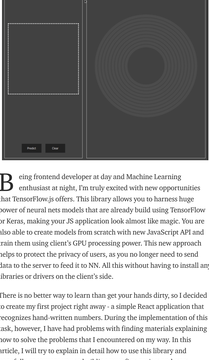
# My first TensorFlow.js project

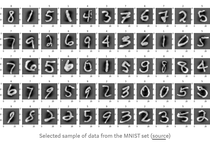
Piotr Skalski Follow
May 16, 2018 · 4 min read




Being frontend developer at day and Machine Learning enthusiast at night, I'm truly excited with new opportunities that TensorFlow.js offers. This library allows you to harness huge power of neural nets models that are already build using TensorFlow or Keras, making your JS application look almost like magic. You are also able to create models from scratch with new JavaScript API and train them using client's GPU processing power. This new approach helps to protect the privacy of users, as you no longer need to send data to the server to feed it to NN. All this without having to install any libraries or drivers on the client's side.

There is no better way to learn than get your hands dirty, so I decided to create my first project right away - a simple React application that recognizes hand-written numbers. During the implementation of this task, however, I have had problems with finding materials explaining how to solve the problems that I encountered on my way. In this article, I will try to explain in detail how to use this library and hopefully encourage you to building your first project and start adventure with ML inside browser. You can find all of the source code on Github as well as fully working demo here.

*Quick note: A PhD in Computer Science is not required to wrap your brain around this article.*



Selected sample of data from the MNIST set (source)

## Preperation is key to success

To start the project off on the right foot I decided to train my own model and use it later on as the heart of my application. The convolutional neural network was created in Keras, trained on MNIST dataset and then saved in form readable by TensorFlow.js. As the matter of fact there are several ways of achieving that goal - we can save the model in python script immediately after training or after the fact from the terminal using `tensorflowjs_converter`. In both cases, the `model.json` file will be created as output, alongside with several shard files. These files describe the structure of NN and the values of weights in nodes. **Make sure that the shard files are located in the same directory otherwise your model is not going to fly.** (those interested in architecture of used neural network, can refer to the full python notebook for more information)

```
import tensorflowjs as tfjs
# creating and training of model using Keras
# ...
tfjs.converters.save_keras_model(model, "./Models15")
```

```
$ tensorflowjs_converter --input_format keras ./Models15/model.h5 ./../Models15
```

## Hit the ground running

I decided to write my application in TypeScript, using React with Redux, but it should work just as well with vanilla JS. The only thing that you really need is `@tensorflow/tfjs` library, which you can add via npm and yarn package managers or HTML script tag.

```
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@0.8.0"> </script>
```

Show time begins! Due to the subject of the article I will skip the details of creating HTML Canvas that allows user to draw inside browser and jump straight into the implementation of the model within the application. First things first - let's import the library and load previously prepared model. It is worth mentioning that files that make up the model usually weigh a"little bit" more than a few bytes, so I used `await` operator to prevent main UI thread of the browser from being locked during loading process. It may also be a good idea to use a **service worker** to minimize the number of downloads.

```
import * as tf from "@tensorflow/tfjs";
// definition of the component supporting the model
// ...
protected async loadModel() {
    this.model = await tf.loadModel(AppSettings.mnistModelUrl);
}
```



Finally, it's time to put model to the test. I used ImageData retrieved from canvas as input for my model, but one of the coolest thing about TF.js is that we can take almost any picture or video, turn it into tensor and feed it to your model. The actual calculation takes place inside the `predict` method, but as you can see below, the matter is a bit more complicated.

```
protected async predict(imageData: ImageData) {

    const pred = await tf.tidy(() => {

        let img:any = tf.fromPixels(imageData, 1);
        img = img.reshape([1, 28, 28, 1]);
        img = tf.cast(img, 'float32');

        const output = this.model.predict(img) as any;

        this.predictions = Array.from(output.dataSync());
    });
```

mennoo, wnacn ensures tnat ar tne end or tne cnacuation au intermediate tensors that were allocated in memory will be removed. Another thing that we can not forget is to provide the right tensor dimensions. This is related to the decisions we made at the stage of choosing the neural network architecure inside Keras.

```
model.add(Conv2D(
    filters = 32,
    kernel_size = [5,5],
    padding = 'Same',
    activation = 'relu',
    input_shape = [28,28,1]
))
```

## And Voila!

If everything went according to plan, model returned a ten-element JS array with probability values for each digit. Now it's only up to you how you visualize the results. The model I created was 99.5% accurate on Kaggle however, I have the impression that its effectiveness is actually a little lower. I am very happy with the final result and I already have a head full of ideas for another project using this fantastic library. This time I'll raise the bar higher.



Follow me if you found this post interesting. Check out other things that I do on my GitHub and Kaggle.

Machine Learning · Artificial Intelligence · Data Science · JavaScript · Web Development

352 claps

WRITTEN BY
Piotr Skalski

Machine Learning Enthusiast / Big Data Engineer @ VirtusLab / Computer Science Student @ AGH UST Cracow

Follow

See responses (3)

## More From Medium

More from Piotr Skalski

How to train Neural Network faster with optimizers?
Piotr Skalski i… · Nov 9, 2018 · 1… · 2K

Related reads

Realtime Face Recognition
Goran Jovanov · Jan 10 · 6 min re…

Related reads

Getting Alexa to Respond to Sign Language Using Your Webcam and TensorFlow.js
TensorFlow … · Aug 8, 2018 … · 3.6K