Deploying Models at Scale

Browser-based Models with TensorFlow.js

N. Rich Nguyen, PhD

SYS 6016

Why have your ML model on the browsers?

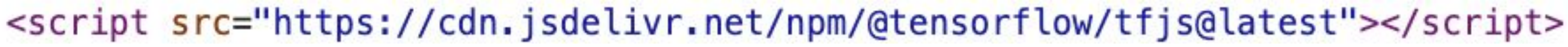
- Run model directly in the user’s browser, no need for an expensive server
- Make your website/apps more reliable
- Remove the need to query server to make prediction
- Reduce latency and make website more responsive
- Protect users’ privacy by not sending their data to a server

TensorFlow.js

- Supports Javascript Developers to make ML easy to integrate with websites
- Tensorflow.js is started in 2017 and launched in 2018
- Developers can run existing models, retrain them, or train from scratch
- Contain 11 pre-trained models for Vision, NLP, and tools to tune parameters

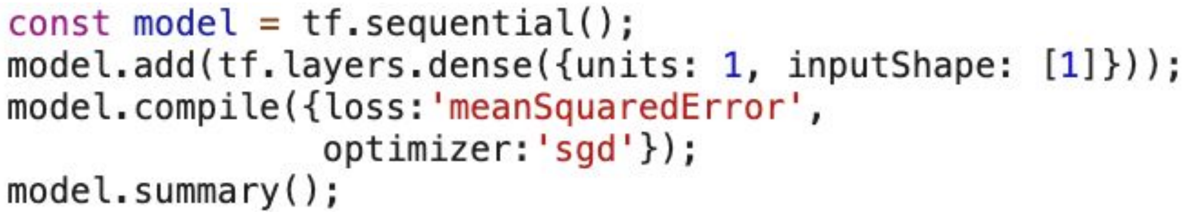
Coding your TensorFlow.js Model

Including the latest TensorFlow.js library:



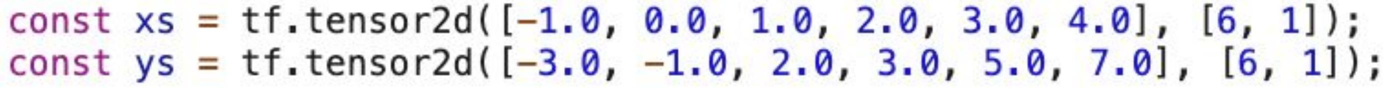
```
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@latest"></script>
```

Loading a sequential model:



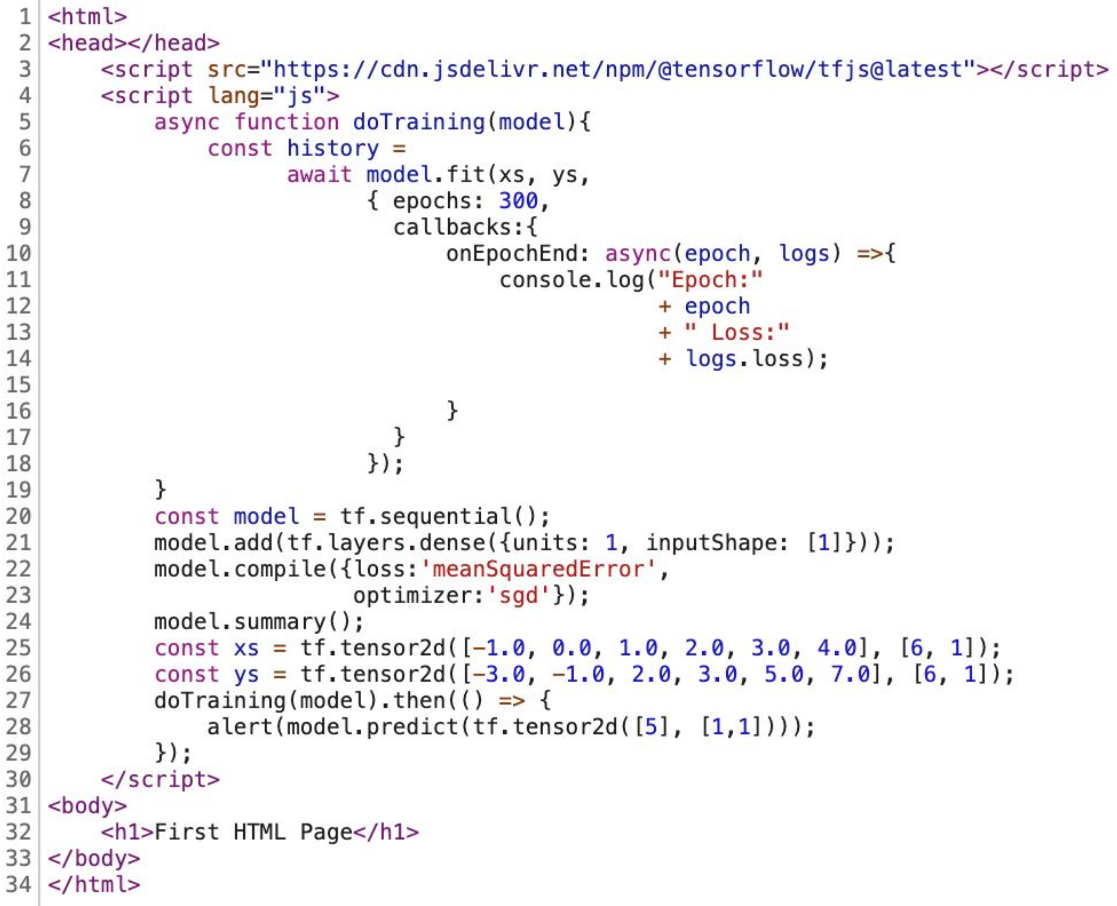
```
const model = tf.sequential();
model.add(tf.layers.dense({units: 1, inputShape: [1]}));
model.compile({loss:'meanSquaredError',
               optimizer:'sgd'});
model.summary();
```

Adding some training data:



```
const xs = tf.tensor2d([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], [6, 1]);
const ys = tf.tensor2d([-3.0, -1.0, 2.0, 3.0, 5.0, 7.0], [6, 1]);
```

Putting them together



```
<html>
<head></head>
    <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@latest"></script>
    <script lang="js">
        async function doTraining(model){
            const history =
                  await model.fit(xs, ys,
                        { epochs: 300,
                          callbacks:{
                              onEpochEnd: async(epoch, logs) =>{
                                  console.log("Epoch:"
                                              + epoch
                                              + " Loss:"
                                              + logs.loss);

                              }
                          }
                        });
        }
        const model = tf.sequential();
        model.add(tf.layers.dense({units: 1, inputShape: [1]}));
        model.compile({loss:'meanSquaredError',
                       optimizer:'sgd'});
        model.summary();
        const xs = tf.tensor2d([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], [6, 1]);
        const ys = tf.tensor2d([-3.0, -1.0, 2.0, 3.0, 5.0, 7.0], [6, 1]);
        doTraining(model).then(() => {
            alert(model.predict(tf.tensor2d([5], [1,1])));
        });
    </script>
<body>
    <h1>First HTML Page</h1>
</body>
</html>
```

Advances in TensorFlow.js

FaceMesh

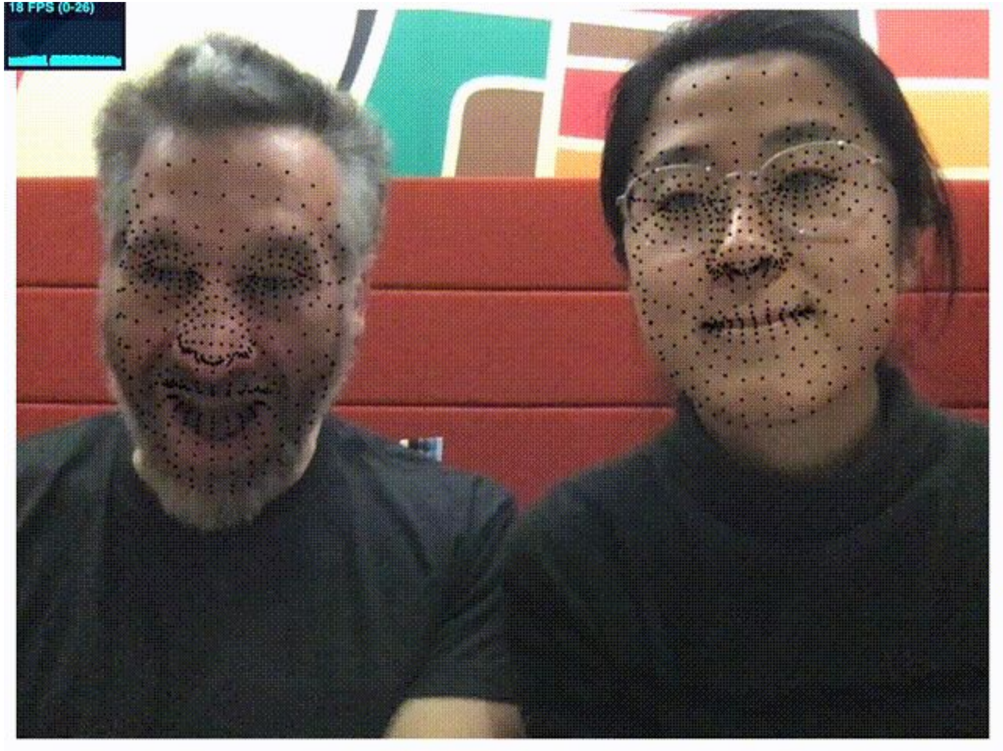
Predicts 486 3D facial landmarks to infer the approximate surface geometry of a human face

Model Size:

- Under 3MB

Performance:

- 15 fps on Pixel 3
- 35 fps on iPhone 11
- 40 fps on MacBook Pro



HandPose

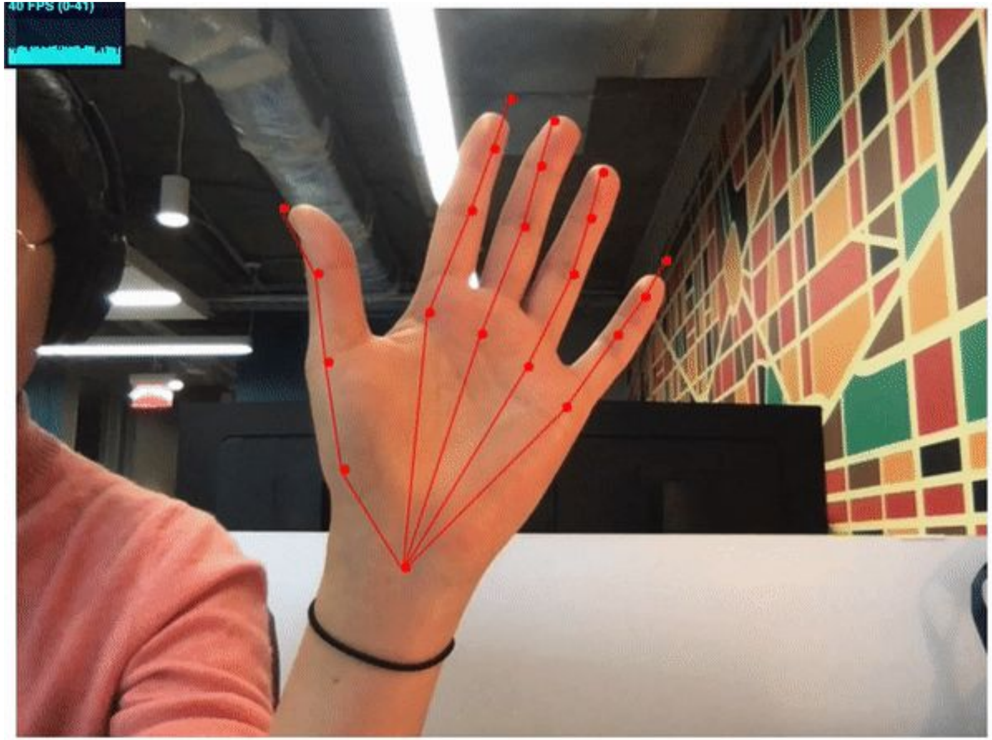
Predicts 21 3D hand keypoints per detected hand

Model Size:

- Under 12MB

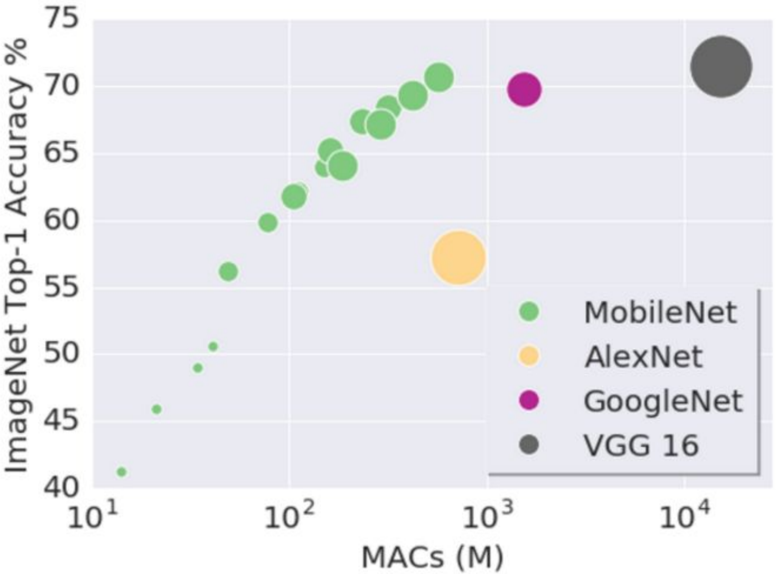
Performance:

- 6 fps on Pixel 3
- 30 fps on iPhone 11
- 40 fps on MacBook Pro



MobileNets

- Small, low-latency, low-power convolutional models parameterized to meet the resource constraints of a variety of use cases.
- Trade off between latency, size and accuracy while comparing favorably with popular models
- Choose the right MobileNet model to fit your latency and size budget.



and many more...

- Ready to be used inside any webpage
- Lightweights and efficient
- Re-trained capabilities

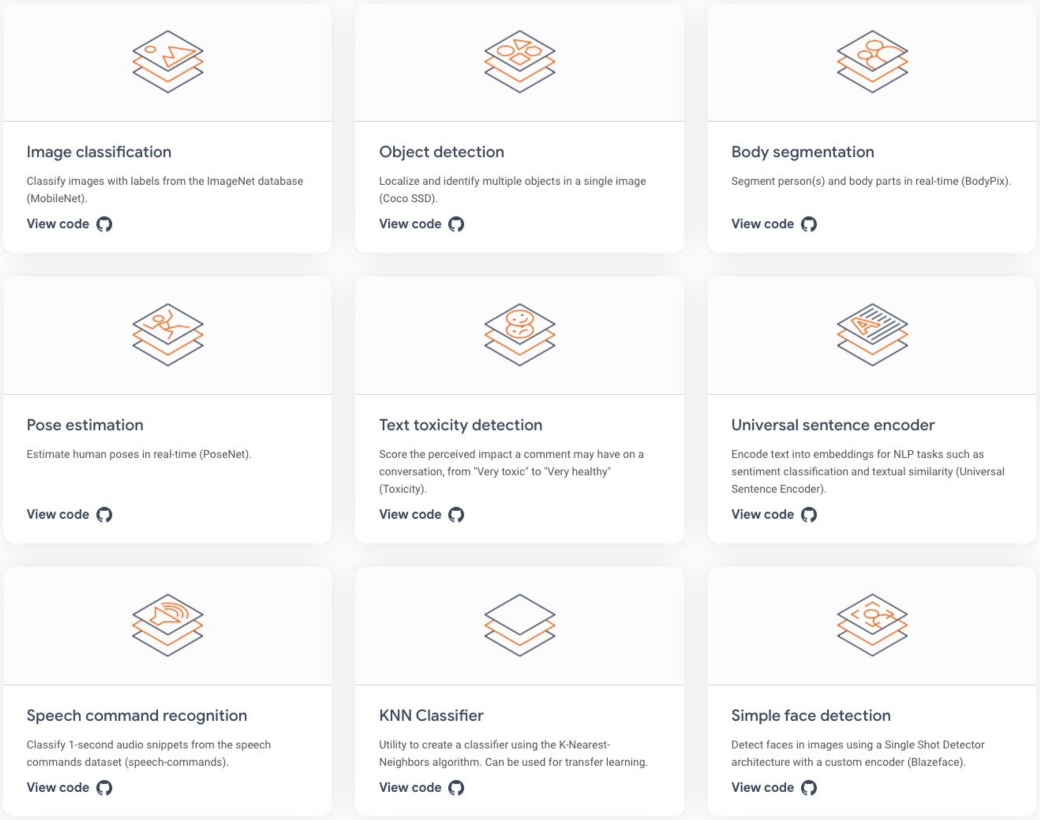


Image classification

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).

View code

Text toxicity detection

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

View code

Universal sentence encoder

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

View code

Speech command recognition

Classify 1-second audio snippets from the speech commands dataset (speech-commands).

View code

KNN Classifier

Utility to create a classifier using the K-Nearest-Neighbors algorithm. Can be used for transfer learning.

View code

Simple face detection

Detect faces in images using a Single Shot Detector architecture with a custom encoder (Blazeface).

View code

Demo: Rock-Paper-Scissor Game with MobileNet on TensorFlow.js

Summary

- Browser-based models provide many benefits for users and developers
- TensorFlow.js is easy to develop and deploy directly on the browsers
- TensorFlow.js is somewhat similar to TensorFlow Python, with a few changes to accomodate for Javascript syntax.
- TensorFlow.js has several advanced pre-trained ML models
- For more, check out: **tensorflow.org/js**

Safari **Firefox** **Chrome** **Edge** new **Opera**