ML in the Browser

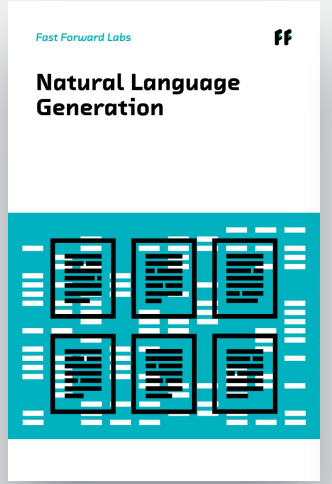
Interactive Experiences with Tensorflow.js

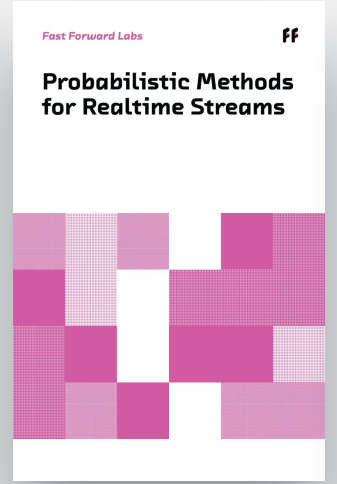
Victor Dibia, PhD
Cloudera **Fast Forward Labs**
@vykthur

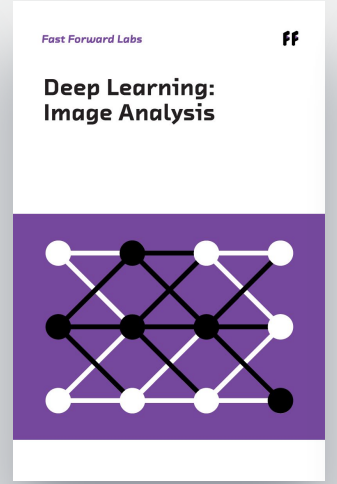
Research Engineer
Cloudera **Fast Forward Labs,** Brooklyn.

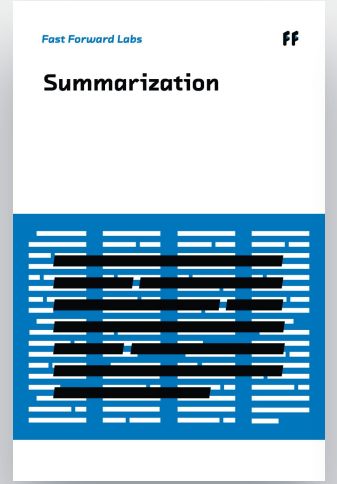
@vykthur | **@victordibia**

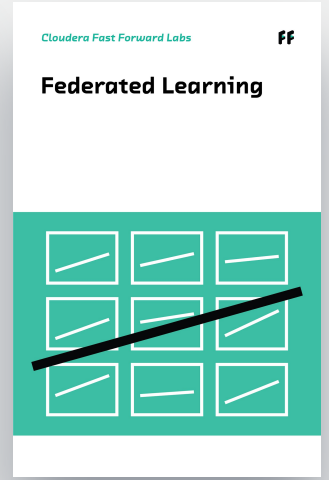
- Background in CS, Human Computer Interaction (HCI), Applied AI.
- Previously Research Scientist at IBM Research New York.
- Community Contributions - GDE for Machine Learning
- Research Engineer (ML) at Cloudera Fast Forward Labs

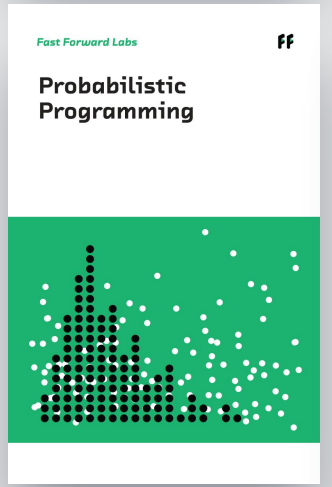

Fast Forward Labs
FF
Natural Language Generation

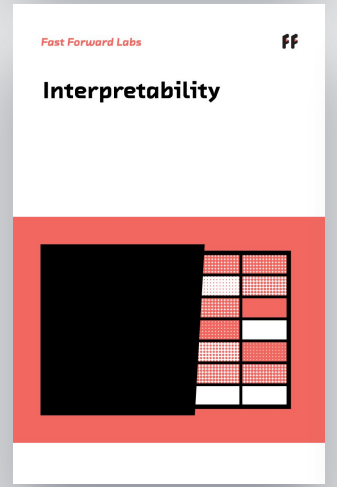

Fast Forward Labs
FF
Probabilistic Methods for Realtime Streams

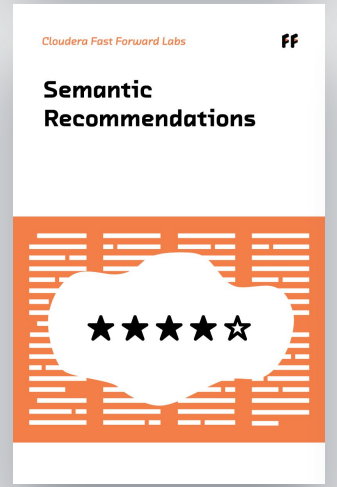

Fast Forward Labs
FF
Deep Learning: Image Analysis

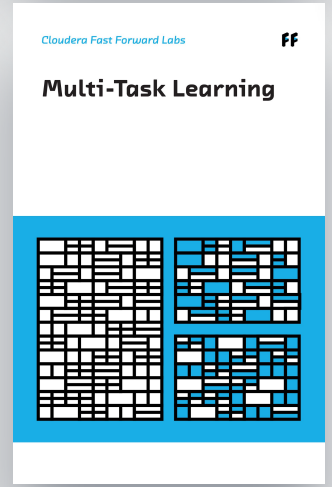

Fast Forward Labs
FF
Summarization


Cloudera Fast Forward Labs
FF
Federated Learning


Fast Forward Labs
FF
Probabilistic Programming


Fast Forward Labs
FF
Interpretability


Cloudera Fast Forward Labs
FF
Semantic Recommendations


Cloudera Fast Forward Labs
FF
Multi-Task Learning

ConvNet Playground | Semantic Search | Model Explorer | FAQ

Image Similarity Search

Welcome!

This demo allows you to perform **semantic image search** using convolutional neural networks [**INCEPTIONV3**]. When you select an image (by clicking it), a neural network *looks* at the content of all images in our dataset and shows you the **top 15** most similar images to the selected image.

Advanced Options

On

Interested in modifying search configurations (try different datasets, models, layers metrics) or a UMAP visualization of the features extracted by each layer? Turn on advanced options.

▲ Search Configuration

Select Dataset

ICONIC200 — 200 images
FASHION200 — 200 images
TINYIMAGENET — 200 images
CIFAR10 — 200 images

ICONIC200 | ? More Info

Select Model

MOBILENET — 4.3M params.
EFFICIENTNE .. — 5.3M params.
DENSENET121 — 8.1M params.
XCEPTION — 22.9M params.
INCEPTIONV3 — 23.9M params.
RESNET50 — 25.6M params.
EFFICIENTNE .. — 30.6M params.
VGG16 — 138.4M params.
VGG19 — 143.7M params.

INCEPTIONV3 — **23.9M Parameters** 313 layers

Select Layer

layer 1 — 864 params
layer 7 — 28.7k params
layer 64 — 725.8k params
layer 185 — 6.5M params
layer 196 — 6.8M params
layer 236 — 9.8M params
layer 294 — 21.4M params
layer 310 — 21.8M params

LAYER 310 — **Type: Concatenate** | MIXED10 21.8M model parameters

Distance Metric

COSINE — distance
EUCLIDEAN — distance
MINKOWSKI — distance

COS .. | EUC .. | MIN ..

COSINE

▼ Visualization of Embeddings (UMAP) for Extracted Features

INCEPTI

▲ **Top 15 results** based on your search configuration [**MODEL:** INCEPTIONV3 | **LAYER:** 310 | **DISTANCE METRIC:** COSINE]

Search result score

90.0%

What is this?

dst: 0.66 | dst: 0.66 | dst: 0.64 | dst: 0.61 | dst: 0.60 | dst: 0.60 | dst: 0.58 | dst: 0.57 | dst: 0.57 | dst: 0.57 | dst: 0.56

dst: 0.56 | dst: 0.56 | dst: 0.56 | dst: 0.54

ConvNet Playground | Semantic Search | Model Explorer | FAQ

Image Similarity Search

? More Info

Welcome!

This demo allows you to perform **semantic image search** using convolutional neural networks [**INCEPTIONV3**]. When you select an image (by clicking it), a neural network looks at the content of all images in our dataset and shows you the **top 15** most similar images to the selected image.

Advanced Options

Off

Interested in modifying search configurations (try different datasets, models, layers and distance metrics) or a UMAP visualization of the features extracted by each layer? Turn on advanced options.

All | By Category

DATASET: [ICONIC200] A dataset of 200 images across 10 categories (20 images per category) crawled from the Flickr API.

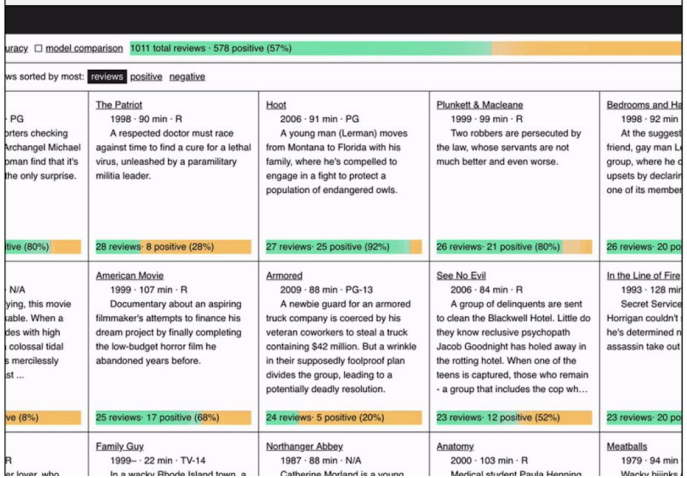
Click an image to search for other similar images.

AI Experiments

A collection of machine learning prototypes, demos, and code by Cloudera Fast Forward Labs.

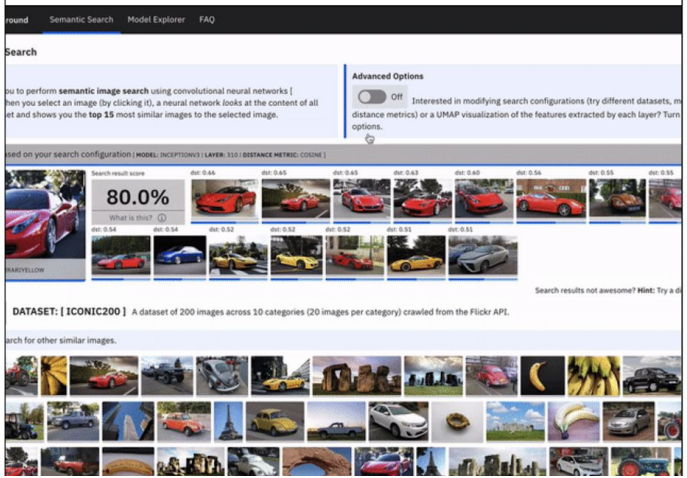
PROTOTYPE

Textflix



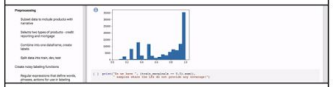
PROTOTYPE

ConvNet Playground



NOTEBOOK

Weak supervision with Snorkel



A notebook showing how to train a complaint classifier with Snorkel. Using data from the Consumer Financial Protection Bureau. github.com/fastforwardlabs/snor...

LIBRARY

Handtrack.js

experiments.fastforwardlabs.com

Agenda

- Why Machine Learning in the Browser?
- Overview of the Tensorflow.js library API with examples
  - The Ops API
  - The Layers API
- Building Handtrack.js - a javascript library for prototyping gesture based interactions in the browser.

Why Machine Learning in the Browser?

But What is Machine Learning?

Let's review some terminology ..

Artificial Intelligence

The broad field that describes efforts to make machines intelligent.

Machine Learning

Algorithms that enable machines to **learn** from data and **make predictions**

Decision trees, random forests, reinforcement learning, neural networks,

Deep Learning

Neural Networks (NN): a set of stacked computation units.

- High accuracy, scales with data
- Advances in NN algorithms
- Amenable to GPU computation

Artificial Intelligence

The broad field that describes efforts to make machines intelligent.

Machine Learning

Algorithms that enable machines **independently** learn from data and **make predictions**.

Decision trees, random forests, reinforcement learning, neural networks,

Deep Learning

Neural Networks (NN): a set of stacked computation units.

- High accuracy, scales with data
- Advances in NN algorithms
- Amenable to GPU computation

Artificial Intelligence

The broad field that describes efforts to make machines intelligent.

Machine Learning

Algorithms that enable machines **independently** learn from data and make **predictions**.

Decision trees, random forests, reinforcement learning, neural networks,

Deep Learning

Neural Networks (NN): a set of stacked computation units.

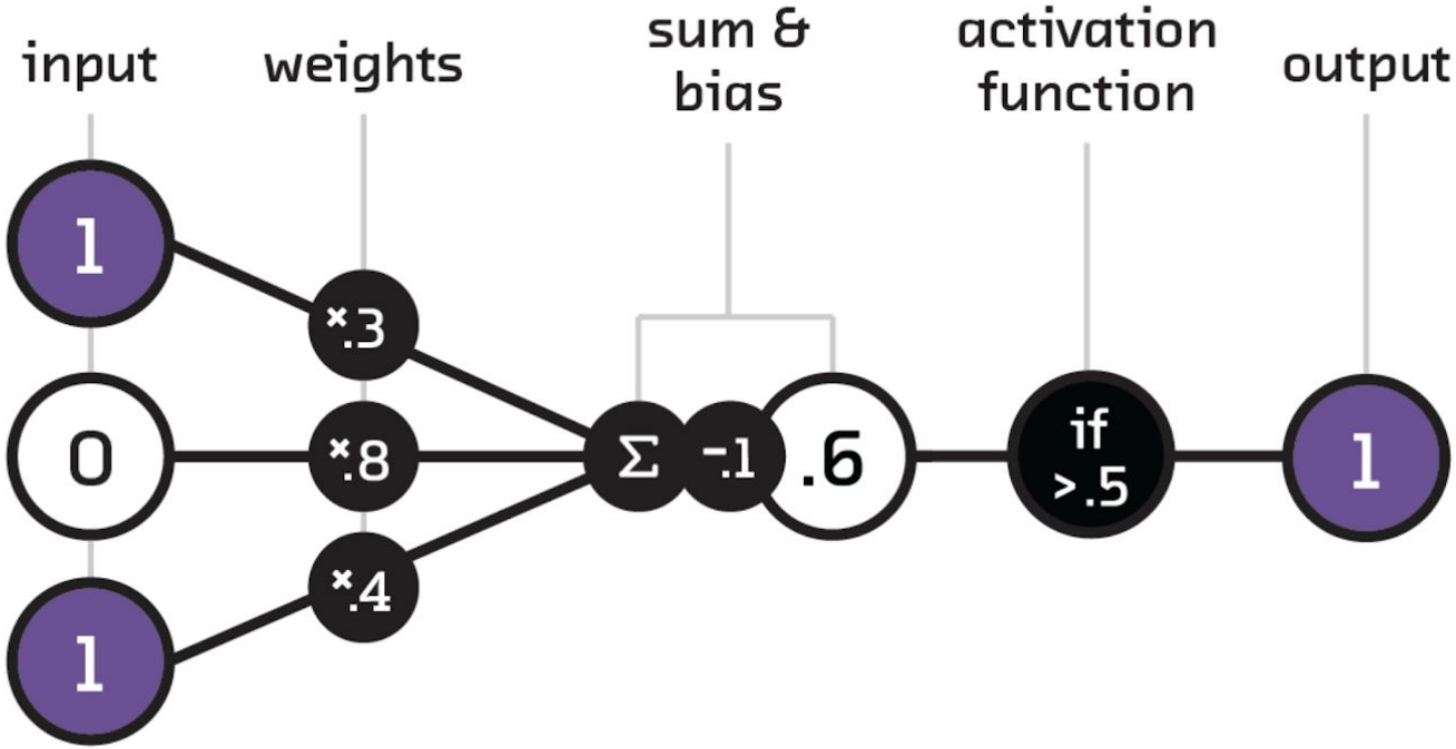
- High accuracy, scales with data
- Advances in NN algorithms
- Amenable to GPU computation

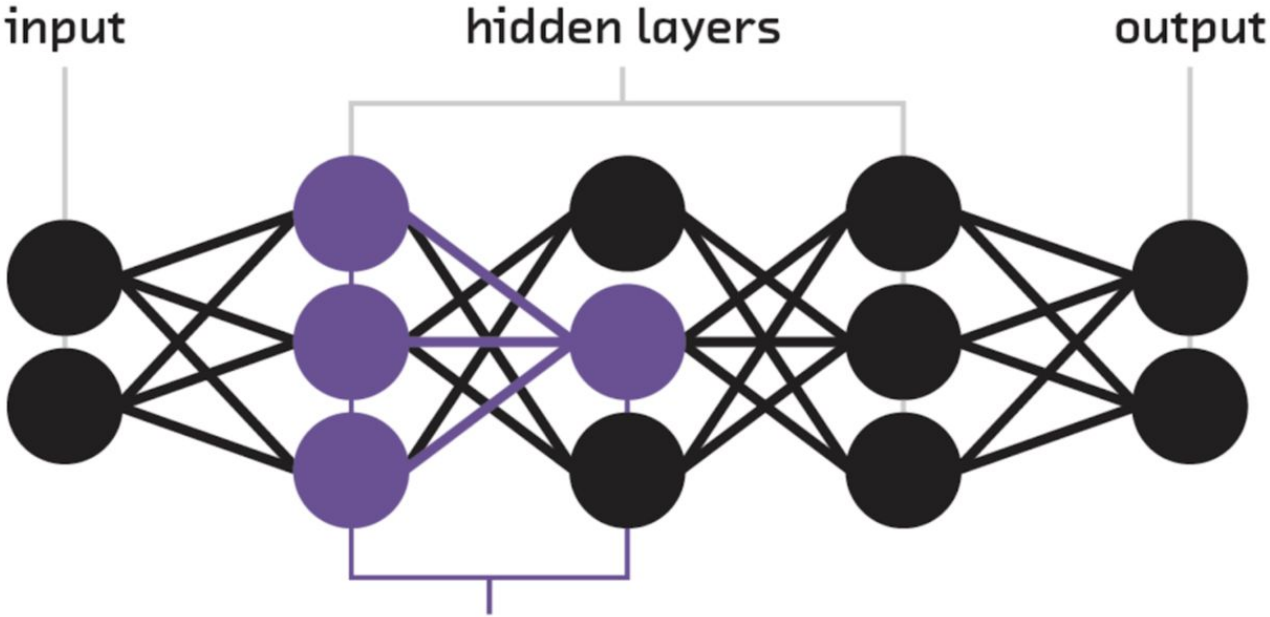
For today, each time I refer to Machine Learning or AI, what I am referring to is

Deep Learning

Fast Forward Labs

Perceptron





Dense layer (nodes in one layer connected to all nodes in the following layer).

The Languages of Machine Learning

The Languages of Machine Learning

python

R

julia

Matlab

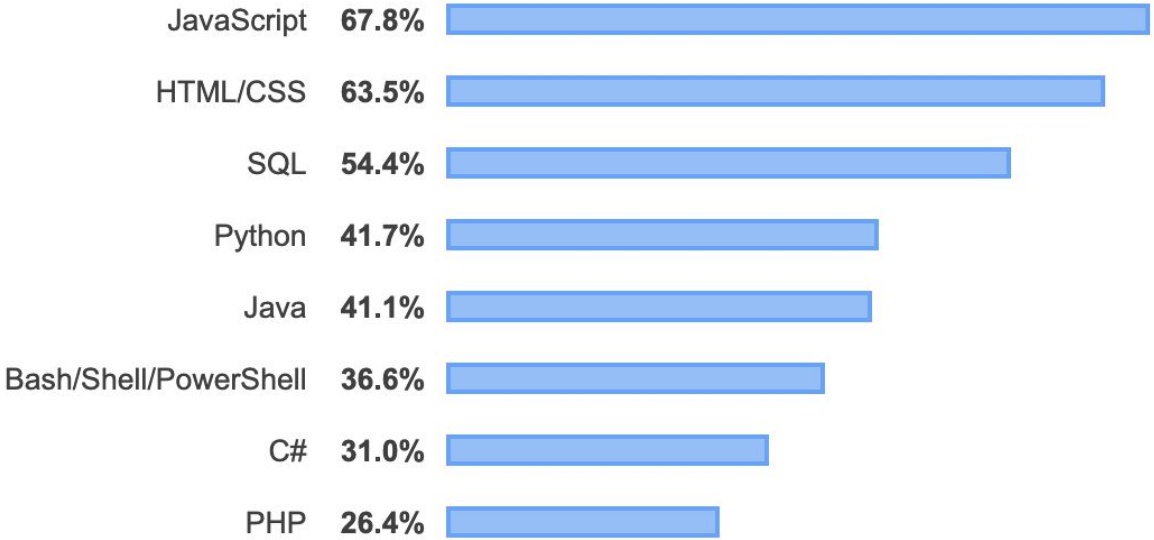
The Languages of Machine Learning

Go

The Languages of Machine Learning

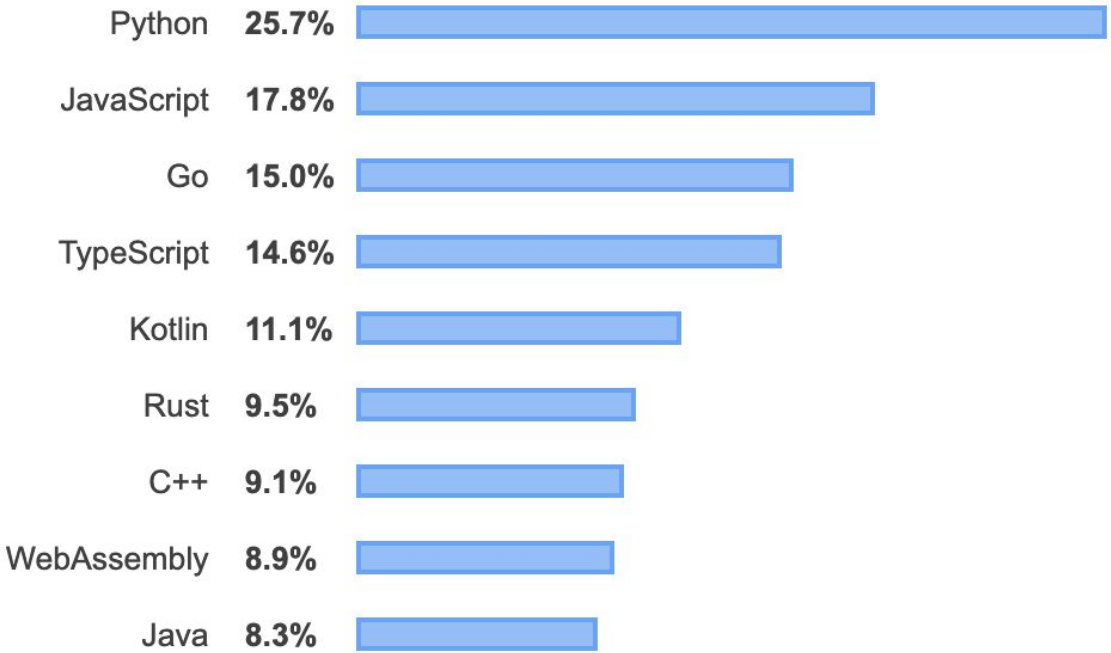
Backend programming languages
.. and they work well!
.. but ..

What programming language do developers use?



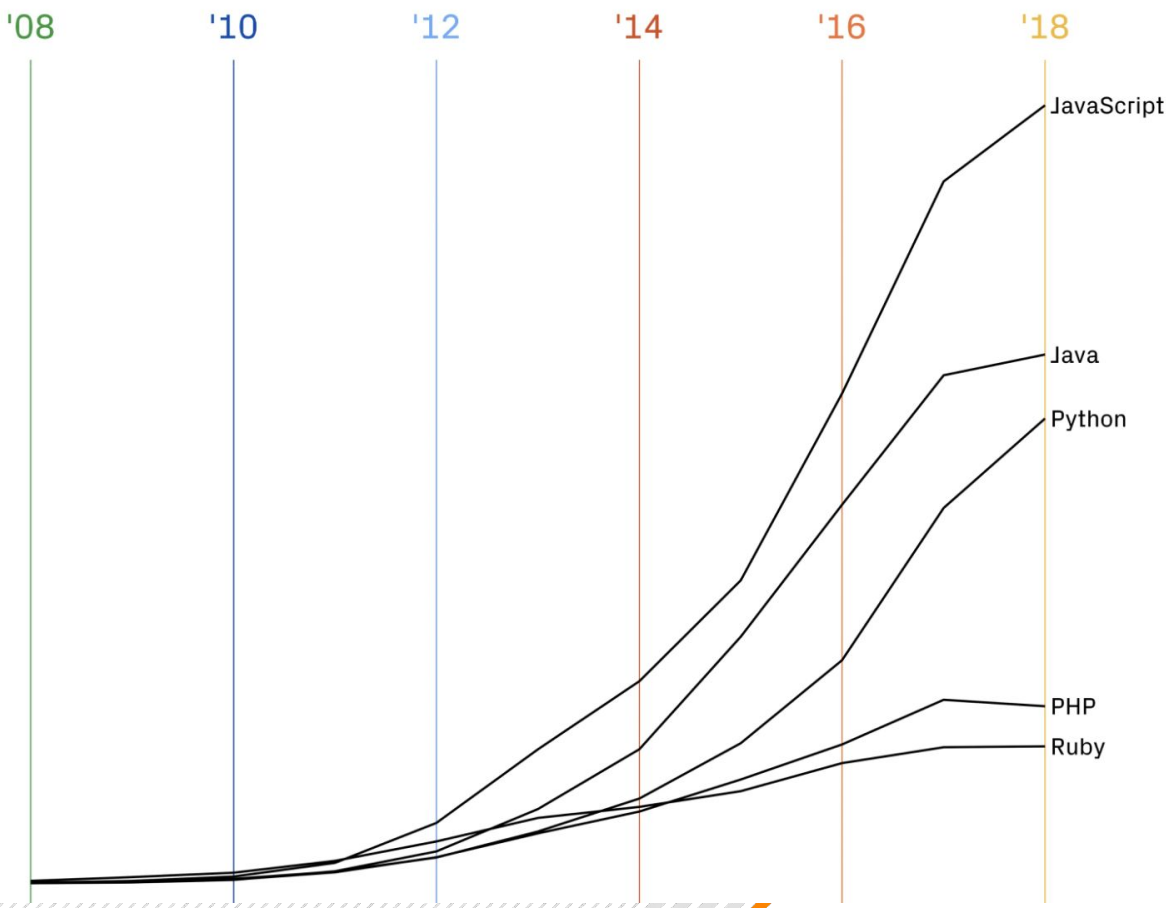
2019 Stackoverflow survey of 90k developers

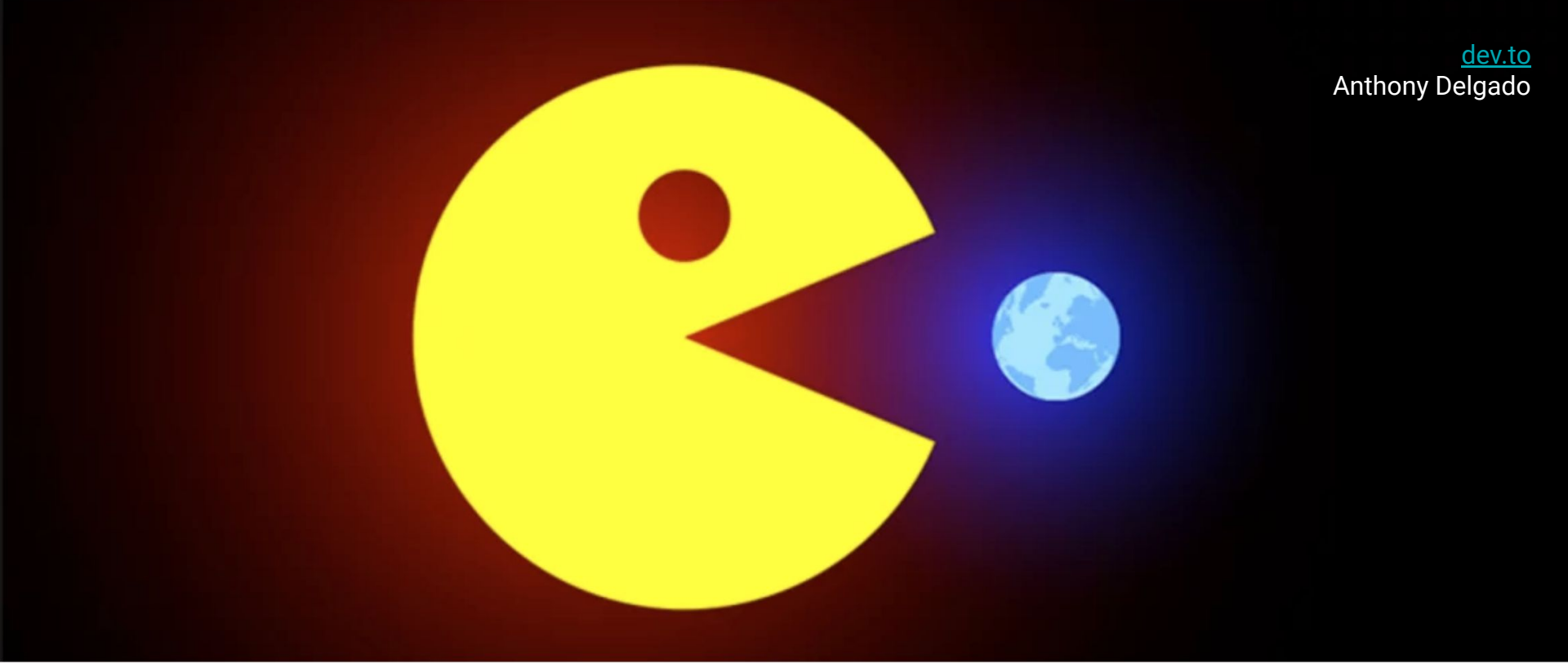
What programming language do developers want to learn?



2019 Stackoverflow survey of 90k developers

Github 2018 Top Programming languages between 2008 and 2018.

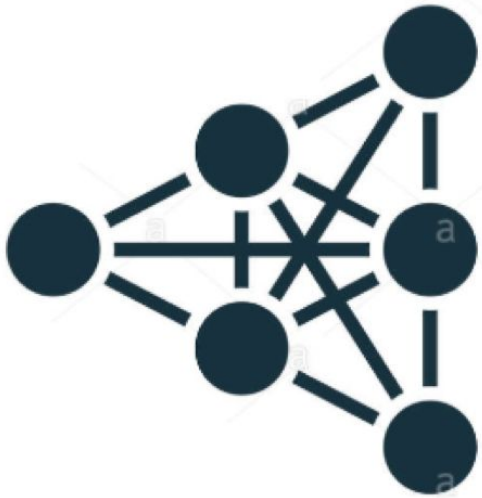




JavaScript Is Eating The World

Anthony Delgado Aug 24 '17 · 3 min read

+



Backend

Node.js

Front End

Vanilla js, Frameworks

Tenor
Bad Idea

Orienta
Boy thinking

Can I really train a neural network in Javascript?

Yes!!

Tensorflow.js - a library for building/training ML models in Javascript.

Will it meet my speed requirements?

Yes!!

Tensorflow.js, is accelerated in the browser using **WebGL** (CPU + GPU) and in Node.js using **Tensorflow C API** (CPU + GPU).

How much effort is this? Can I re-use my existing models/pipelines?

Not much!

- Tensorflowjs API is Similar to Keras
- Tensorflow.js **converter** allows you import pretrained **Keras** (hd5, savedModel), **Tensorflow** SavedModel, **Tensorflow Hub** Module

Oprah
“You get a car”

Why Machine Learning in the Browser?

- Privacy
- Distribution
- Latency
- Interactivity

Why Machine Learning in the Browser?

- **Privacy**
- Distribution
- Latency
- Interactivity

- **Strong Notion of Privacy**
  We do not see your data, it never gets to our servers.

- Example use case: sensitive documents identification, domain specific text autocomplete for chat systems etc

Why Machine Learning in the Browser?

- Privacy
- **Distribution**
- Latency
- Interactivity

- **No installs, drivers and dependency issues.**
- Navigate to a URL and your application is available to end users.
- Free model asset hosting/versioning/distribution (NPM/jsdelvr), excellent for end user applications.

Why Machine Learning in the Browser?

- Privacy
- Distribution
- **Latency**
- Interactivity

- **ML models can be optimized for speed and can run fast on mobile and in the browser**

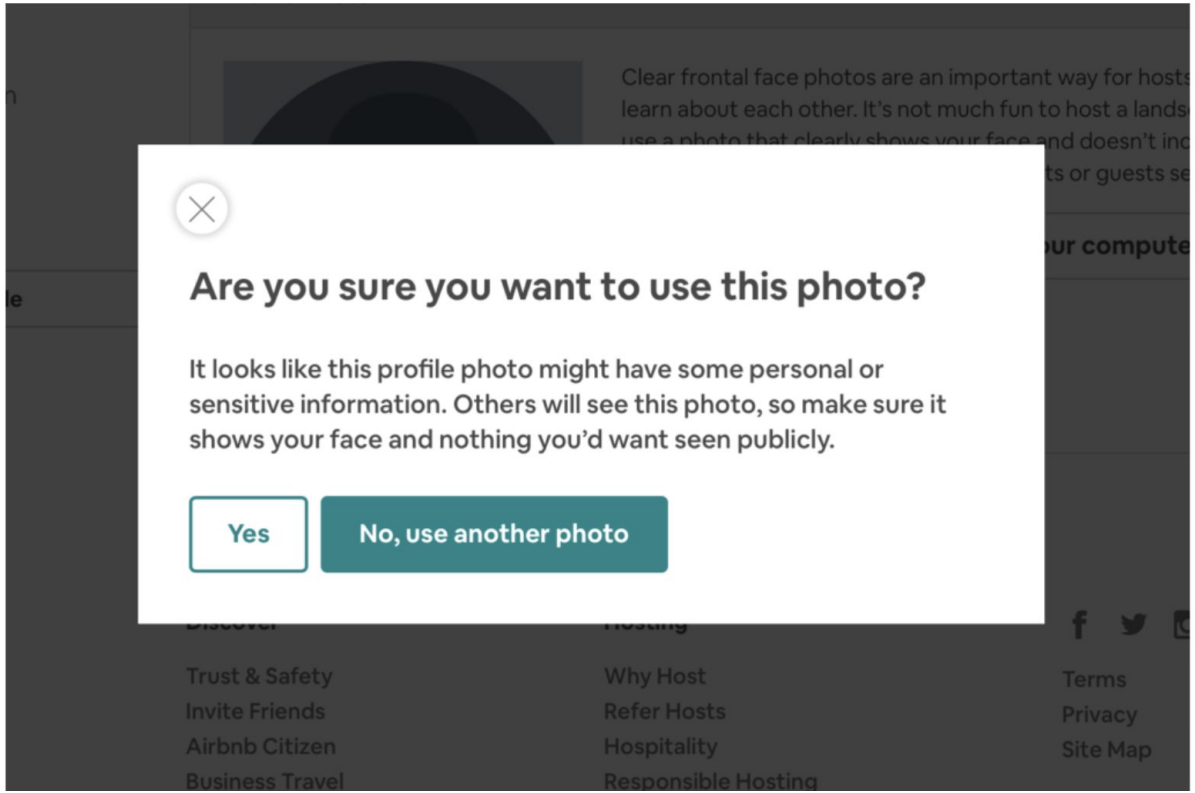
- In some cases, it is faster to run your model in the browser, rather than send make round trip requests to a server.
- Avoid reliability issues with intermittent connectivity environments

Why Machine Learning in the Browser?

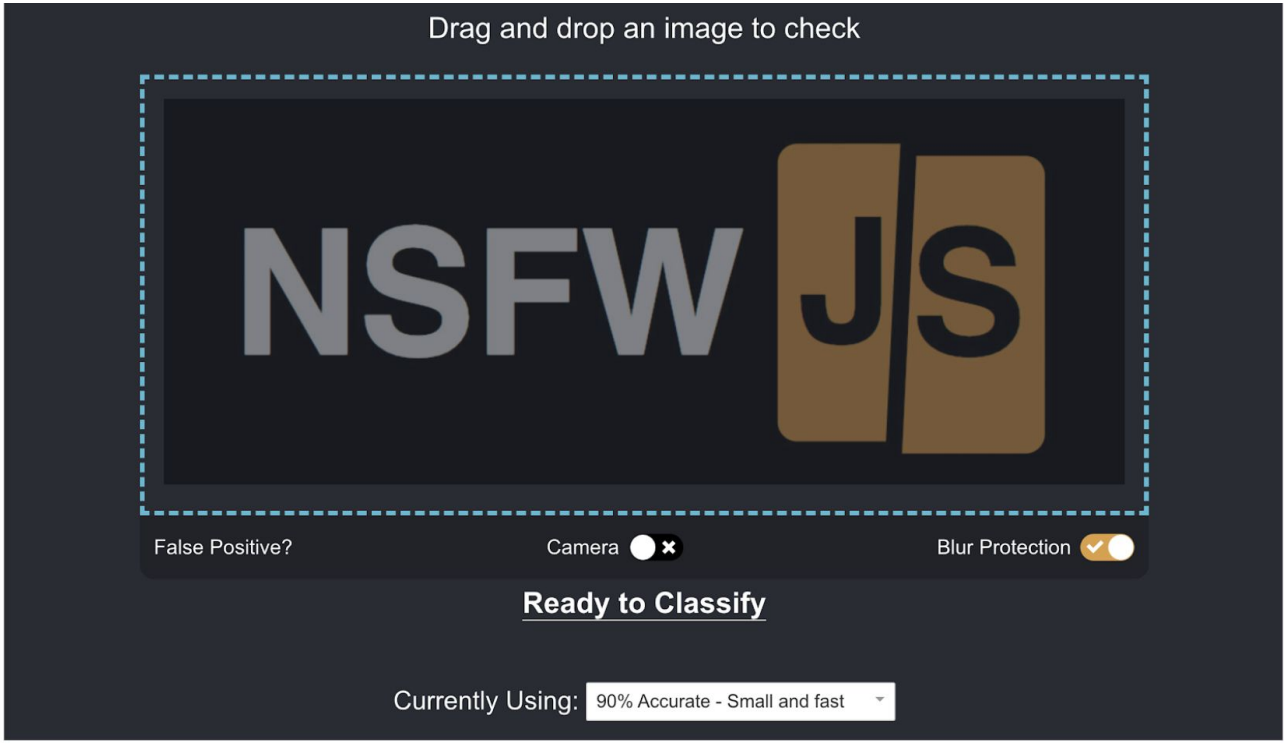
- Privacy
- Distribution
- Latency
- **Interactivity**

- **The Browser is designed for interactive experiences. ML can supercharge this!**

- Build models on the fly using rich user data available in the browser (camera, file uploads), run inference, retrain existing models, enable dynamic behaviour.
- ML education, retail, advertising, arts, entertainment, gaming.



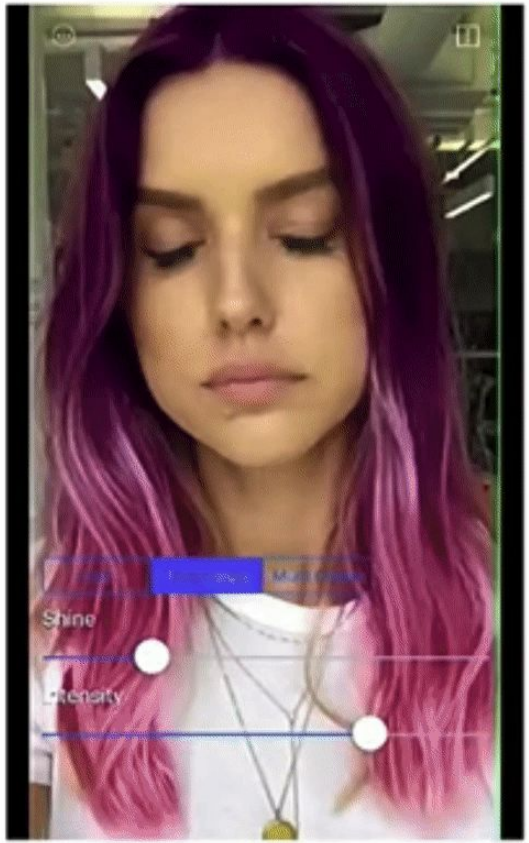
Privacy Preserving Sensitive Content Detection.

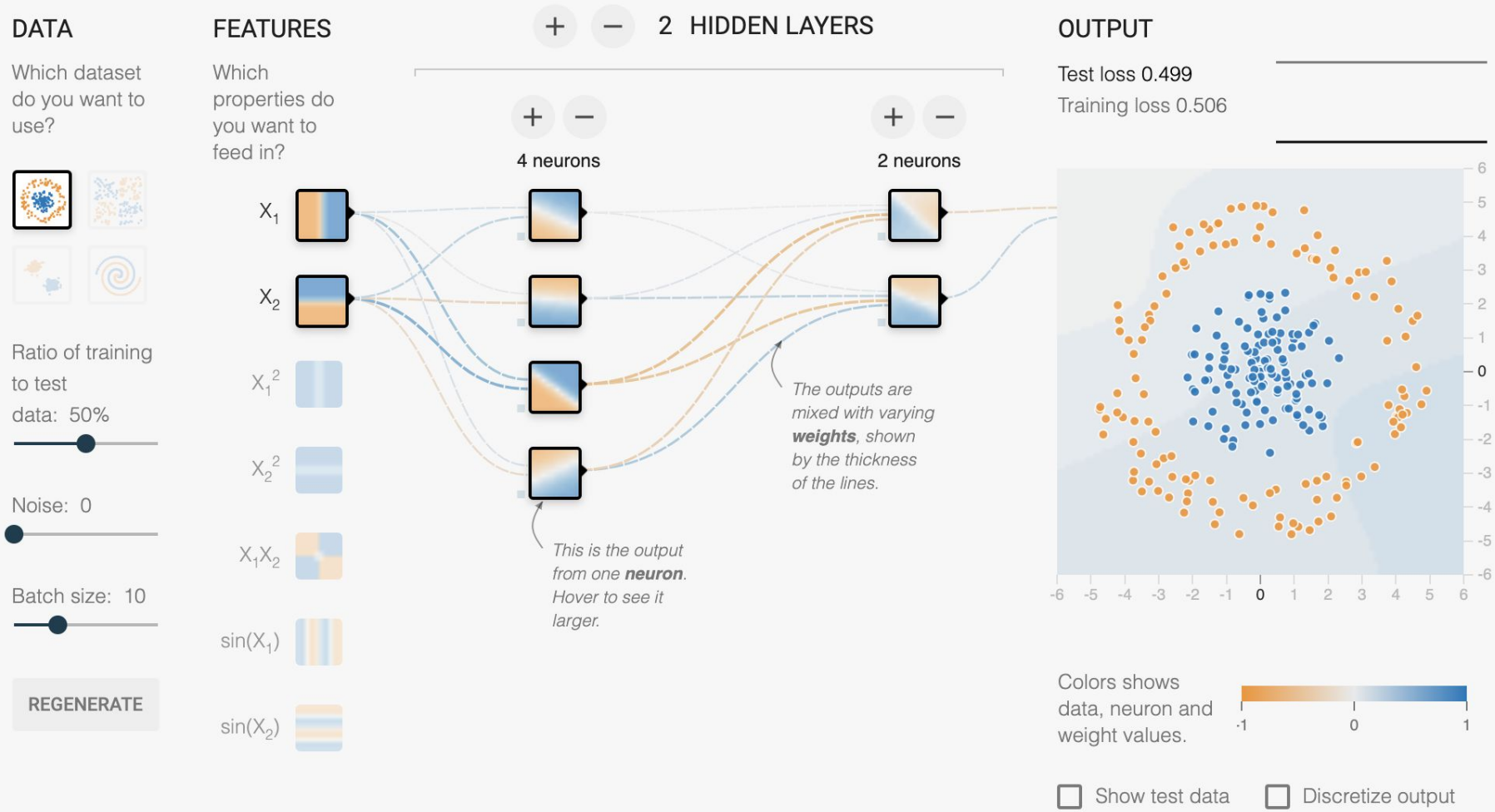


Client-side indecent content checking nsfw.js.com

Wechat mini-app

- ModiFace virtual try-on app: 1.8MB。
  - JavaScript file: 983KB
  - Model Files: 829KB
- ModiFace virtual try-on speed: 25FPS。
  - Model inference time: 30ms
  - UI render:10ms
- Smallest fastest virtual try-on app so far




DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
X1
X2
X1²
X2²
X1X2
sin(X1)
sin(X2)
2 HIDDEN LAYERS
4 neurons
2 neurons
This is the output from one neuron. Hover to see it larger.
The outputs are mixed with varying weights, shown by the thickness of the lines.
OUTPUT
Test loss 0.499
Training loss 0.506
Colors shows data, neuron and weight values.
-1
0
1
Show test data
Discretize output

Tensorflow.js

A library for machine learning in JavaScript

What can you do with Tensorflow.js

- **Author models**
- Import pre-trained models for inference
- Re-train imported models

Online Flow

- Design/Train/Test/ Inference, all in your javascript code.

What can you do with Tensorflow.js

- Author models
- **Import pre-trained models for inference**
- Re-train imported models

Offline Flow

- Design/Train/Test model on your preferred hardware (GPU, TPU clusters)
- Import into javascript for inference

What can you do with Tensorflow.js

- Author models
- Import pre-trained models for inference
- **Re-train imported models**

Hybrid Flow

- Design/Train/Test model on your preferred hardware (GPU, TPU clusters)
- Import into javascript application
- Retrain/Run inference

Tensorflow.js - Backend and Frontend

Browser
(Front end)

Server
(Back end, Node.js)

Tensorflow.js - Backend and Frontend

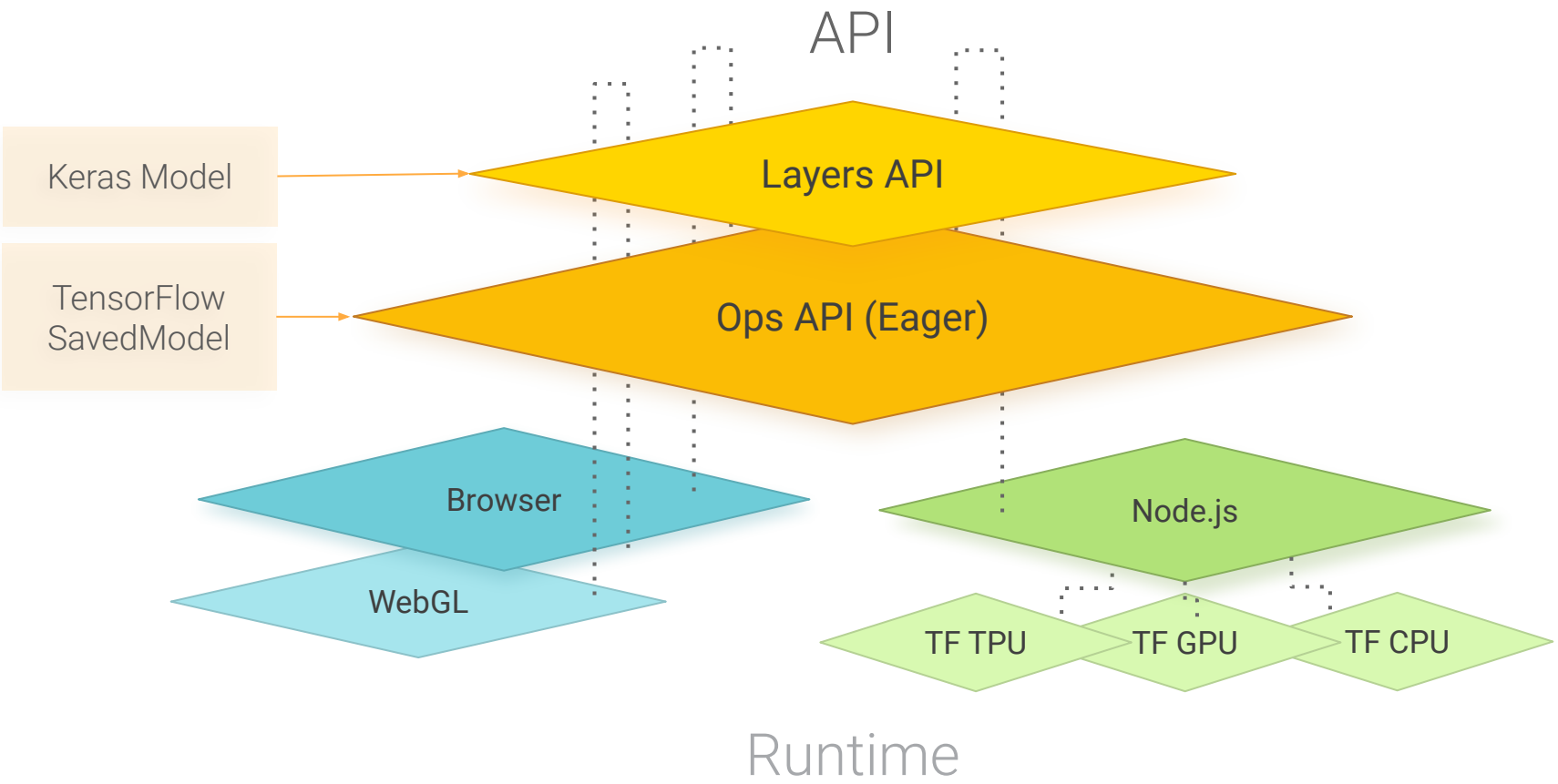
Browser
(Front end)

- Usage
  - Vanilla js script tag import (like jquery)
  - Import - NPM install (build tools, frameworks such as React, Vue, etc).
- Acceleration
  - **WebGL**
  - Tensors implemented as Shader programs (will use GPUs where available)

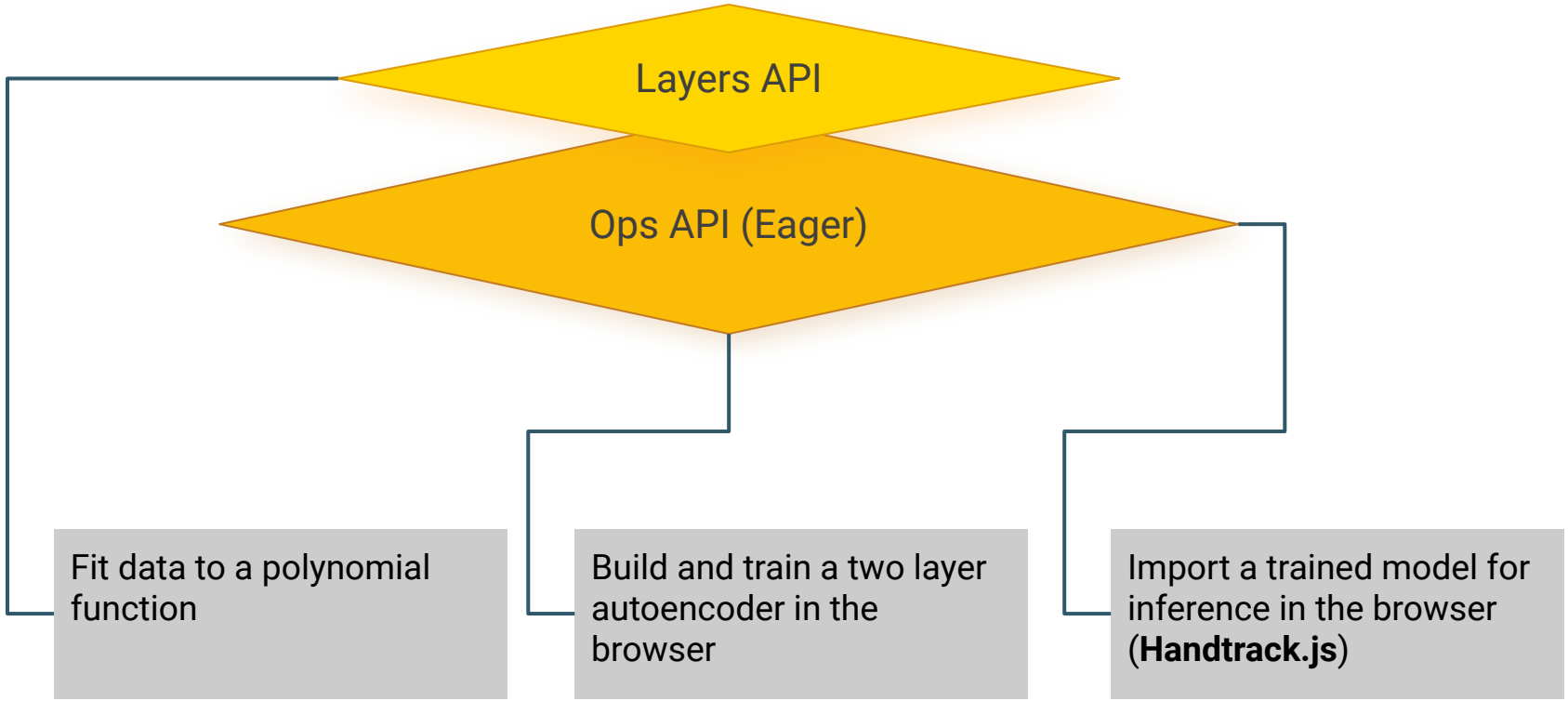
Tensorflow.js - Backend and Frontend

- Usage
  - NPM install
  - (tfjs-node, tfjs-node-gpu)
- Acceleration
  - **Tensorflow C** binary

Server
(Back end, Node.js)


API
Keras Model
TensorFlow SavedModel
Layers API
Ops API (Eager)
Browser
WebGL
Node.js
TF TPU
TF GPU
TF CPU
Runtime

API

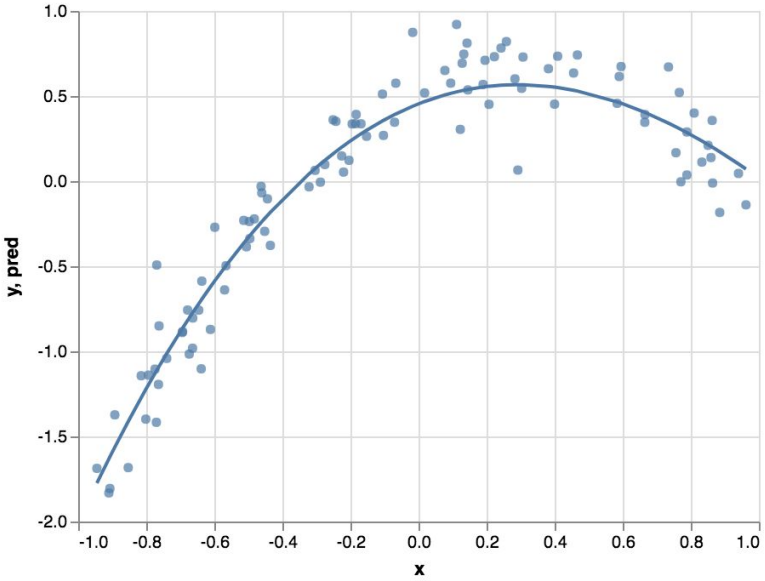

Layers API
Ops API (Eager)
Fit data to a polynomial function
Build and train a two layer autoencoder in the browser
Import a trained model for inference in the browser (**Handtrack.js**)

Tensorflow.js OPS API

Use with caution

The **Ops API** - Fit a Polynomial

$f(x) = ax^2 + bx + c$



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

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

const a = tf.tensor(0.1).variable();
const b = tf.tensor(0.1).variable();
const c = tf.tensor(0.1).variable();
```

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

const a = tf.tensor(0.1).variable();
const b = tf.tensor(0.1).variable();
const c = tf.tensor(0.1).variable();

// f(x) = a*x^2 + b^x + c
const f = x => a.mul(x.square()).add(b.mul(x)).add(c);
// ...
```

```
// ...

// Mean-squared error
const loss = (preds, label) =>
preds.sub(label).square().mean();
```

```
// ...

// Mean-squared error
const loss = (preds, label) =>
preds.sub(label).square().mean();
const optimizer = tf.train.sgd(learningRate);
```

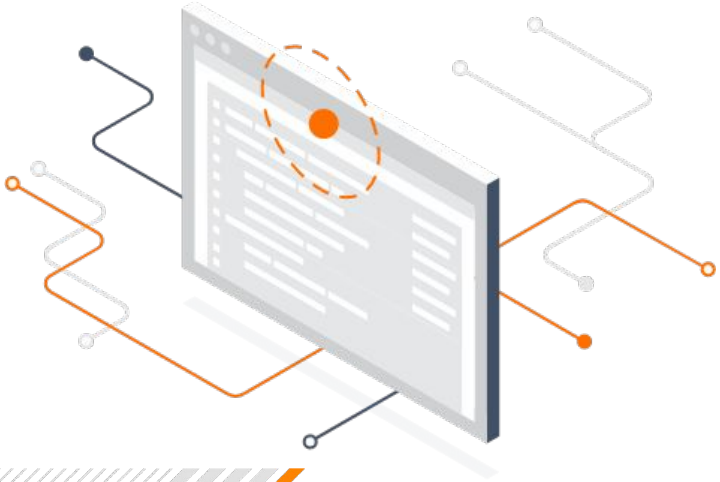
```
// ...

// Mean-squared error
const loss = (preds, label) =>
preds.sub(label).square().mean();
const optimizer = tf.train.sgd(learningRate);
for (let i = 0; i < EPOCHS; i++) {
  optimizer.minimize(() => loss(f(data.xs), data.ys));
}
```

Tensorflow.js Layers API

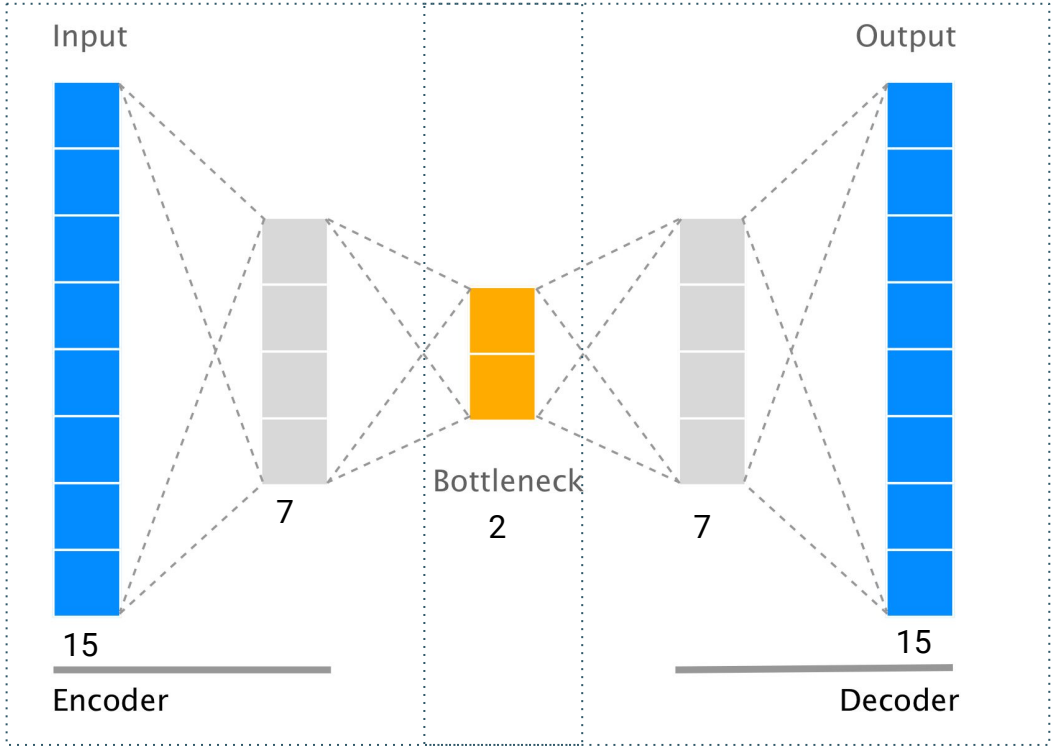
Very much like keras

The **Layers API**

- Recommended API for neural networks
- Similar in spirit to the **Keras Library**. Keras is a well designed api standard for composing and training neural networks.



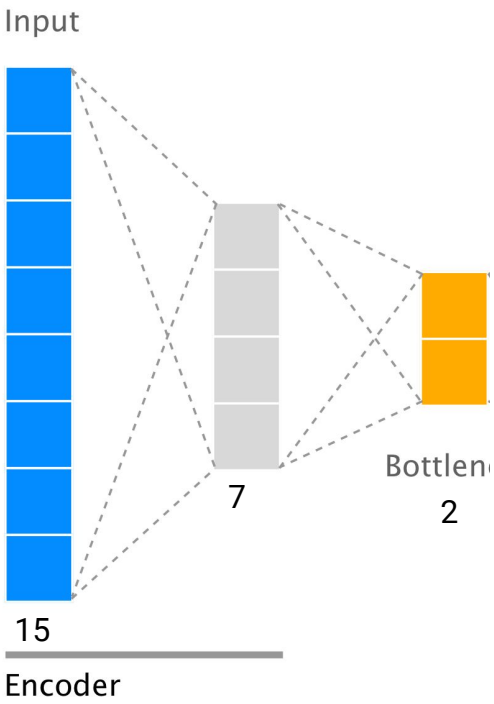
An Autoencoder



Dimensionality reduction

- Neural Network with two parts.
- Given some high dimension data (e.g. tabular data with many fields), find a smaller representation.
- This representation should support reconstruction.

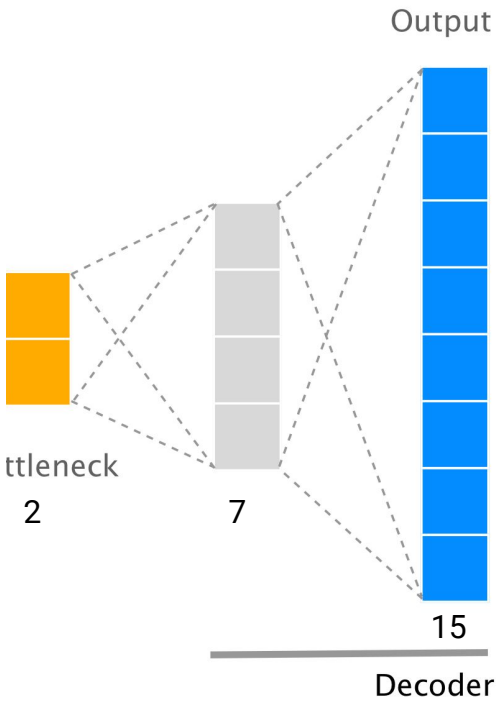
An Autoencoder



Encoder

- Take a high dimensional input.
- Learn to compress it to a *meaningful* smaller representation (bottleneck)

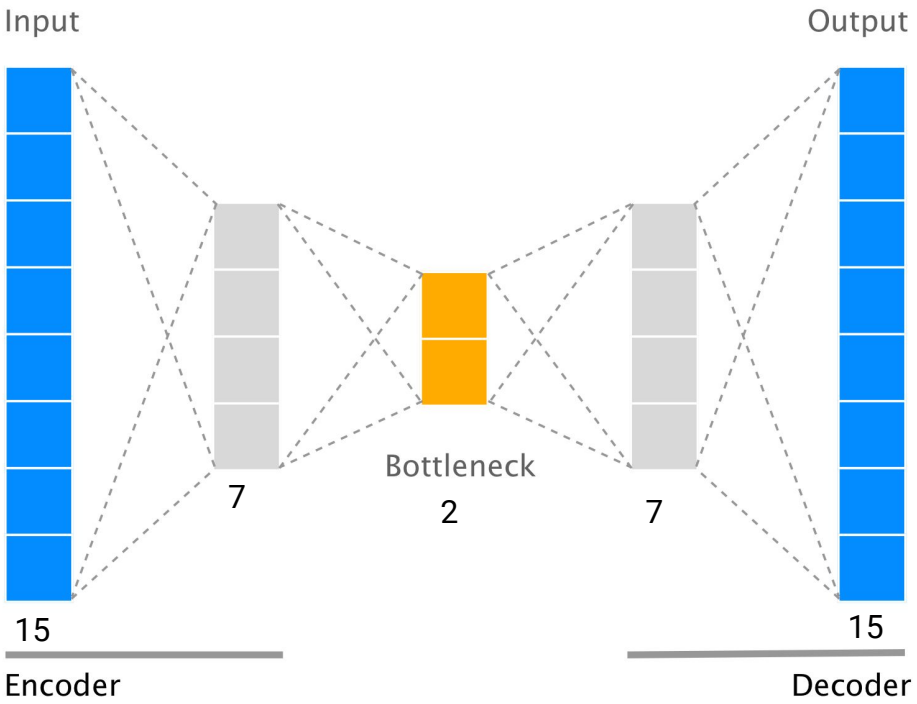
An Autoencoder



Decoder

- Takes a bottleneck vector
- Learn to reconstruct original input from z.

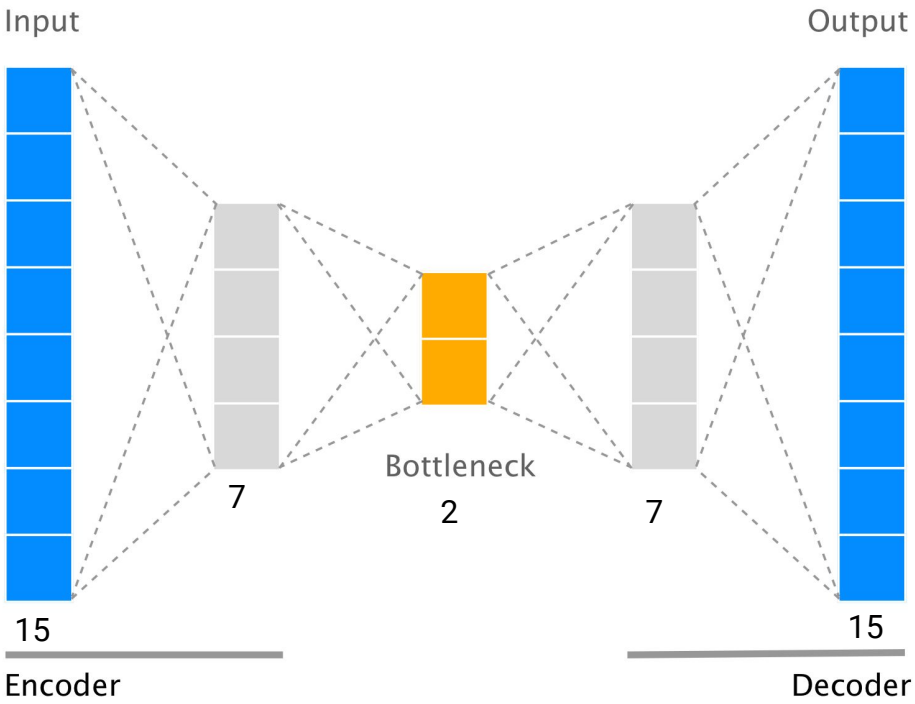
An Autoencoder for Detecting Anomalies

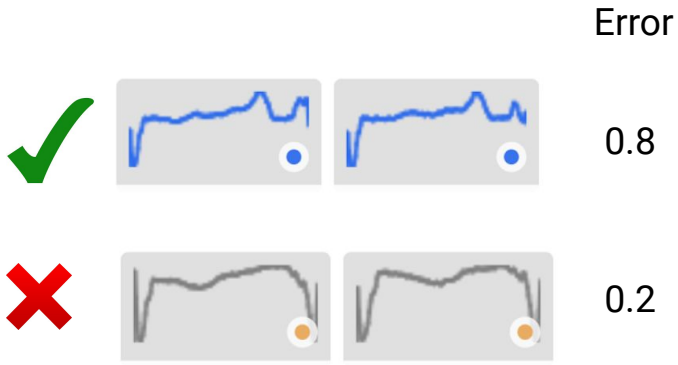


Anomaly Detection

- Train on **Normal Data**
- Learns a bottleneck vector suitable for reconstructing normal signals
- High reconstruction error (MSE) for abnormal images. Flag ***high error*** as anomalies.

An Autoencoder for Detecting Anomalies





Example, ECG data

```python
# Keras, Python

# Encoder
inputs =
Input(shape=(num_feature
s,))
# define hidden layers
```

```cpp
// Tensorflow JS

// Encoder
const input = tf.input({
shape: [numFeatures] })
# define hidden layers
```

```python
# Keras, Python

# define hidden layers
enc_hidden = Dense(15,
activation='relu')(inpu
ts)
```

```cpp
// Tensorflow JS

# define hidden layers
let encHidden =
tf.layers.dense({ units: 15,
activation: "relu"
}).apply(input);
```

```python
# Keras, Python

# define hidden layers
enc_hidden = Dense(7,
activation='relu')
(enc_hidden)
```

```javascript
// Tensorflow JS

# define hidden layers
let encHidden =
encHidden =
tf.layers.dense({ units: 7,
activation: "relu"
}).apply(encHidden);
```

```python
# Keras, Python

# Encoder

z_ =
Dense(latent_dim)(enc_hidden)

encoder = Model(inputs, z_,
name='encoder')
```

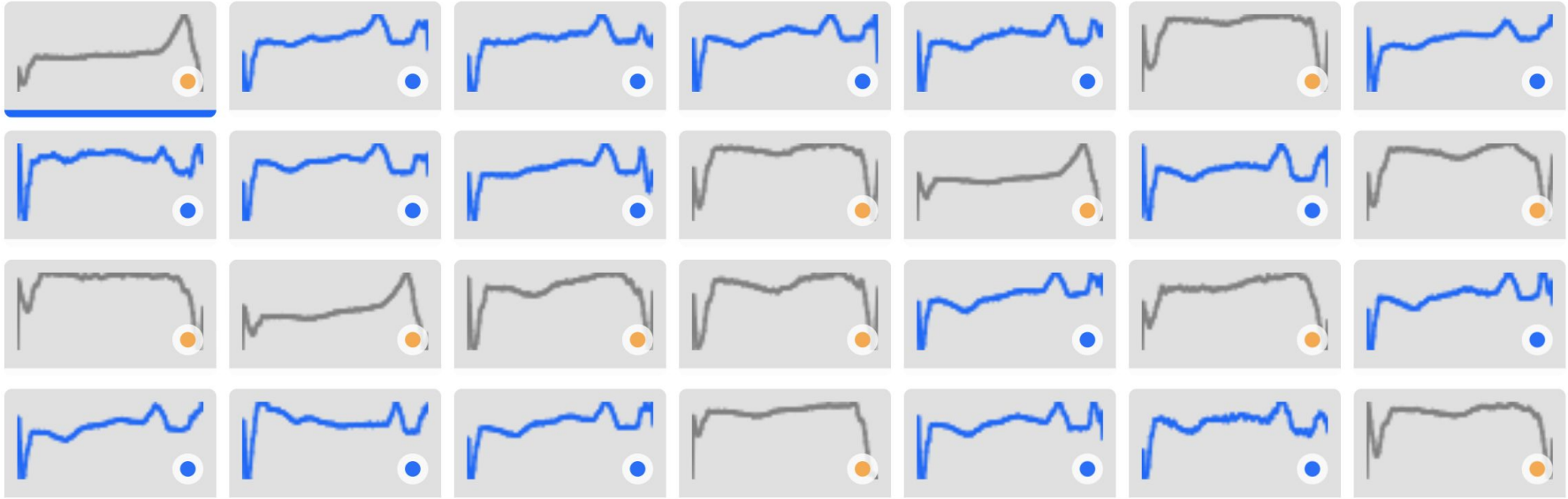
```javascript
// Tensorflow JS

// Encoder

const z_ = tf.layers.dense({
units: latentDim
}).apply(encHidden);
const encoder = tf.model({
inputs: input, outputs: z_,
name:"encoder" })
```

```
Model: "encoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
encoder_input (InputLayer)   (None, 140)               0
_________________________________________________________________
encoder_hidden_0 (Dense)     (None, 15)                2115
_________________________________________________________________
encoder_hidden_1 (Dense)     (None, 7)                 112
_________________________________________________________________
z_ (Dense)                   (None, 2)                 16
=================================================================
Total params: 2,243
Trainable params: 2,243
Non-trainable params: 0
_________________________________________________________________
None
Model: "decoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
z_ (InputLayer)              (None, 2)                 0
_________________________________________________________________
decoder_hidden_0 (Dense)     (None, 7)                 21
_________________________________________________________________
decoder_hidden_1 (Dense)     (None, 15)                120
_________________________________________________________________
decoder_output (Dense)       (None, 140)               2240
=================================================================
Total params: 2,381
Trainable params: 2,381
Non-trainable params: 0
```

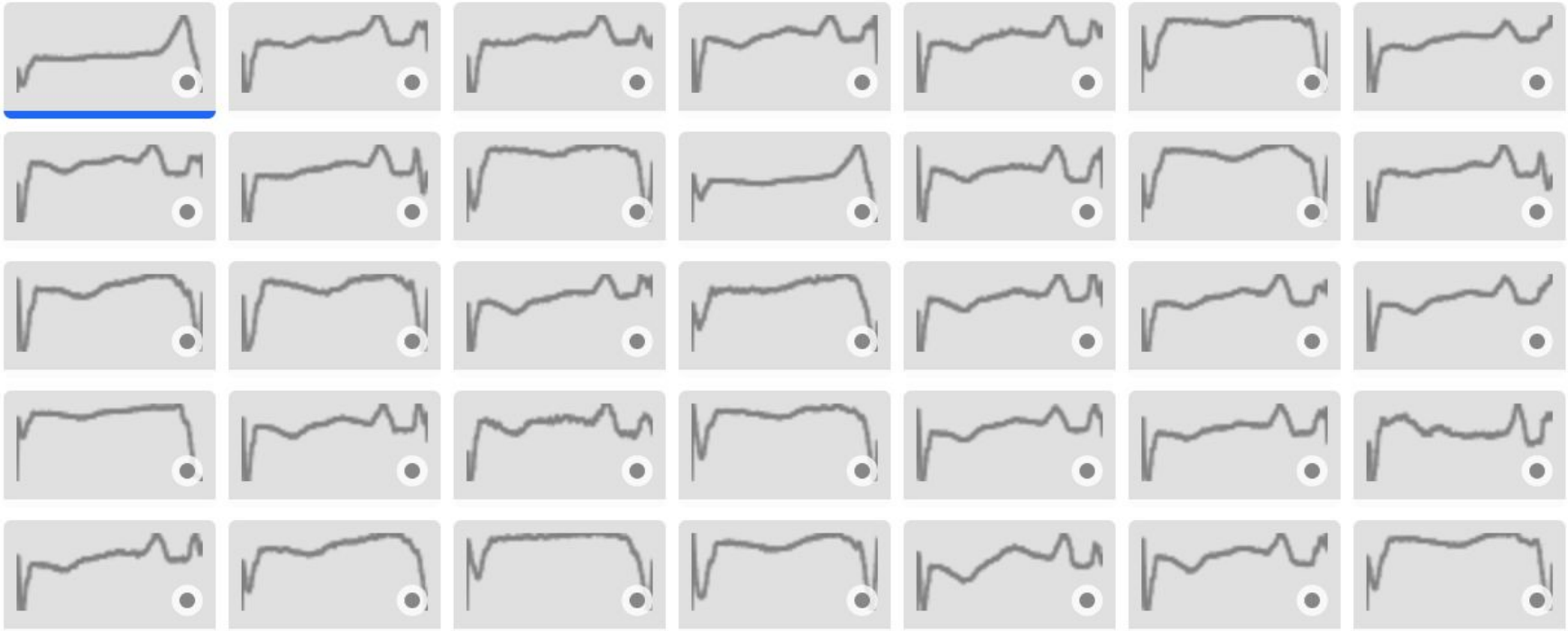
```
_________________________________________________________________
Layer (type)                 Output shape              Param #
=================================================================
input1 (InputLayer)          [null,140]                0
_________________________________________________________________
dense_Dense1 (Dense)         [null,15]                 2115
_________________________________________________________________
dense_Dense2 (Dense)         [null,7]                  112
_________________________________________________________________
dense_Dense3 (Dense)         [null,2]                  16
=================================================================
Total params: 2243
Trainable params: 2243
Non-trainable params: 0
_________________________________________________________________
_________________________________________________________________
Layer (type)                 Output shape              Param #
=================================================================
input2 (InputLayer)          [null,2]                  0
_________________________________________________________________
dense_Dense4 (Dense)         [null,7]                  21
_________________________________________________________________
dense_Dense5 (Dense)         [null,15]                 120
_________________________________________________________________
dense_Dense6 (Dense)         [null,140]                2240
=================================================================
Total params: 2381
Trainable params: 2381
Non-trainable params: 0
```

Train - Data

- Normal
- R-on-T Premature Ventricular Contraction
- Supraventricular Premature or Ectopic Beat
- Premature Ventricular Contraction
- Unclassifiable Beat



Test - Data



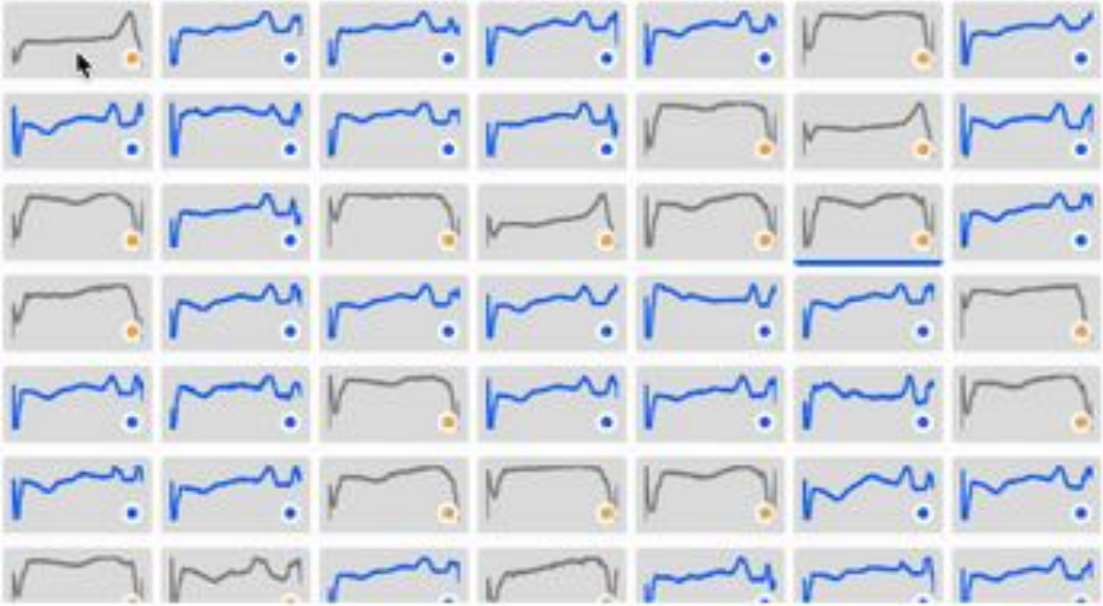
Anomagram | Introduction | Train a Model | FAQ

Anomaly Detection with Deep Learning in the Browser!

Anomagram is an interactive visualization tool for exploring deep learning models applied to the task of anomaly detection (on non-time series data).

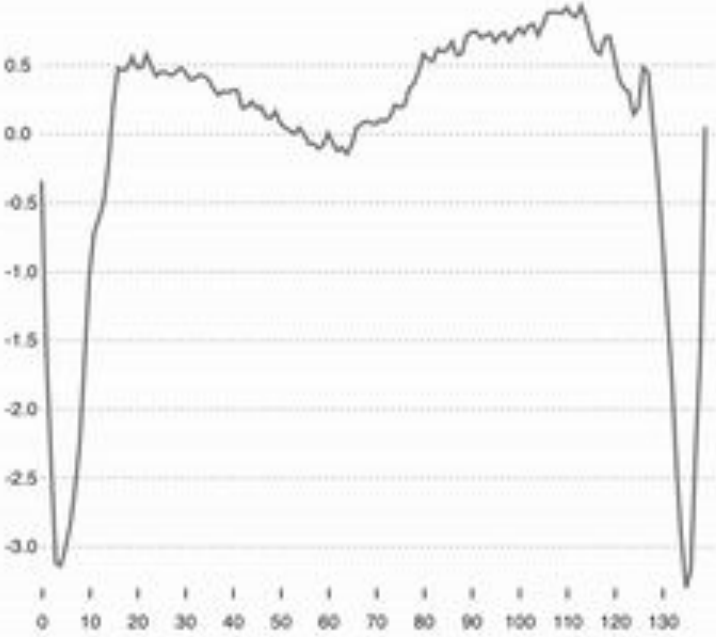
ECG 200

- Normal
- R-on-T Premature Ventricular Contraction
- Supraventricular Premature or Ectopic Beat
- Premature Ventricular Contraction
- Unclassifiable Beat

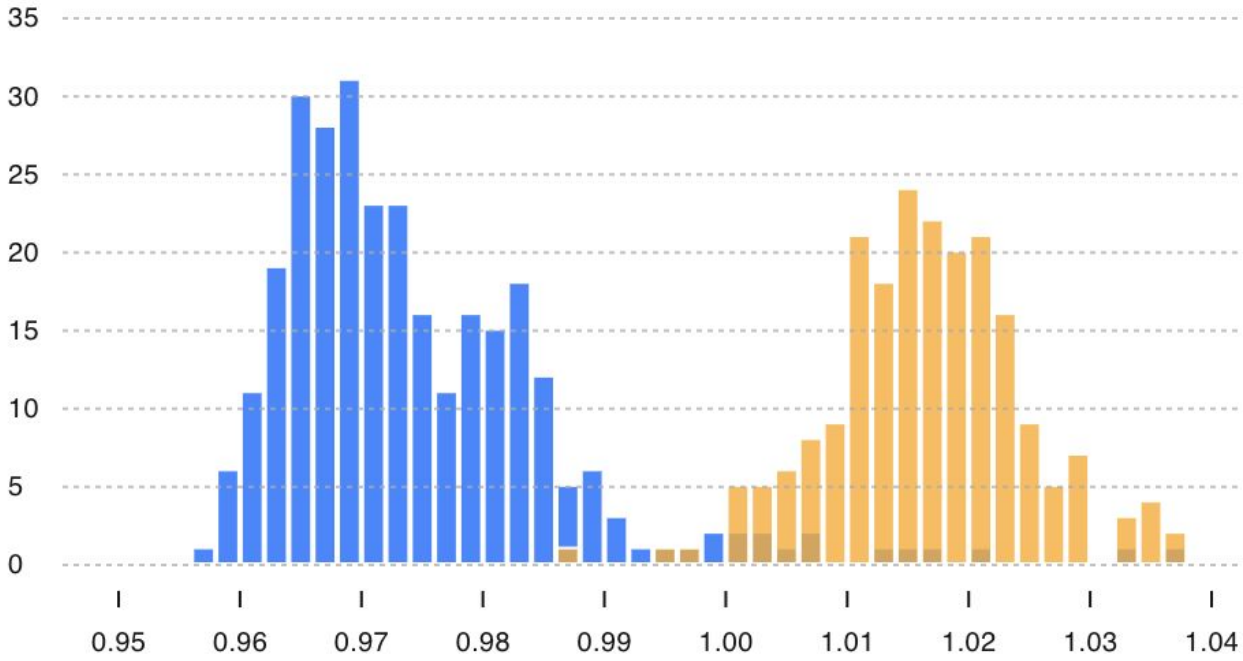


Model Output

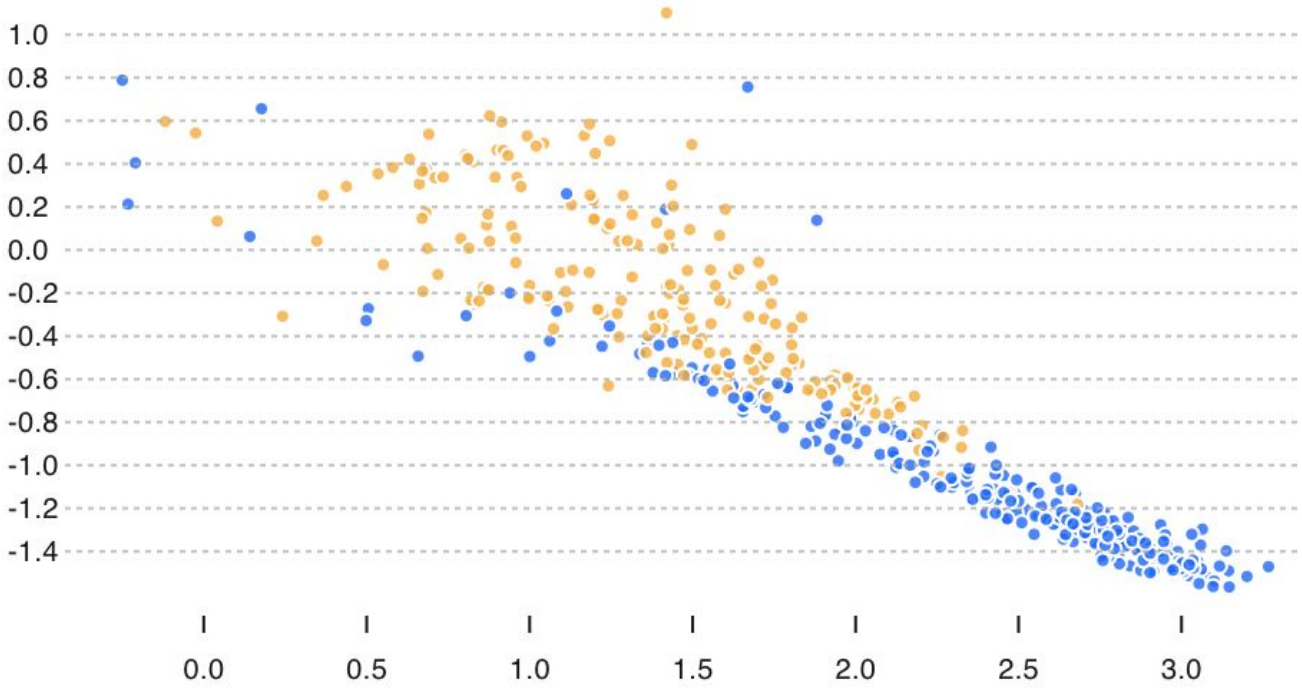
ABNORMAL



Interactive Visualizations of Predictions

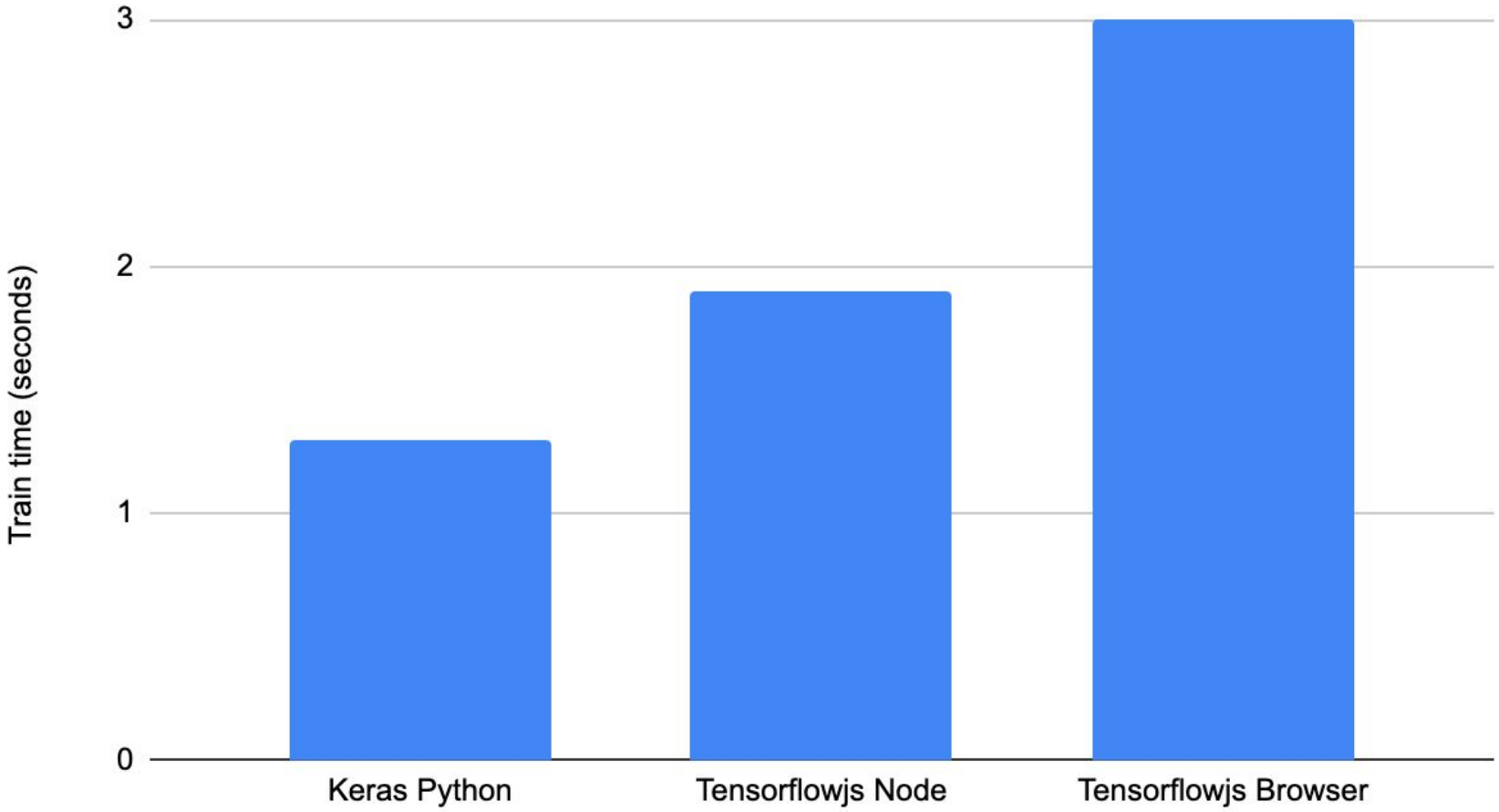


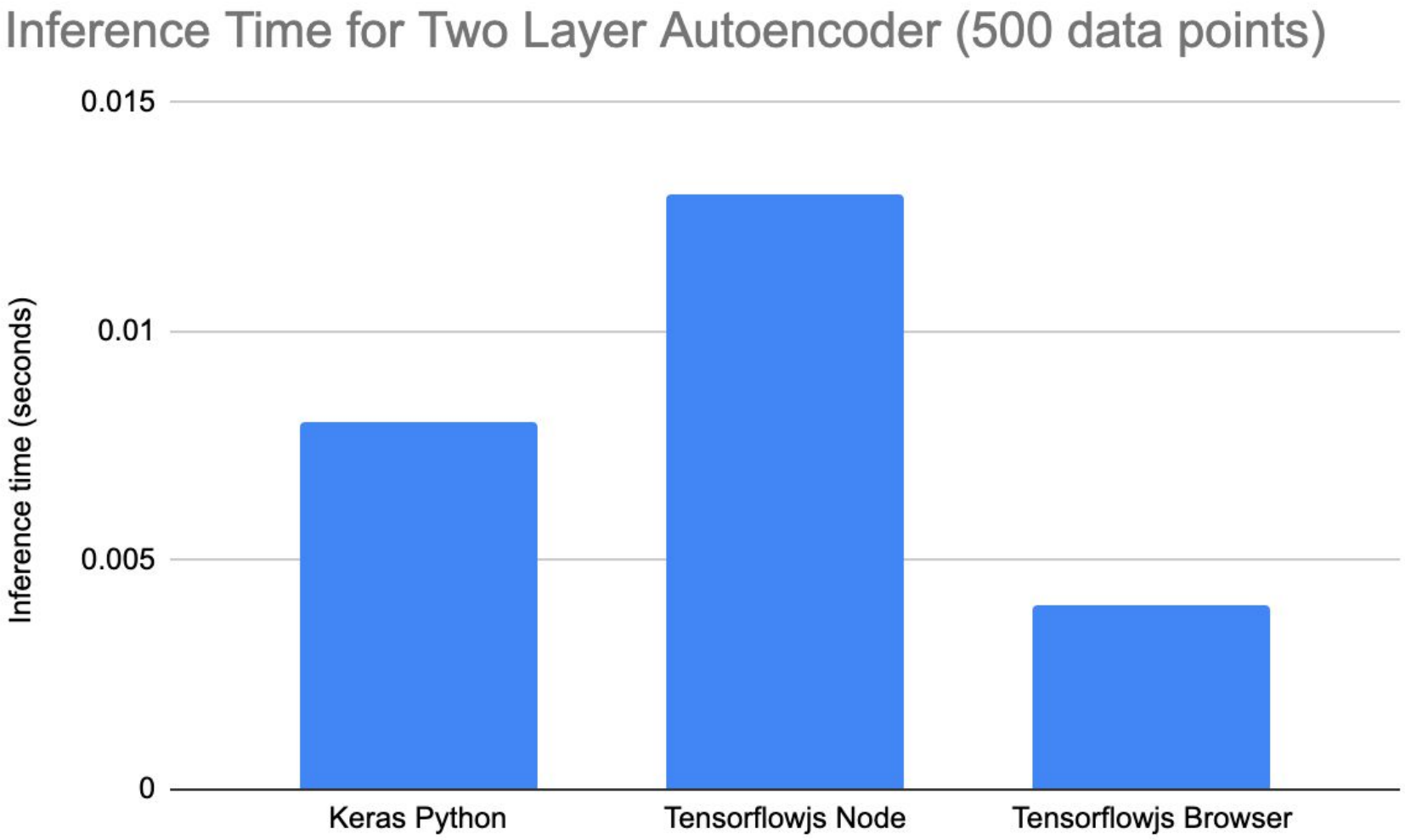
Interactive Visualizations of Bottleneck Predictions



Train Time for Two Layer Autoencoder (20 epochs)

2600 data points




Inference Time for Two Layer Autoencoder (500 data points)
Inference time (seconds)
0.015
0.01
0.005
0
Keras Python
Tensorflowjs Node
Tensorflowjs Browser

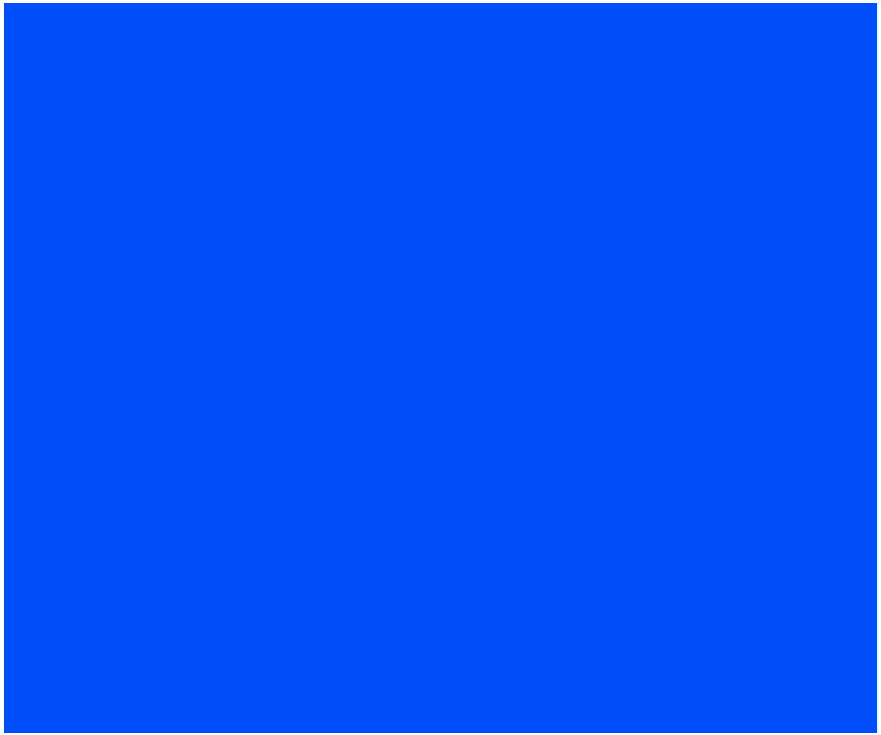
Building Handtrack.js,

A library for prototyping hand gestures in the browser

https://github.com/victordibia/handtrack.js/

Handtrack.js

- Given an image, predict the location (bounding box) of all hands.
- Uses a **Convolutional Neural Network** (Object Detection)
- Bundled into a JS library to enable developers prototype interactive.
- Open source.



```
<script
src="https://cdn.jsdelivr.net/npm/handtrackjs/dist/handtrack.min.js">
</script>
```

```
// Load the model.
handTrack.load().then(model => {
    // detect objects in the image.
    model.detect(img).then(predictions => {
        console.log('Predictions: ', predictions);
    });
});
```

```
[{
bbox: [x, y, width, height],
class: "hand",
score: 0.8380282521247864
},
{ bbox: [x, y, width, height],
class: "hand",
score: 0.74644153267145157
}]
```

How was this built?

How was this built?

Data Assembly

- Egohands dataset + Other sources

Model Training

TensorFlow

- Tensorflow Object Detection Api.
- Export trained model (saved model, frozenmodel)

Model Conversion

TensorFlow.js

A JavaScript library for training and deploying ML models in the browser and on Node.js

- Convert to tensorflowjs webmodel format
- Strip post processing step to improve performance

Data Assembly (Egohands Dataset)

Model Training

- Convert images to **Tensorflow Records format**
  - For each image, list of bbox coordinates, and labels.
- Train using Tensorflow **Object Detection API (python)**
  - Framework and sample code for training various object detection models.
  - Mobilenet, SSD (fast, optimized for mobile).
- Export Trained Model as **savedModel** from checkpoints.

Model Conversion

- Use tensorflow.js converter
  - Identify model output node variable name
  - Specify model format
  - Specify input/output directories
- Supports
  - **savedModel** (default for Tensorflow)
  - Keras Model
  - Tensorflow Hub

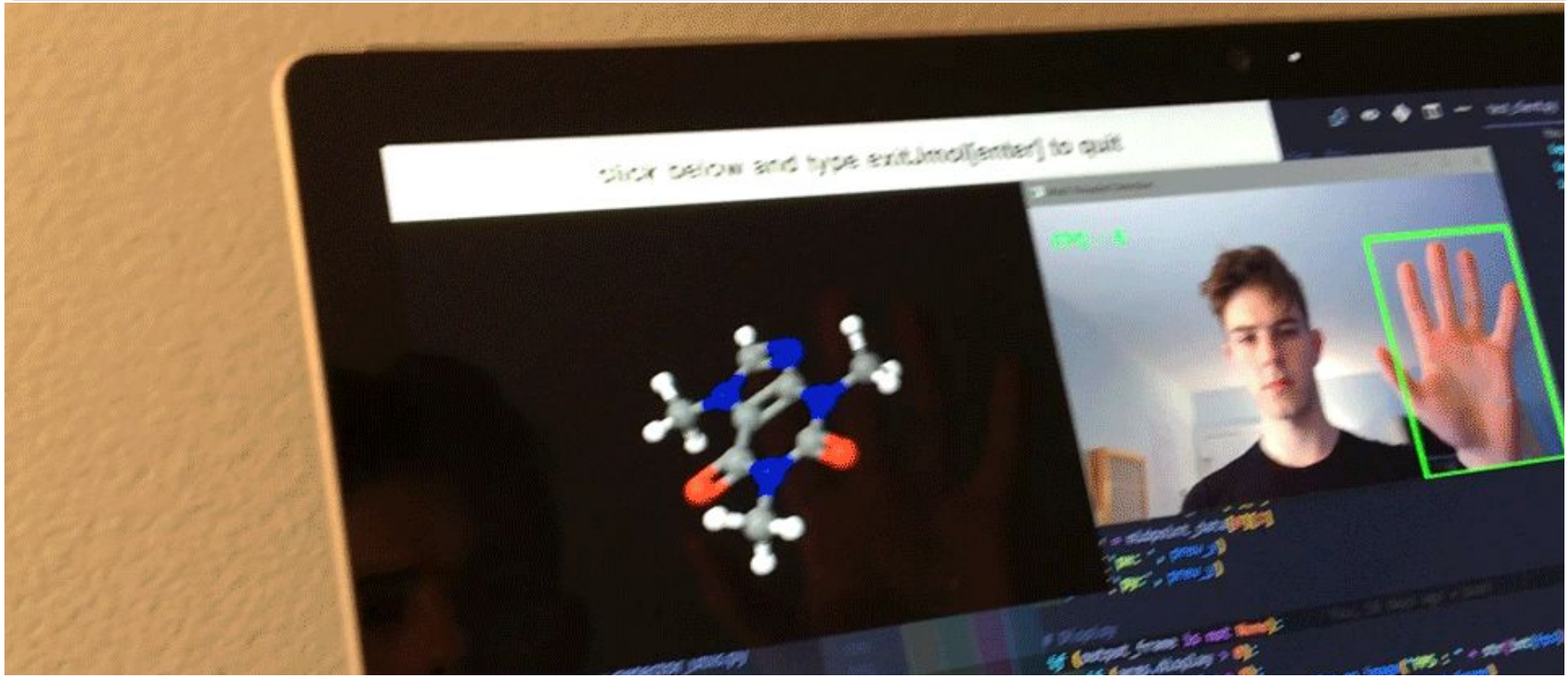
```
$ pip install tensorflowjs
$ tensorflowjs_converter \
    --input_format tf_saved_model \
    --output_node_names
'Predictions/Reshape_1' \
    path/to/tf_saved_model \
    path/to/tfjs_model
```

Exported model can be imported in javascript.

```
const model = await tf.loadGraphModel('path/to/model.json');
```

Exported Model

- **18.5mb** .. final model size
- Sharded into 5 files with a maximum size of 4.2mb
-
- Bundled into an NPM library, hosted on NPM (with benefits!)
  - Model artifact versioning
  - CDN serving (jsdelivr)
-


Click below and type exit.bmol[enter] to quit

translate3d(759px,
-46px, 0)
scale(1.2447062642214 61,
1.2447062642214 61)
rotate3d(0,0,1,-13.2288527806891

Demo!

- So .. does it work live?
- **https://victordibia.github.io/handtrack.js/#/**

Some Challenges

- **Manual Memory Management**
  - Tensorflow.js implements tensors using **WebGLTextures**.
  - Copied to GPU and cached to enable fast computation
  - Need to be explicitly deleted using the **tensor.dispose()**.
  - Use **tf.memory()** to see how many tensors your application is holding onto

Some Challenges

- General **limitations of the Browser**.
  - The browser is single threaded
  - Compute heavy operations **will** block the UI thread.

Some Challenges

- **Device Fragmentation**
  - Differences in devices can lead to wildly varying performance
    - CPU (Mobile, Desktop, Embedded)
    - GPU availability
    - Device power mode

Some Good Practices

- **Optimize Your Model**
  - Replace full Convolutional Operations with Depthwise separable convolutions (less parameters).
  - Remove optional post processing from model graph during model export.
  - Explore model compression methods (quantization, distillation, pruning).
  - Explore NAS/AutoML.

.

Some Good Practices

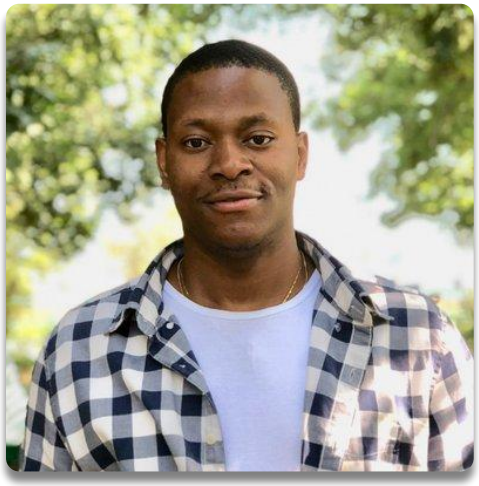
- **Design and Test Considerations**
  - Asynchronous methods (async await, promises) to avoid blocking the UI thread.
  - Visually communicate model load/inference latency.
  - Communicate model uncertainty where possible.
  - Test and benchmark across multiple devices!!
  - Chrome Profiler is your friend!
  - See Google PAIR (People + AI Guidebook) document on how to design human-centered AI products.

Conclusion

- ML in the browser is valuable when your objectives are focused on **privacy**, ease of **distribution**, **low latency** and **interactivity.**

- **Tensorflow.js** provides an API for ML in Javascript (online, offline, hybrid workflow). Fast, expressive, integrated with the rest of the TF ecosystem.

- Research advances in ML (compression, distillation, pruning) + Advances in acceleration in the browser (**WebAssembly, WebGPU**), positions ML in the browser for faster and wider integration.



Thank you!

Source Code: Handtrack.js
https://github.com/victordibia/handtrack.js

Research Engineer
Cloudera **Fast Forward Labs,** Brooklyn.
@vykthur | **@victordibia**