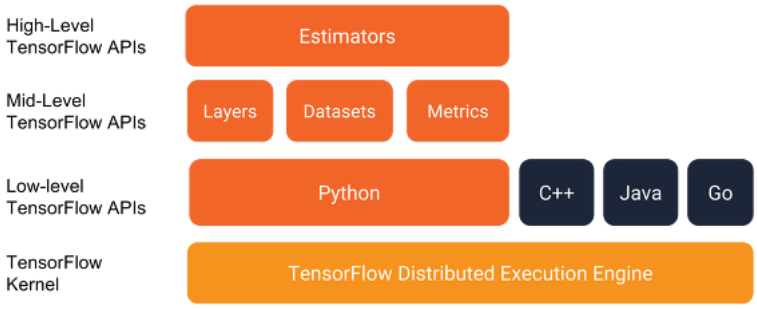
Tensorflow.js

Tensorflow란?

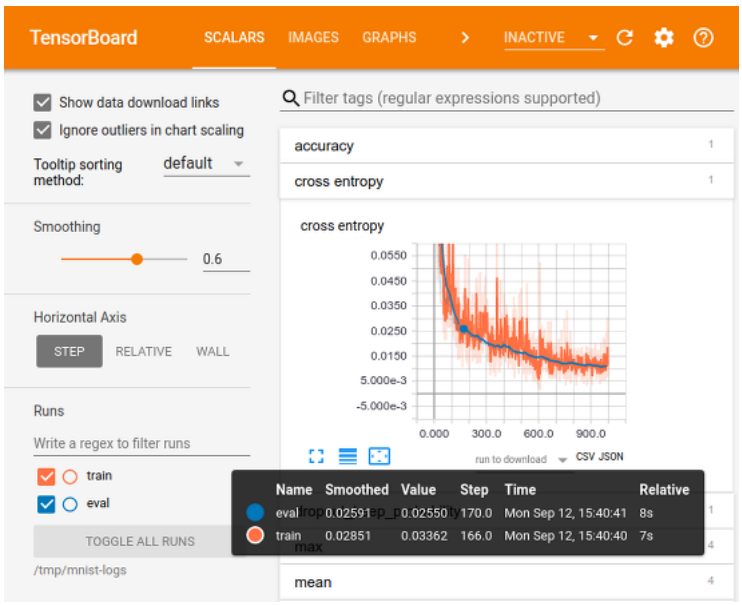
Tensorflow는 Google에서 만든, 딥러닝 프로그램을 쉽게 구현할 수 있도록 다양한 기능을 제공해주는 라이브러리다. Tensorflow자체는 기본적을 C++로 구현이 되어있다.

하지만 아래 그림처럼 Python, Java Go 언어 등 다양한 언어를 지원한다.

파이썬을 최우선으로 지원하기 때문에 대부분의 편한 기능들이 파이썬 라이브러리로만 구현되어 있어 Python에서 개발하는 것이 편하다.



또한, 브라우저에서 실행 가능한 시각화 도우인 텐서보드(TensorBoard)를 제공하여, 딥러닝 학습 과정을 추적하는데 유용하게 사용된다.



Tensorflow 추상화 및 사전 학습된 모델

Tensorflow에서는 Keras나 TF-Slim 과 같은 추상화 라이브러리를 제공하여 저수준 Tensorflow 라이브러리에 대해 손쉽게 고수준 접근이 가능하게 해준다. 이를 이용하여 간단하게 딥러닝 모델을 구현할 수 있다.또한, 아래의 그림처럼 Tensorflow는 사전에 학습된 모델들을 제공해준다. 이러한 모델을 사용자들이 단 몇 줄의 코드로 구현할 수 있으며, 새롭게 학습을 할 필요 없이 바로 실무에 적용할 수 있고, 새로운 데이터에 맞게 모델을 조정할 수도 있다.

Tensorflow외 라이브러리

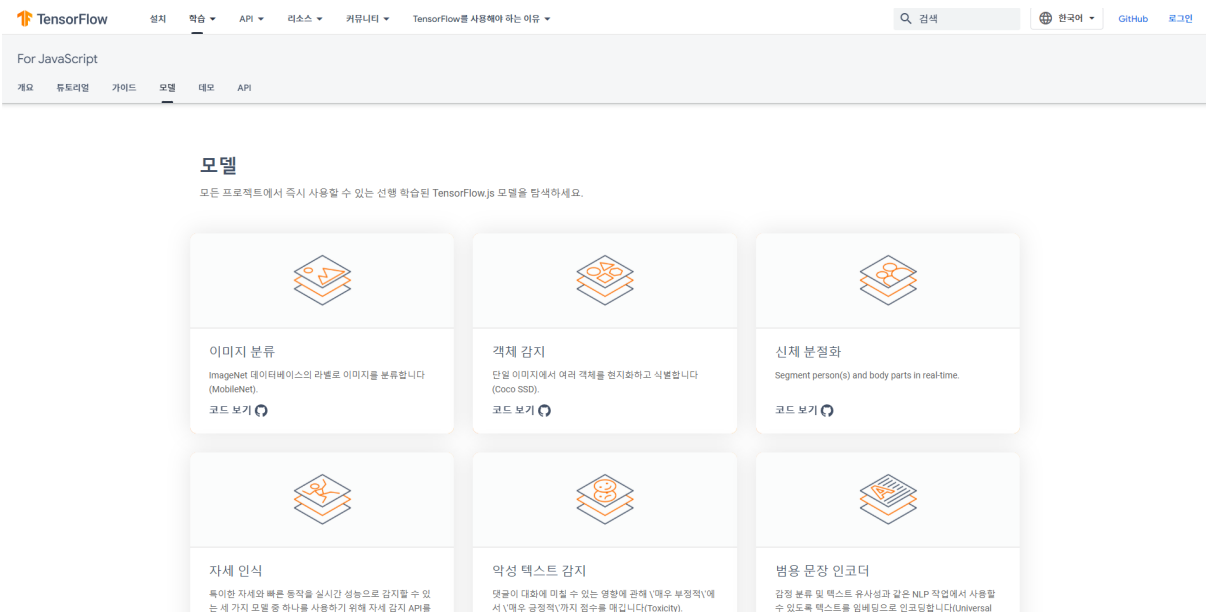
- TensorFlow
- ConvNetJS
- Brain.js
- Caffe2
- PyTorch
- theano

시작하기

Tensorflow는 다음과 같은 방법으로 사용할 수 있다.

1. Tensorflow가 제공하는 기존 모델을 가지고 실행할 수 있는 방법
2. Tensorflow가 제공하는 모델에 추가적인 학습을 시키는 방법
3. Javascript를 작성하여 Tensor모델로 변경하여 모델을 학습 시키는 방법

1. 기존 모델 실행 방법



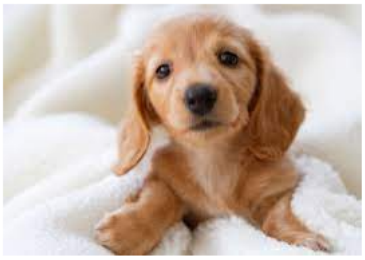
Tensorflow.js 모델

Tensorflow는 여러가지 학습되어있는 모델을 제공해준다.

1-1. 이미지 분류

```
<!-- Load TensorFlow.js. This is required to use MobileNet. -->
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@1.0.1"> </script>
<!-- Load the MobileNet model. -->
<script src="https://cdn.jsdelivr.net/npm/@tensorflow-models/mobilenet@1.0.0"> </script>

<!-- Replace this with your image. Make sure CORS settings allow reading the image! -->
<img id="img" src="cat.jpg"></img>

<!-- Place your code in the script tag below. You can also use an external .js file -->
<script>
  // Notice there is no 'import' statement. 'mobilenet' and 'tf' is
  // available on the index-page because of the script tag above.

  const img = document.getElementById('img');

  // Load the model.
  mobilenet.load().then(model => {
    // Classify the image.
    model.classify(img).then(predictions => {
      console.log('Predictions: ');
      console.log(predictions);
    });
  });
</script>
```

Git에서 제공하는 **via Script Tag**이다.



다음처럼 강아지 이미지를 넣게 되면

```
Predictions:
▼(3) [{…}, {…}, {…}]
  ▶0: {className: 'golden retriever', probability: 0.44991329312324524}
  ▶1: {className: 'cocker spaniel, English cocker spaniel, cocker', probability: 0.27668148279190063}
  ▶2: {className: 'Sussex spaniel', probability: 0.11602012068033218}
    length: 3
  ▶[[Prototype]]: Array(0)
```

해당 강아지가 어떤 종인지 확률로 나오게 된다.

1-3. 악성 텍스트 감지

```
npm install @tensorflow/tfjs @tensorflow-models/toxicity
```

```
require('@tensorflow/tfjs');
const toxicity = require('@tensorflow-models/toxicity');

// The minimum prediction confidence.
const threshold = 0.9;

// Load the model. Users optionally pass in a threshold and an array of
// labels to include.
toxicity.load(threshold).then(model => {
  const sentences = ['you suck'];

  model.classify(sentences).then(predictions => {
    // `predictions` is an array of objects, one for each prediction head,
    // that contains the raw probabilities for each input along with the
    // final prediction in `match` (either `true` or `false`).
    // If neither prediction exceeds the threshold, `match` is `null`.

    console.log(predictions);
    /*
    prints:
    {
      "label": "identity_attack",
      "results": [{
        "probabilities": [0.9659664034843445, 0.03403361141681671],
        "match": false
      }]
    },
    {
      "label": "insult",
      "results": [{
        "probabilities": [0.08124706149101257, 0.9187529683113098],
        "match": true
      }]
```

```
    },
    ...
     */
  });
});
```

sentences 에는 어떤 특정 문장이 들어가고, predictions에서는 해당 문장이 위험도에 따라 얼마나 부정적인지 긍정적인지 나뉘어 지게 된다.

Tensorflow.js로 모델 만들기

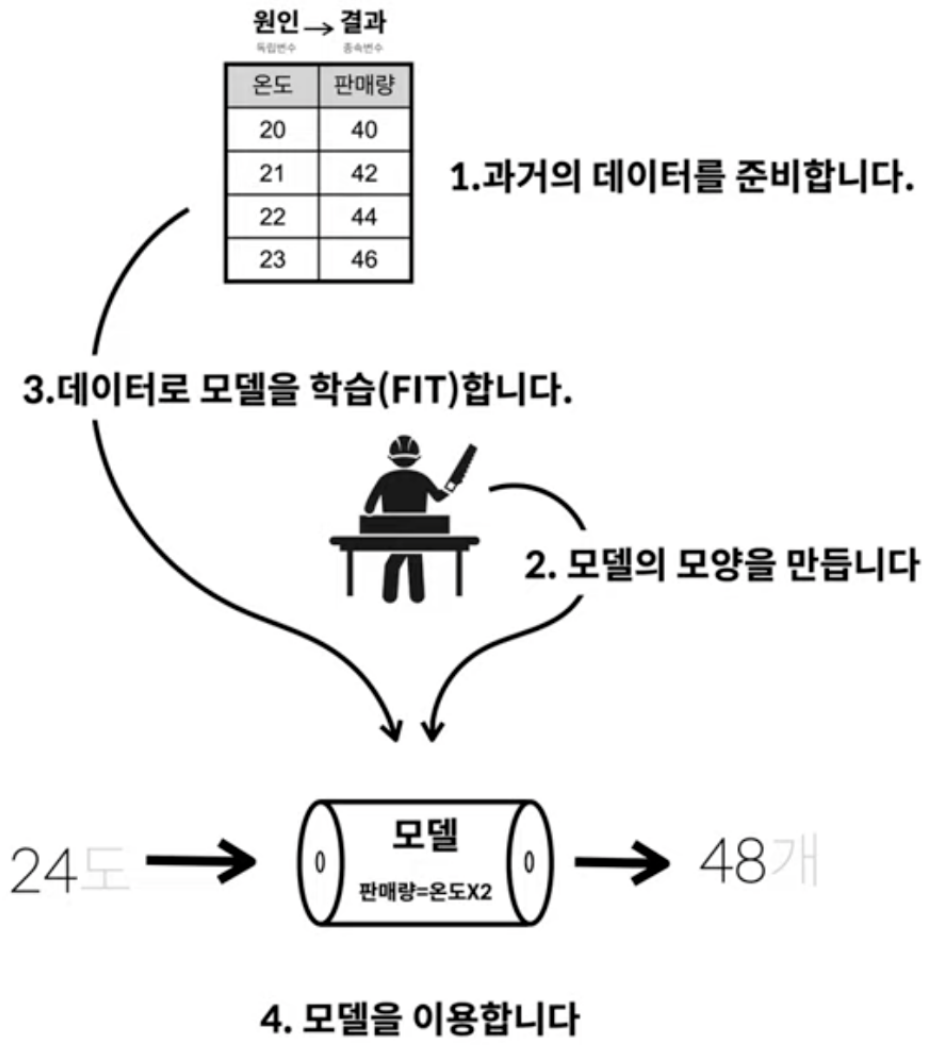
지도 학습

Machine Learning 에는 지도 학습, 비지도 학습, 강화 학습과 같은 기술들이 있다. 그 중에서도 지도 학습은 데이터가 값의 형태일 때 사용하는데 회귀라는 방법으로 학습하게 된다. 회귀 과정은 아래와 같다.

1. 과거의 데이터를 준비한다.
2. **모델**의 모양을 만든다.
3. 데이터로 **모델**을 학습(FIT)한다.
4. **모델**을 이용한다.

레모네이드를 판매하는 장사를 한다고 가정해보자. 레모네이드 판매량이 그 날 온도에 영향을 미친다면 내일 몇 개의 레모네이드가 판매될지 예측하는 모델을 어떻게 만들 수 있을까?

기본적인 모델을 만드는 과정은 아래 그림과 같다.



Node.js 사용법

```
//pure javascript
npm install @tensorflow/tfjs

//c++ binding
npm install @tensorflow/tfjs-node

//using gpu
npm install @tensorflow/tfjs-node-gpu
```

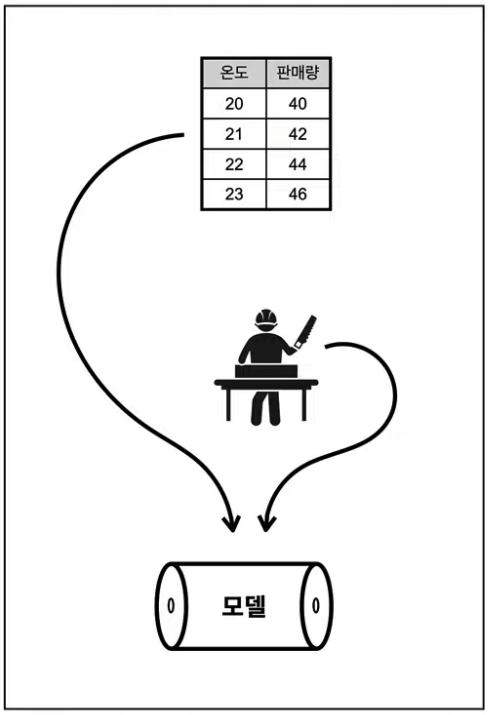
HTML에서 실행법

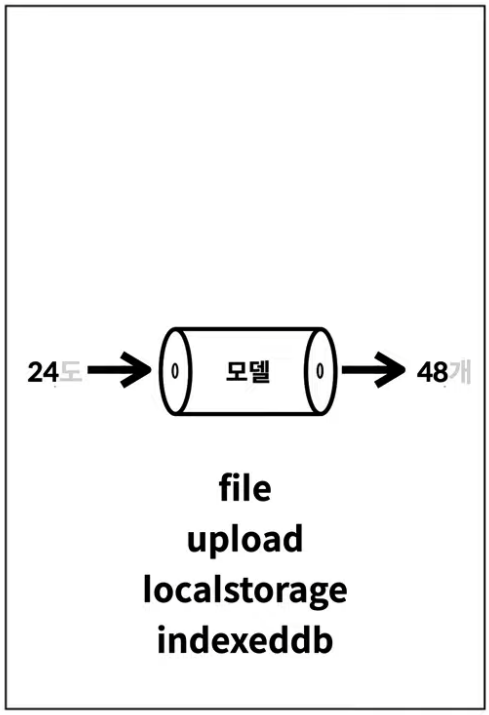
```
<!DOCTYPE html>
<html>
<head>
    <title>TensorFlow.js Tutorial - lemon</title>

    <!-- Import TensorFlow.js -->
    <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@1.0.0/dist/tf.min.js"></script>

</head>

<body>
    <script>
        ...
    </script>
</body>

</html>
```

이제 레모네이드 판매량 데이터를 학습하여 모델을 만드는 코드를 작성해보자.

```
// 1. 과거의 데이터를 준비합니다.
var 온도 = [20, 21, 22, 23];
var 판매량 = [40, 42, 44, 46];
var 원인 = tf.tensor(온도);
var 결과 = tf.tensor(판매량);

// 2. 모델의 모양을 만듭니다.
var X = tf.input({ shape: [1] }); //컬럼이 하나인 값 -> 온도는 하나니깐. (입력)
var Y = tf.layers.dense({ units: 1 }).apply(X); // units:1 -> 하나의 값을 출력 (출력)
var model = tf.model({ inputs: X, outputs: Y }); // 모델이라는 변수에 학습과 예측작업을 한다.
var compileParam = { optimizer: tf.train.adam(), loss: tf.losses.meanSquaredError } //optimizer(효율)과 loss(모델이 잘만들어졌는지 측정)라는 값
model.compile(compileParam); //모델을 만드는 실질적인 부분.

// 3. 데이터로 모델을 학습시킵니다.
var fitParam = { epochs: 100 }; //학습 횟수.
model.fit(원인, 결과, fitParam).then(function (result) {
// 4. 모델을 이용합니다.
// 4.1 기존의 데이터를 이용
var 예측한결과 = model.predict(원인);
     예측한결과.print(); //예측한 결과는 tensor에서 제공하는거라 .print()작성하여 출력
 });

// 4.2 새로운 데이터를 이용
var 다음주온도 = [15, 16, 17, 18, 19]
var 다음주원인 = tf.tensor(다음주온도);
var 다음주결과 = model.predict(다음주원인);
다음주결과.print();
```

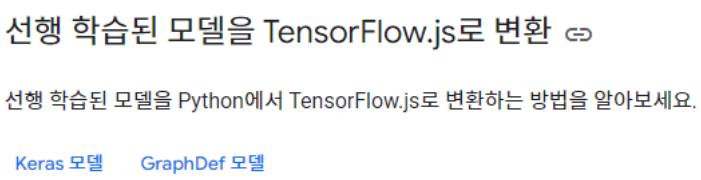
Tensorflow.js Model 저장 및 불러오기





https://www.tensorflow.org/js/guide/save_load?hl=ko

Python으로 훈련시킨 모델 Tensorflow.js로 가져오기



Tensorflow는 Python으로 학습시킨 Keras, GraphDef 모델 또한 Tensorflow.js로 변환이 가능하다.

Python으로 학습시켜 모델을 export 하고 그 모델을 Tensorflow.js로 import 하면 된다.

Keras모델

GraphDef모델

자료

Tensorflow.js 홈페이지

출처

https://excelsior-cjh.tistory.com/148

https://velog.io/@jieun7583/TensorFlow.js-으로-ML-모델-만들기