

Valgfag: Introduktion til anvendt Machine Learning i webapplikationer

AI with:



TensorFlow.js

Machine learning in the browser
Day 2



Agenda (Part 1 + 2)

- 01 Tensorflow core concepts →
- 02 Garbage collection →
- 03 Lab 3.1 →
- 04 Data preparations →
- 05 Lab 4.1 →
- 06 Starting HomeWork (in groups)



Core Concepts

TensorflowJS APIs

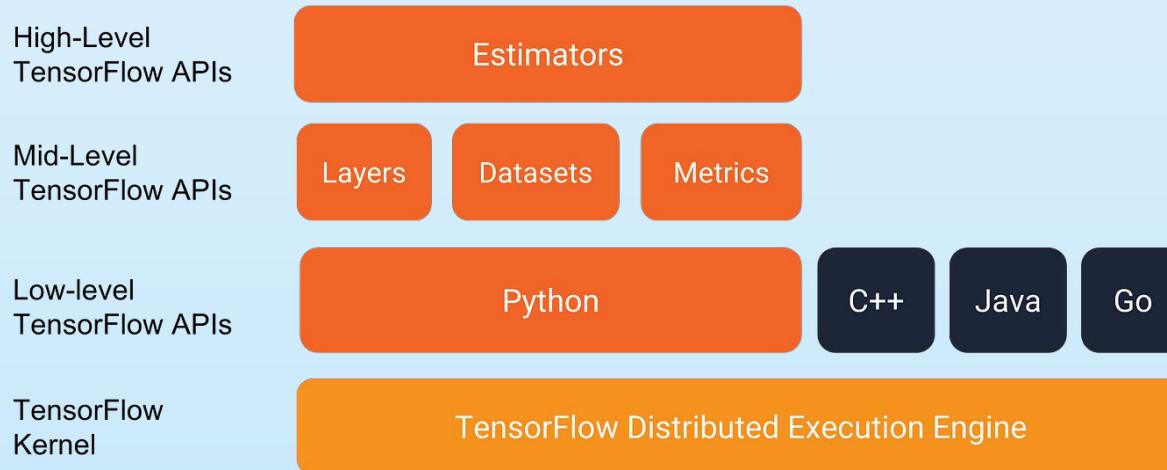
- Low level: Operations OPS
- High level: Layers



TensorFlow.js

Low level part of Tensorflow API's

- The TensorflowJS api are similar the Python API
- We will use mathematical operations to manipulate data.
- Build learning models (the hard way)



High Level part og TensorFlow.js APIs

- Defining complex models easily
- Can be compared to Keras API in Python
- Used for Artificial Neural Networks

High-Level
TensorFlow APIs

Estimators

Mid-Level
TensorFlow APIs

Layers

Datasets

Metrics

Low-level
TensorFlow APIs

Python

C++

Java

Go

TensorFlow
Kernel

TensorFlow Distributed Execution Engine

Demo

Go to: <https://js.tensorflow.org/api/latest/>

User browser search (ctrl/cmd +f) and search for:
operations

Find the functions you can call under operations.

User browser search (ctrl/cmd +f) and search for:

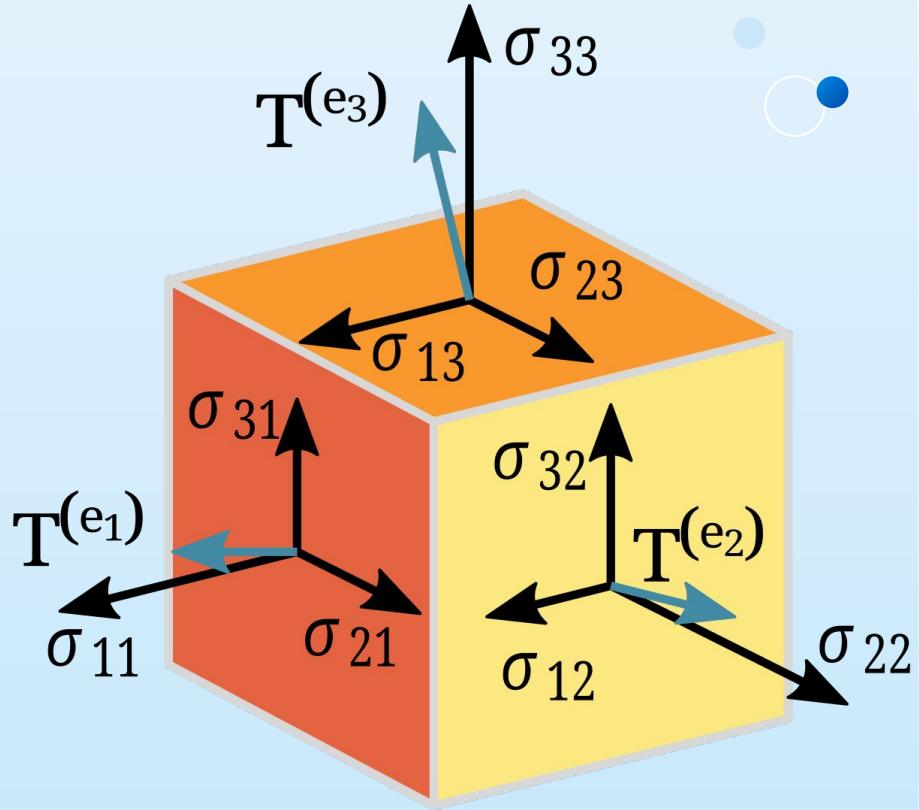
Layers

Find the functions you can call under Layers in the left side

The screenshot shows a web browser displaying the TensorFlow.js API documentation at <https://js.tensorflow.org/api/latest/>. The navigation bar includes links for Overview, API, and operations. The main content area is titled "Operations" and lists several function names under categories like Rescaling, CenterCrop, Resizing, CategoryEncoding, RandomWidth, and Tensors. To the right of the list, there is a sidebar with a heading "Tensors" and a brief description: "Tensors are the core data generalization of vectors and dimensions." Below the sidebar, another section titled "Tensors / Creation" is partially visible with a similar descriptive text.

What is a Tensor

A tensor is a mathematical, multidimensional array of numbers that generalizes scalars, vectors, and matrices to higher dimensions, acting as a container for data in \mathbb{N} -dimensions.



What we need to know as developer

We do NOT need to understand all the math!!!

We need to understand:

What a tensor is in practical terms and how we can create a Tensor

How can we create a tensor:

<https://js.tensorflow.org/api/latest/#Tensors>

Tensors

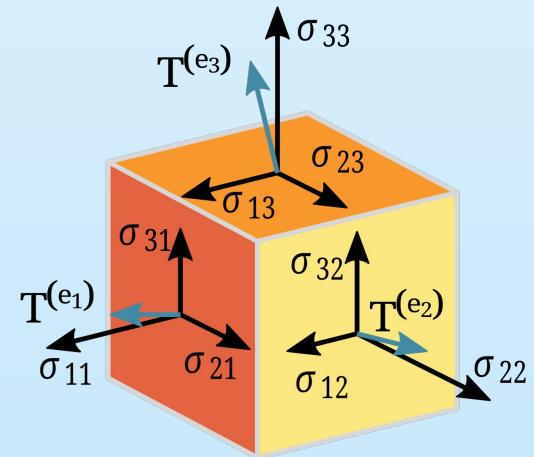
Tensors are the core datastructure of TensorFlow.js. They are the generalization of vectors and matrices to potentially higher dimensions.

Tensors / Creation

We have utility functions for common cases like Scalar, Vector, Matrix, and 4D tensors, as well a number of functions to initialize tensors useful for machine learning.

Defining a Tensor

- A list of values
- A data type int, float, string
- A number of dimensions
- A shape defines how the values are distributed through the system



Scalar Tensor

- It holds a single value
- 0 dimensions aka rank 0

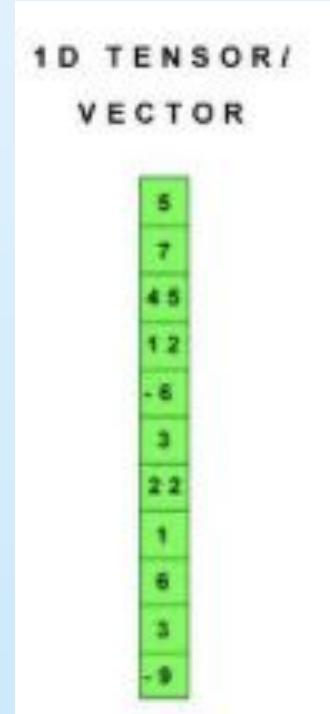
```
tf.scalar(32) = tf.tensor([32],[])
```

It is simpler and safer to use scalar so there is no misunderstanding about the rank we want here.

Scalar	Vector	Matrix	Tensor
1	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$	$\begin{bmatrix} \begin{bmatrix} 1 & 2 \end{bmatrix} & \begin{bmatrix} 3 & 2 \end{bmatrix} \\ \begin{bmatrix} 1 & 7 \end{bmatrix} & \begin{bmatrix} 5 & 4 \end{bmatrix} \end{bmatrix}$

1D Tensor

- A list of values and positions
- `tf.tensorId([4,7,45,...])` (recommended)
- `tf.tensor([1,2,3],[3])` is 1d (Do not use this version)



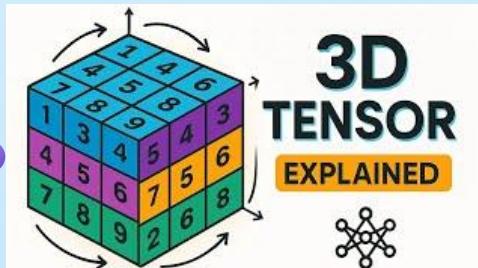
<https://mylearningsinai.ml.wordpress.com/tensorflow/tensors/>

2D Tensor

- Array of arrays
- `tf.tensor2d([-9,4,2,5,7],[3,0,12,8,61],....)`

3d is a cube:

We will be working with 1-3 dimensions.



2 D T E N S O R /
M A T R I X

- 9	4	2	5	7
3	0	1 2	8	6 1
1	2 3	- 6	4 5	2
2 2	3	- 1	7 2	6

Tensor math operations

- Tf.add (a,b) simple
- Try to find some of the more adv. Operations

Why use math operations

- Prepare data for processing (normalisation)
- Build low level models with operations (to low level)
- Extend [Tensorflow.js](#) write optimisation

The diagram shows the addition of two 2x3 matrices. The first matrix is $\begin{bmatrix} 3 & 8 \\ 4 & 6 \end{bmatrix}$. The second matrix is $\begin{bmatrix} 4 & 0 \\ 1 & -9 \end{bmatrix}$. The result is $\begin{bmatrix} 7 & 8 \\ 5 & -3 \end{bmatrix}$. Arrows point from the numbers 3 and 4 to the sum 7, with the annotation $3+4=7$. Arrows also point from the numbers 8 and 0 to the result 8.

$$\begin{bmatrix} 3 & 8 \\ 4 & 6 \end{bmatrix} + \begin{bmatrix} 4 & 0 \\ 1 & -9 \end{bmatrix} = \begin{bmatrix} 7 & 8 \\ 5 & -3 \end{bmatrix}$$
$$3+4=7$$

Memory management

In TensorFlow.

Tensors are stored in WebGL
Operations happen in WebGL

WebGL has no garbage collecting

Note that underlying GL object will be automatically marked for deletion when the JS object is destroyed, but can we wait for it?

PERFORMANCE

Memory

[tf.tidy](#)
[tf.dispose](#)
[tf.keep](#)
[tf.memory](#)

Timing

[tf.time](#)
[tf.nextFrame](#)

Profile

[tf.profile](#)

ENVIRONMENT

[tf.Environment](#)

Performance

Performance / Memory

[tf.tidy \(nameOrFn, fn?\)](#)

function [source](#)

Executes the provided function `fn` and after it is executed, cleans up all intermediate tensors allocated by `fn` except those returned by `fn`. `fn` must not return a Promise (async functions not allowed). The returned result can be a complex object.

Using this method helps avoid memory leaks. In general, wrap calls to operations in `tf.tidy()` for automatic memory cleanup.

NOTE: Variables do *not* get cleaned up when inside a `tidy()`. If you want to dispose variables, please use `tf.disposeVariables()` or call `dispose()` directly on variables.

```
// y = 2 ^ 2 + 1
const v = tf.tidy(() => {
```

Memory management

Tf.memory : returns info about memory usage

Tf.dispose : clean memory from container

If we use tf.tidy we can better control when we want to clean all that are ready for cleaning

Tf.keep is used for keeping a bit for later use.

DEMO



What have we learned from part 1

- Getting to know [TensorFlow.js](#) APIs
- A tensor has values, dimensions, shape, data type
- [TensorFlow.js](#) provides math operations for tensors
- Optimised depending on backend, eg. WebGL
- Memory usage must be managed



Lab 3.1

3.1 Tensor Math and Memory Management



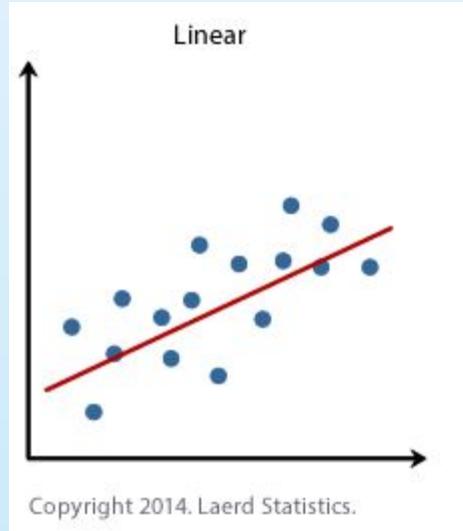
Part 2: Data preparations

- Import CSV
- Visualise data
- Features and Labels
- Normalising data
- Split tensor into sub tensors



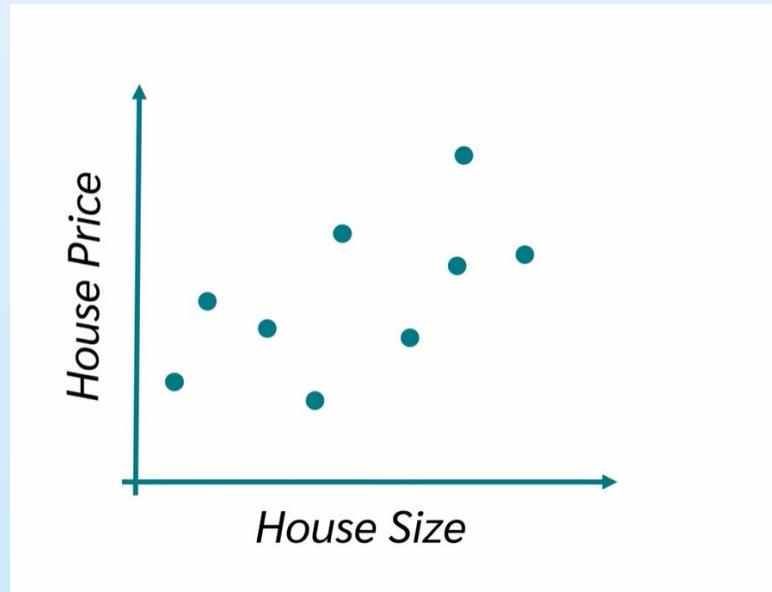
Linear regression

- Simple machine learning problem
- Plot a line to represent a correlation
- Make predictions for new data points



Linear regression

- Prepare the data
- Create a model
- Train the model
- Test the model
- Make predictions



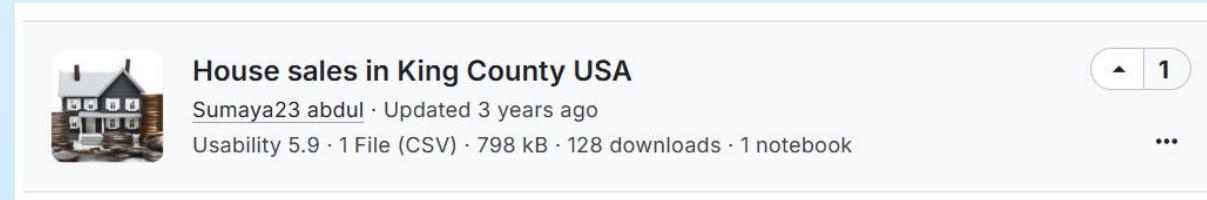
<https://www.youtube.com/watch?v=T5AoqxQFkzY>

Preparing the data

1. What is the problem we want to solve?

Go to <https://www.kaggle.com/datasets>

Find this



What data can we use to predict the sales price of a house? Keep it simple
If you can't find the data then look in itsLearning.

Explore the data

Open the csv and find the data we talked about before
Are there any problems with the data?

How much data are there? Is it enough?

Lets import using `tf.data.csv` – find the documentation to start with.



The screenshot shows a portion of the TensorFlow Python API documentation. On the left, under the 'DATA' section, there's a 'Creation' heading with several links: `tf.data.array`, **`tf.data.csv`** (which is highlighted with a red underline), `tf.data.generator`, and `tf.data.microphone`. To the right, the detailed documentation for `tf.data.csv` is shown. It starts with the function signature **`tf.data.csv (source, csvConfig?)`**, followed by a 'function' label and a 'source' link. A descriptive text follows: 'Create a `csvDataset` by reading and decoding CSV file(s) from provided URL or local path if it's in Node environment.'

Hosting the csv



Setting up the web server is dependant on your OS.

Find out :) get advice from fellow student, use ai chat, or google a video on it, find more info on itsLearning...

When everyone in the group are ready let me know.

```
<script type="text/javascript">  
    Const houseSalesDataset = tf.data.csv("--URL to file--")  
</script>
```



Inspect

In the documentation, inspect the following:

Tf.data.csv

 csvConfig

 columnNames

tf.data.CSVDataset (is returned)



Verify that you can read data

```
<script type="text/javascript">  
  Const houseSalesDataset = tf.data.csv("--URL to file--")  
  log.console(houseSalesDataset.take(10).toArray())  
  
</script>
```

Why is this not working? What is promise??

[Operations](#)

[tf.data.zip](#)

[Classes](#)

[tf.data.CSVDataset](#)

- .columnNames
- .tf.data.Dataset
- .batch
- .concatenate
- .filter
- .forEachAsync
- .map
- .map4Async
- .prefetch
- .repeat
- .skip
- .shuffle
- .take
- .toArray

[VISUALIZATION](#)

[TIL](#)

[tf.util.assert](#)

[tf.createShuffledIndices](#)

[tf.decodeString](#)

[tf.encodeString](#)

[Data / Classes](#)

[**tf.data.CSVDataset**](#) extends [tf.data.Dataset](#) class [source](#)

Represents a potentially large collection of delimited text records.

The produced `TensorContainer`s each contain one key-value pair for every column of the table. When a field is empty in the incoming data, the resulting value is `undefined`, or throw error if it is required. Values that can be parsed as numbers are emitted as type `number`, other values are parsed as `string`.

The results are not batched.

[**columnNames**](#) () method [source](#)

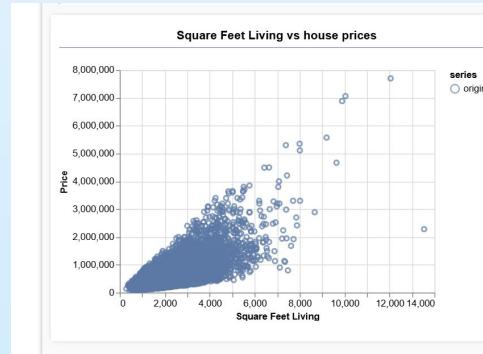
Returns column names of the csv dataset. If `configuredColumnOnly` is true, return column names in `columnConfigs`. If `configuredColumnOnly` is false and `columnNames` is provided, `columnNames`. If

Visualising data

Inspect tfjs-vis.

<https://github.com/tensorflow/tfjs/tree/master/tfjs-vis>

- `tfvis.render.scatterplot`



Visualization

tfjs-vis is a companion library for TensorFlow.js that provides in-browser visualization capabilities for training and understanding models. [API docs for tfjs-vis are available here](#)

Preparing features and labels

- Create a single array of x-values.
(features) and place it in a 2d-tensor
- Create a single array of y-values (Label)
and place it in a 2d-tensor
- Print to see result.

Label
(output)

```
const featurevalues = await points.map(p => p.x).toArray();
const featureTensor = tf.tensor2d(featurevalues, [featurevalues.length, 1]);

const labelvalues = await points.map(p => p.y).toArray();
const labelTensor = tf.tensor2d(labelvalues, [labelvalues.length, 1]);

featureTensor.print();
labelTensor.print();
```



Feature
(input)

Lab 4.1

4.1 Prepare the Data



Thank you

See you next time.

You find next time in timeedit.net search for course name or you initials to find your personal schema.

