
0.4, 1.4, 0.4, 0.3, 0.3, 0.2, 0.1, 1.2, 0.1, 0.3,
0.4, 1.1, 0.9, 0.1, 0.9, 2.1, 0.2

0.5242,0.4461,0.1817,0.1304,0.8073,0.8974,0.1202,0.009,0.7317,0.3325,0.1818,0.5831,0.5179,0.0827,0.5567,0.7725,0.006,0.7318,0.7306,0.303,0.1504,0.2043,0.1813,0.9047,0.518,0.7597,0.2904,0.0814,0.3253,0.0614,0.2592,0.1714,0.5539,0.2187,0.2944,0.4321,0.984,0.0607,0.6009,0.0868,0.9982,0.3101,0.2956,0.9457,0.7098,0.6005,0.8776,0.9629,0.7912,0.2803,0.004,0.049,0.5366,0.222,0.5365,0.515,0.7925,0.7512,0.8123,0.3454,0.2955,0.2891,0.5253,0.0965,0.2076,0.1861,0.3465,0.3174,0.2632,0.936,0.9174,0.1646,0.2415,0.9373,0.5873,0.3228,0.6988,0.8271,0.4713,0.831,0.7772,0.9996,0.6928,0.35,1.0,1.368,0.5337,0.8176,0.9159,0.3958,0.0793,0.5087,0.9047,0.5065,0.2389,0.8918,0.0594,0.724,0.5033,0.2021,0.3615,0.8783,0.5795,0.6566,0.2295,0.613,0.4762,0.9096,0.872,0.1397,0.8864,0.1594,0.1062,0.7781,0.0614,0.3452,0.2413,0.0513,0.3179,0.8473,0.5869,0.2397,0.5455,0.6366,0.2269,0.0372,0.5976,0.8362,0.3707,0.203,0.6425,0.4955,0.7133,0.544,0.5095,0.9857,0.4497,0.8266,0.1112,0.4102,0.3713,0.7033,0.5795,0.4084,0.2098,0.4597,0.2921,0.4915,0.8997,0.6514,0.3653,0.4593,0.9562,0.7118,0.6548,0.9393,0.3509,0.9367,0.0818,0.5975,0.317,0.8093,0.5739,0.6361,0.425,0.7528,0.4812,0.48,0.47,0.0609,0.1684,0.0573,0.657,0.8689,0.0303,0.3746,0.1861,0.5946,0.2355,0.8827,0.9118,0.0872,0.0178,0.7941,0.3518,0.942,0.632,0.2585,0.8876,0.3277,0.4141,0.5018,0.7151,0.1447,0.5231,0.5059,0.808,0.1344,0.0685,0.442,0.9066,0.4893,0.268,0.6492,0.5797,0.5519,0.5602,0.571,0.8795,0.4011,0.5966,0.2432,0.7739,0.3253,0.9454,0.4476,0.9841,0.5957,0.4239,0.8138,0.116,0.3869,0.7463,0.2471,0.4084,0.3897,0.7843,0.1107,0.8328,0.0095,0.4837,0.2196,0.3808,0.0471,0.7902,0.9545,0.5542,0.4743,0.8329,0.3118,0.8771,0.1527,0.2652,0.0339,0.7535,0.9903,0.5545,0.4977,0.9152,0.6517,0.9984,0.2141,0.9467,0.3868,0.6666,0.2617,0.7793,0.233,0.0859,0.8933,0.18,0.831,0.1605,0.9867,0.3076,0.7197,0.905,0.2947,0.6607,0.2236,0.1655,0.2437,0.485,0.4271,0.8163,0.3872,0.9182,0.9864,0.4608,0.1789,0.239,0.5931,0.0716,0.716,0.4934,0.4961,0.862,0.1839,0.1898,0.5863,0.4034,0.1506,0.5161,0.6818,0.9903,0.1705,0.5091,0.6959,0.9049,0.2001,0.9479,0.4237,0.1544,0.3024,0.9392,0.1944,0.0367,0.0093,0.2892,0.1098,0.3482,0.7537,0.3178,0.7401,0.0129,0.674,0.025,0.3216,0.2894,0.3335,0.7886,0.4539,0.1647,0.7501,0.7568,0.4831,0.649,0.2707,0.9931,0.3981,0.3286,0.6862,0.7469,0.5069,0.4281,0.9486,0.0139,0.9222,0.5303,0.382,0.7777,0.96,0.3994,0.5014,0.3313,0.1737,0.3094,0.2928,0.7804,0.232,0.466,0.0838,0.8606,0.95,0.6478,0.9787,0.73,0.4521,0.4996,0.599,0.5906,0.3735,0.4865,0.9532,0.1854,0.804,0.035,0.1312,0.2978,0.3543,0.9008,0.4797,0.1253,0.821,0.029,0.0558,0.709,0.3614,0.3774,0.3685,0.8508,0.5211,0.0487,0.966,0.4926,0.54,0.1254,0.8275,0.2483,0.1231,0.8226,0.1774,0.0164,0.659,0.8834,0.3254,0.1735,0.5906,0.8165,0.2979,0.9925,0.3096,0.709,0.2457,0.638,0.6644,0.2757,0.8488,0.1583,0.5993,0.907,0.4,0.9139,0.0175,0.1821,0.823,0.8906,0.7415,0.2099,0.0733,0.8714,0.2419,0.1908,0.7517,0.588,0.9941,0.8181,0.7892,0.76,0.6702,0.5665,0.3725,0.5754,0.7125,0.9118,0.364,0.3795,0.2,0.8726,0.4068,0.5327,0.3243,0.7733,0.4413,0.8614,0.2185,0.6995,0.7421,0.3301,0.1346,0.3499,0.617,0.1564,0.1607,0.9293,0.1363,0.017,0.9875,0.2806,0.4118,0.922,0.0437,0.3859,0.1934,0.6441,0.8812,0.3922,0.0655,0.3618,0.6019,0.0873,0.574,0.6694,0.8662,0.8826,0.1784,0.901,0.5218,0.2439,0.7187,0.7598,0.5106,0.0674,0.1798,0.2852,0.198,0.8387,0.8089,0.0574,0.3764,0.5926,0.0212,0.0532,0.5576,0.3768,0.4361,0.9015,0.4988,0.6284,0.8411,0.9519,0.0951,0.5345,0.8871,0.4324,0.8699,0.988,0.4255,0.1602,0.0679,0.4148,0.996,0.8668,0.4389,0.7054,0.1838,0.1899,0.0868,0.793,0.662,0.2818,0.677,0.4179,0.3602,0.7621,0.7729,0.5243,0.0896,0.8967,0.074,0.3582,0.0473,0.4413,0.7451,0.0128,0.3339,0.3004,0.3552,0.008,0.315,0.558,0.7988,0.2268,0.1625,0.9034,0.0398,0.1529,0.3236,0.5123,0.0134,0.6257,0.7139,0.1514,0.3641,0.2485,0.7836,0.2305,0.5434,0.1049,0.7934,0.8972,0.2895,0.064,0.29,0.5523,0.6963,0.7259,0.0931,0.5828,0.8206,0.9445,0.7052,0.3374,0.6886,0.3661,0.0287,0.9531,0.1669,0.5364,0.6089,0.1322,0.8844,0.0917,0.1104,0.1904,0.0337,0.7934,0.6178,0.1339,0.269,0.3316,0.4836,0.2366,0.8869,0.6365,0.9455,0.7407,0.9484,0.3356,0

KNN



<https://medium.com/capital-one-tech/k-nearest-neighbors-knn-algorithm-for-machine-learning-e883219c8f26>

Finetune demo

<https://github.com/chrisachard/tfjs-demo>

What if you can't finetune a
model?

Data

| Rank ↕ | Country or dependent territory ↕ | Population ↕ | % of world ↕ | Date ↕ |
|--------|-----------------------------------------------------------------------------------------------------------------|---------------|--------------|-------------|
| 1 |  China ^{†[b]} | 1,411,778,724 | 17.9% | 1 Nov 2020 |
| 2 |  India ^{†[c]} | 1,378,250,968 | 17.5% | 16 Jun 2021 |
| 3 |  United States ^{†[d]} | 331,853,982 | 4.21% | 16 Jun 2021 |
| 4 |  Indonesia [†] | 271,350,000 | 3.45% | 31 Dec 2020 |
| 5 |  Pakistan ^{†[e]} | 225,200,000 | 2.86% | 1 Jul 2021 |
| 6 |  Brazil [†] | 213,278,433 | 2.71% | 16 Jun 2021 |
| 7 |  Nigeria [†] | 211,401,000 | 2.68% | 1 Jul 2021 |
| 8 |  Bangladesh [†] | 170,847,360 | 2.17% | 16 Jun 2021 |
| 9 |  Russia ^{†[f]} | 146,171,015 | 1.86% | 1 Jan 2021 |
| 10 |  Mexico [†] | 126,014,024 | 1.60% | 2 Mar 2020 |

https://en.wikipedia.org/wiki/List_of_countries_and_dependencies_by_population

Option 1 - use Python

```
# bash
```

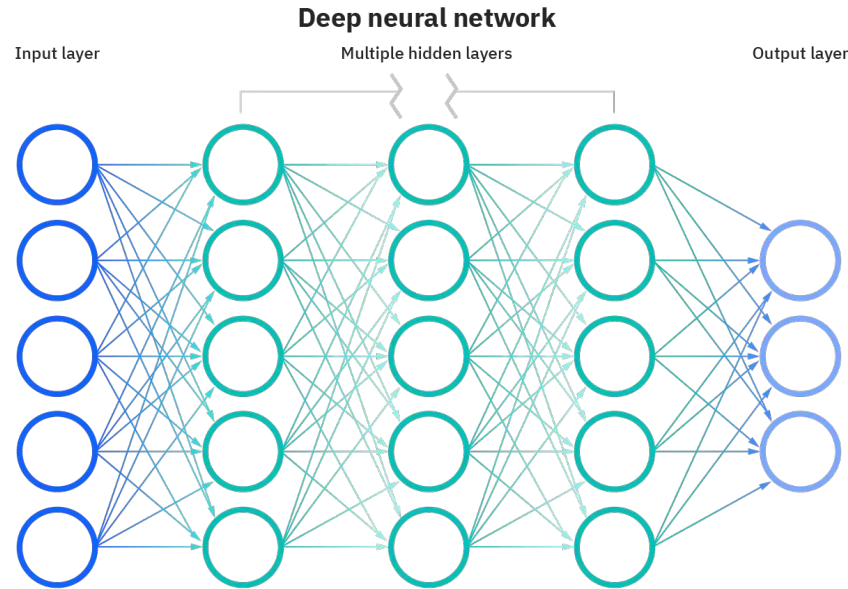
```
tensorflowjs_converter --input_format keras \  
    path/to/my_model.h5 \  
    path/to/tfjs_target_dir
```

```
import * as tf from '@tensorflow/tfjs';
```

```
const model = await tf.loadLayersModel('https://foo.bar/tfjs_artifacts/model.json');
```

https://www.tensorflow.org/js/tutorials/conversion/import_keras

Option 2 - create new network













Network types - fully connected

```
const NUM_OUTPUT_CLASSES = 10;
model.add(tf.layers.dense({
  units: NUM_OUTPUT_CLASSES,
  kernelInitializer: 'varianceScaling',
  activation: 'softmax'
}));
```

<https://codelabs.developers.google.com/codelabs/tfjs-training-classfication/index.html>

Fully connected = “one row at a time”

| Rank ↕ | Country or dependent territory ↕ | Population ↕ | % of world ↕ | Date ↕ |
|--------|-----------------------------------------------------------------------------------------------------------------|---------------|--------------|-------------|
| 1 |  China ^{†[b]} | 1,411,778,724 | 17.9% | 1 Nov 2020 |
| 2 |  India ^{†[c]} | 1,378,250,968 | 17.5% | 16 Jun 2021 |
| 3 |  United States ^{†[d]} | 331,853,982 | 4.21% | 16 Jun 2021 |
| 4 |  Indonesia [†] | 271,350,000 | 3.45% | 31 Dec 2020 |
| 5 |  Pakistan ^{†[e]} | 225,200,000 | 2.86% | 1 Jul 2021 |
| 6 |  Brazil [†] | 213,278,433 | 2.71% | 16 Jun 2021 |
| 7 |  Nigeria [†] | 211,401,000 | 2.68% | 1 Jul 2021 |
| 8 |  Bangladesh [†] | 170,847,360 | 2.17% | 16 Jun 2021 |
| 9 |  Russia ^{†[f]} | 146,171,015 | 1.86% | 1 Jan 2021 |
| 10 |  Mexico [†] | 126,014,024 | 1.60% | 2 Mar 2020 |

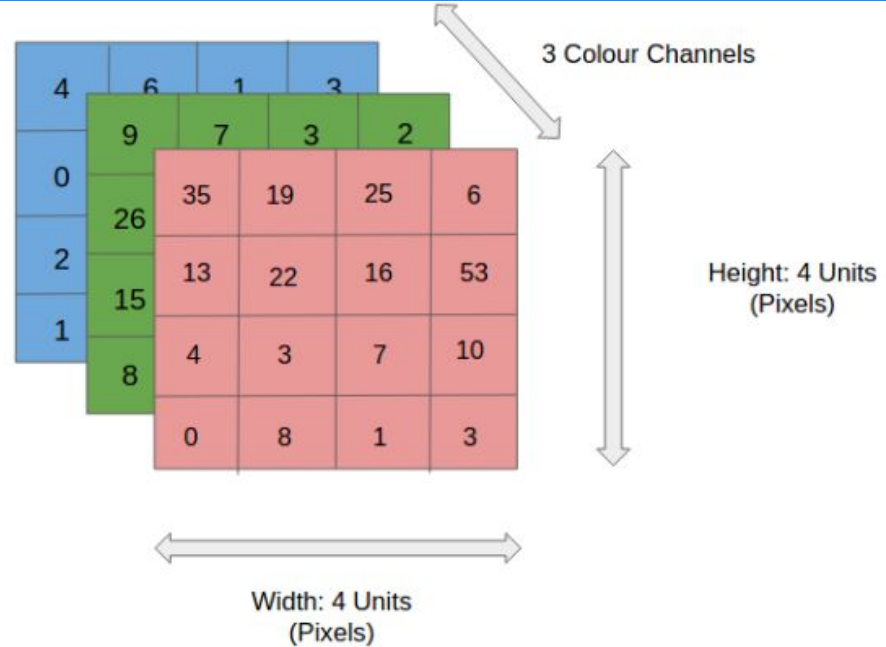
https://en.wikipedia.org/wiki/List_of_countries_and_dependencies_by_population

Network types - CNN

```
model.add(tf.layers.conv2d({  
    inputShape: [IMAGE_WIDTH, IMAGE_HEIGHT, IMAGE_CHANNELS],  
    kernelSize: 5,  
    filters: 8,  
    strides: 1,  
    activation: 'relu',  
    kernelInitializer: 'varianceScaling'  
}));
```

<https://codelabs.developers.google.com/codelabs/tfjs-training-classfication/index.html>

CNN = “sliding window”



Network types - RNN

```
const lstm1 = (data: tf.Tensor2D, c: tf.Tensor2D, h: tf.Tensor2D) =>
  tf.basicLSTMCell(forgetBias, lstmKernel1, lstmBias1, data, c, h);
const lstm2 = (data: tf.Tensor2D, c: tf.Tensor2D, h: tf.Tensor2D) =>
  tf.basicLSTMCell(forgetBias, lstmKernel2, lstmBias2, data, c, h);
const lstm3 = (data: tf.Tensor2D, c: tf.Tensor2D, h: tf.Tensor2D) =>
  tf.basicLSTMCell(forgetBias, lstmKernel3, lstmBias3, data, c, h);
```










https://github.com/magenta/magenta-demos/blob/master/performance_rnn/performance_rnn.ts

RNN = variable length input

The quick brown fox
jumps over the lazy dog

Things to remember

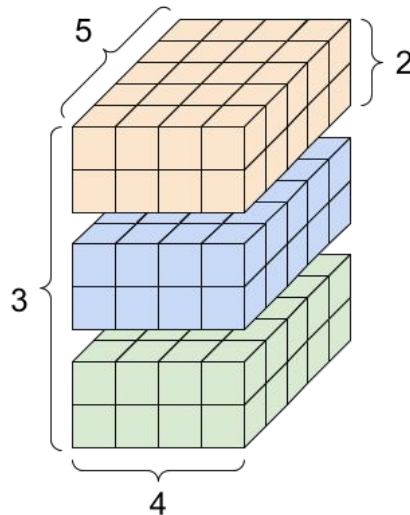
1. Look for existing models

| | | |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  Image classification Classify images with labels from the ImageNet database (MobileNet). View code  |  Object detection Localize and identify multiple objects in a single image (Coco SSD). View code  |  Body segmentation Segment person(s) and body parts in real-time (BodyPix). View code  |
|  Pose estimation Estimate human poses in real-time (PoseNet). |  Text toxicity detection Score the perceived impact a comment may have on a conversation, from "Very toxic" to "Very healthy" (Toxicity). |  Universal sentence encoder Encode text into embeddings for NLP tasks such as sentiment classification and textual similarity (Universal Sentence Encoder). |

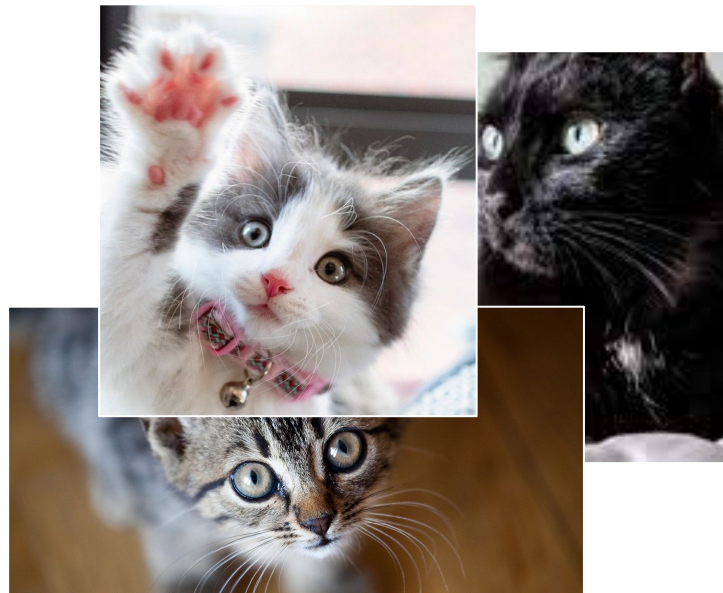
<https://www.tensorflow.org/js/models>

2. Data = tensors

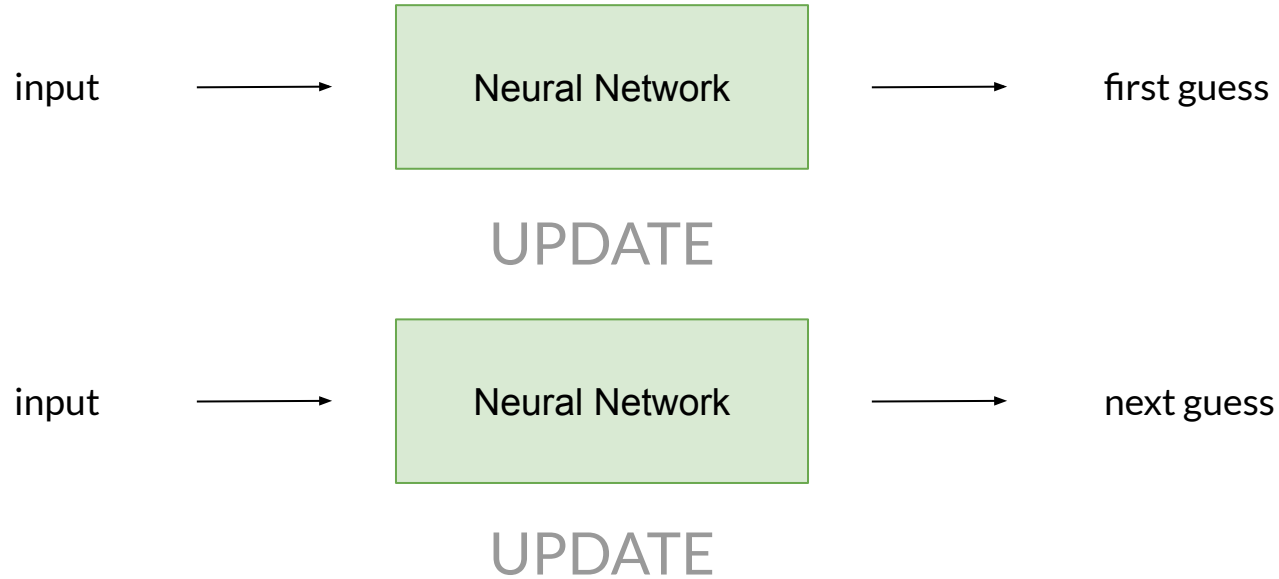
Multidimensional
Array of Numbers



3. Training data




4. Training loop




5. Why use TensorFlow.js?

- AI in JS
 - Works in the browser, server, mobile, IoT...
 - Privacy, inference speed (no server roundtrip)
 - Can be as fast (or faster!) as the python counterparts
-

 TensorFlow

InstallLearn ▾API ▾Resources ▾More ▾

 Search

English ▾

For JavaScript

OverviewTutorialsGuideModelsDemosAPI

Get started

Setup

Upgrade to TensorFlow.js 3.0

Train models

Fit a curve to two-dimensional data

Recognize handwritten digits with CNNs

Predict baseball pitch types in Node.js

Transfer learning

What is transfer learning?



Build an image classifier

Build an audio recognizer

Have a question? Connect with the community at the TensorFlow Forum

Visit Forum

TensorFlow > Learn > For JavaScript > Tutorials

Rate and review  

Get Started

TensorFlow.js is a JavaScript Library for training and deploying machine learning models in the browser and in Node.js.

See the sections below for different ways you can get started.

<https://www.tensorflow.org/js/tutorials>



Questions?



@chrisachard
chrisachard.com
