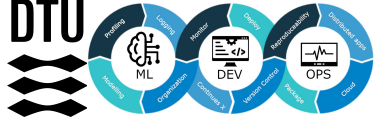
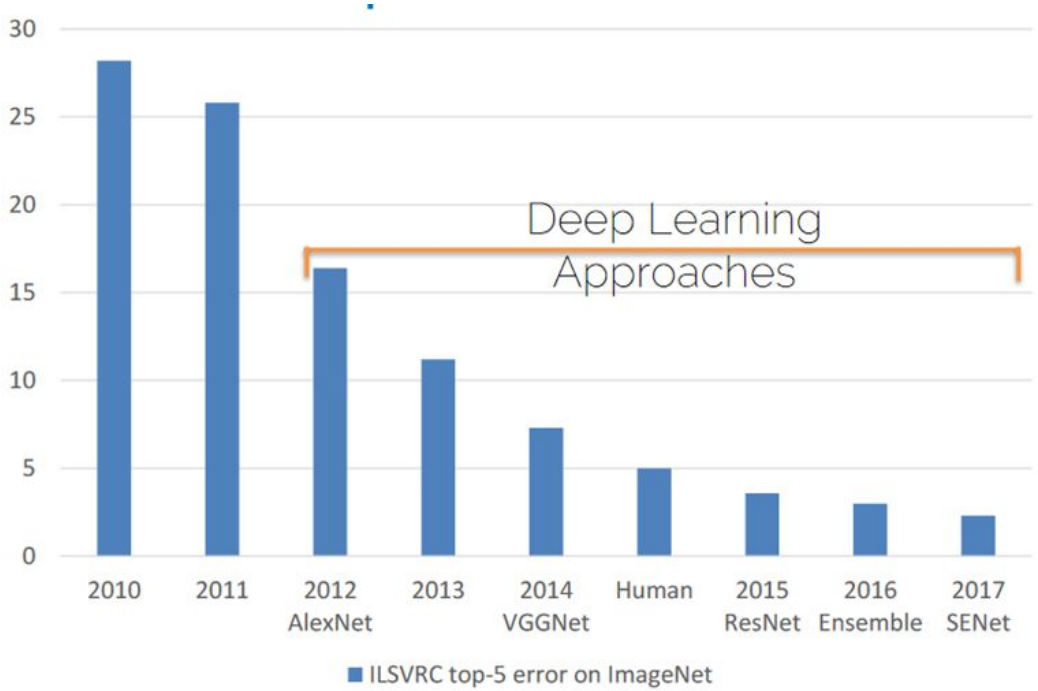


Deep Learning Software

02476 Machine Learning Operations

Nicki Skafte Detlefsen

The Deep Learning Revolution

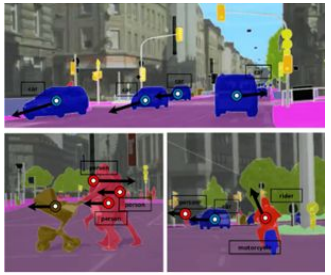


Since then

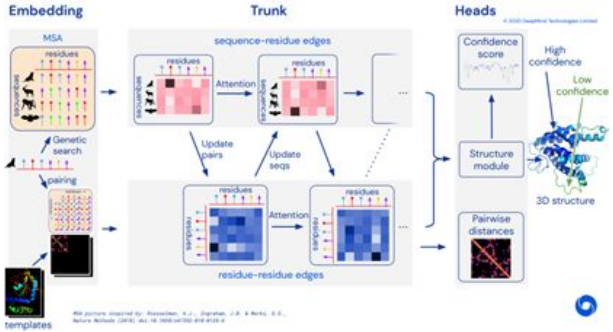
AlphaGO:
Beating humans

```
Q: What is your favorite animal?
A: My favorite animal is a dog.

Q: Why?
A: Because dogs are loyal and friendly.

Q: What are two reasons that a dog might be in a bad mood?
A: Two reasons that a dog might be in a bad mood are if it is hungry or if it is hot.

Q: How many eyes does a giraffe have?
A: A giraffe has two eyes.

Q: How many legs does a frog have?
A: A frog has four legs.

Q: Are there any animals with three legs?
A: No, there are no animals with three legs.

Q: Why don't animals have three legs?
A: Animals don't have three legs because they would fall over.
```

GPT-3:
Having conversations



U-net:
Real-time semantic segmentation

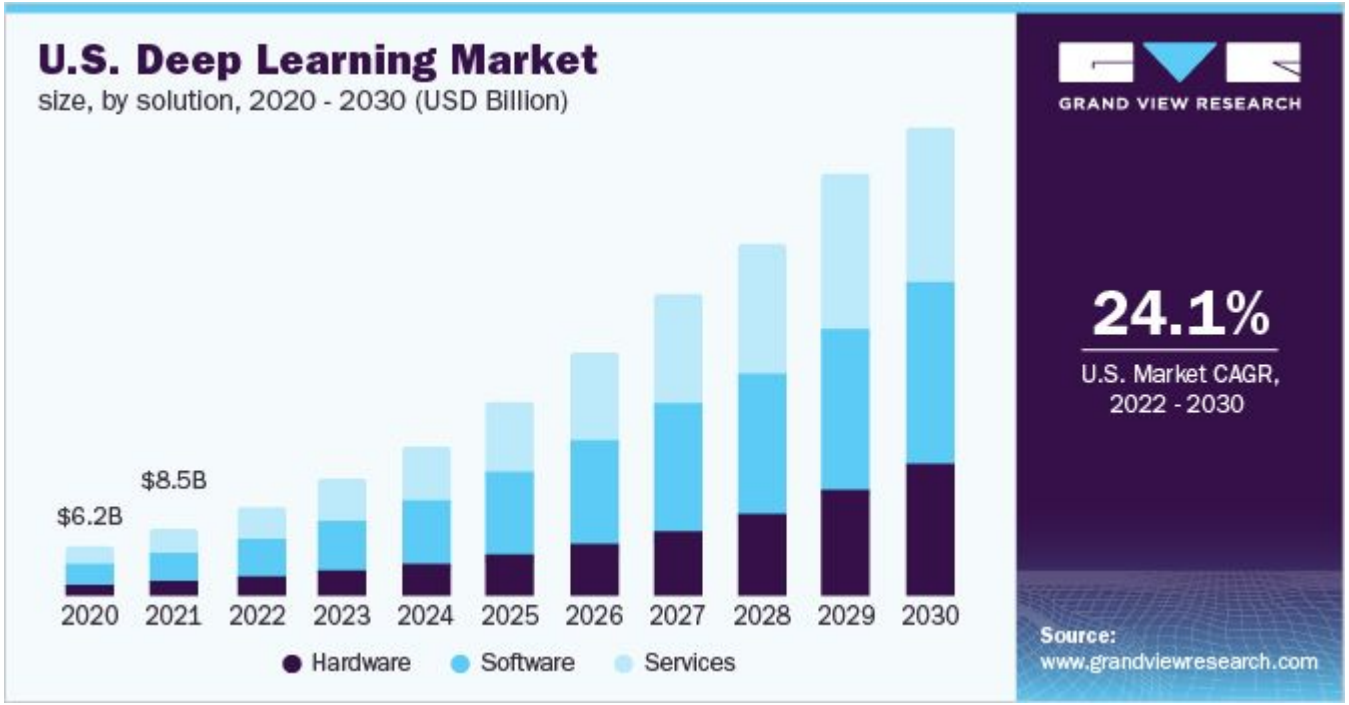


Alphafold:
Solving protein engineering

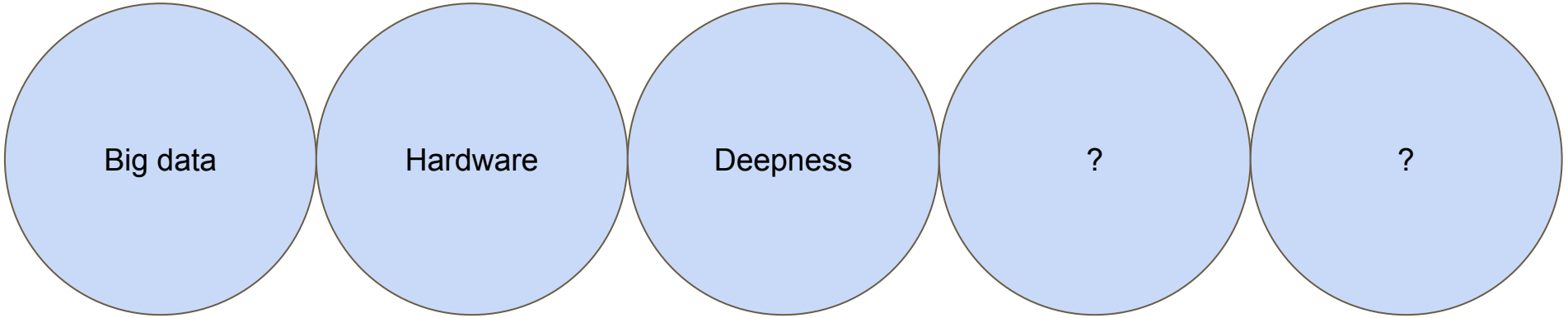
Stable diffusion:
Text to image translation

Why you should care

Usage of Deep learning in industry is increasing fast!



The drivers of the revolution

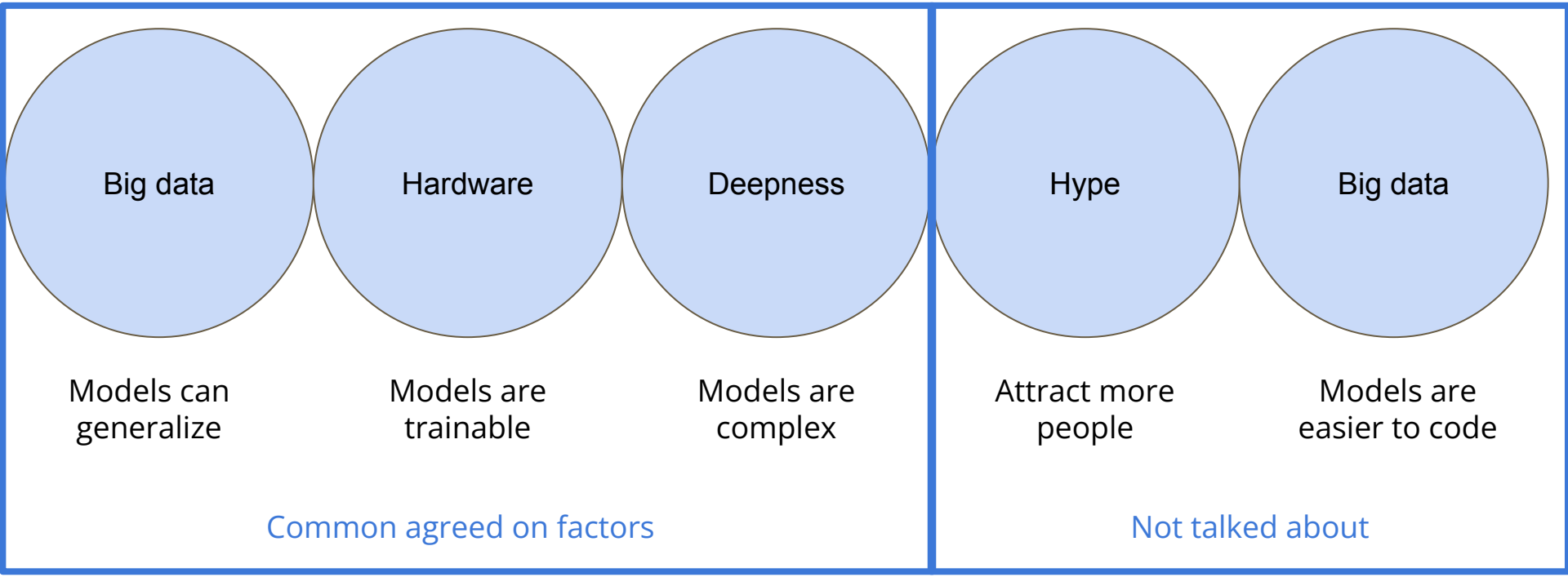


Models can generalize

Models are trainable

Models are complex

The drivers of the revolution



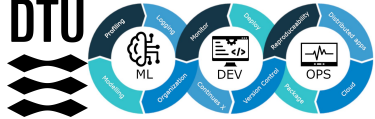
Why do we need frameworks for DL?

Deep learning is a lot of simple math

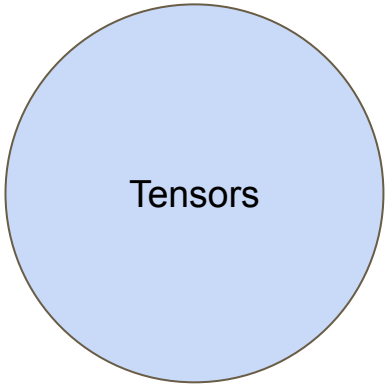
1. But we need to do it efficiently
2. We need to take care of hardware acceleration (=CUDA)
3. We need to take care of gradient backprob
4. Optimizers, data interface etc. also complicates thing

We do not really want to deal with...

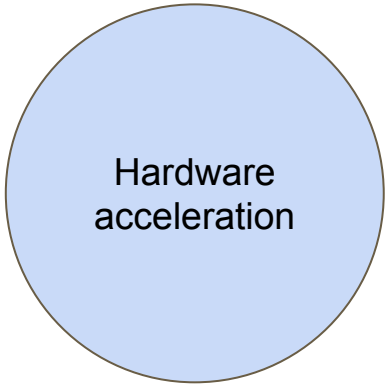
```python
import numpy as np

class Linear(object):
    def __init__(self, input_dim: int, num_hidden: int = 1):
        self.weight = np.random.randn(input_dim, num_hidden)
        self.bias = np.zeros(num_hidden)

    def __call__(self, x):
        self.x = x
        output = x @ self.weight + self.bias
        return output

    def backward(self, gradient):
        self.weight_gradient = self.x.T @ gradient
        self.bias_gradient = gradient.sum(axis=0)
        self.x_gradient = gradient @ self.weight
        return self.x_gradient

    def update(self, lr):
        self.weight = self.weight - lr * self.weight_gradient
        self.bias = self.bias - lr * self.bias_gradient


if __name__ == "__main__":
    x = np.random.randn(10, 5)
    layer = Linear(5, 1)
    y = layer(x)
    grad = layer.backward(np.ones((10, 5)))
    layer.update(1e-2)
```



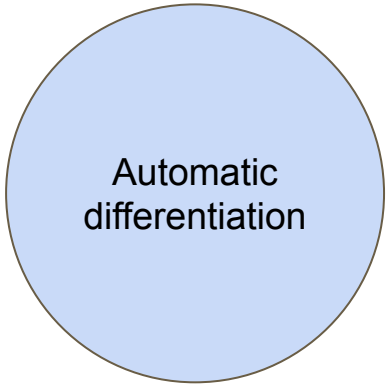
How to make a modern DL framework



Abstraction to higher order data

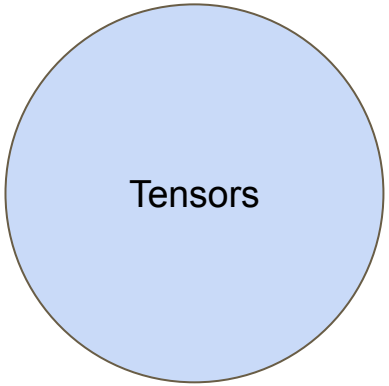


Faster computations

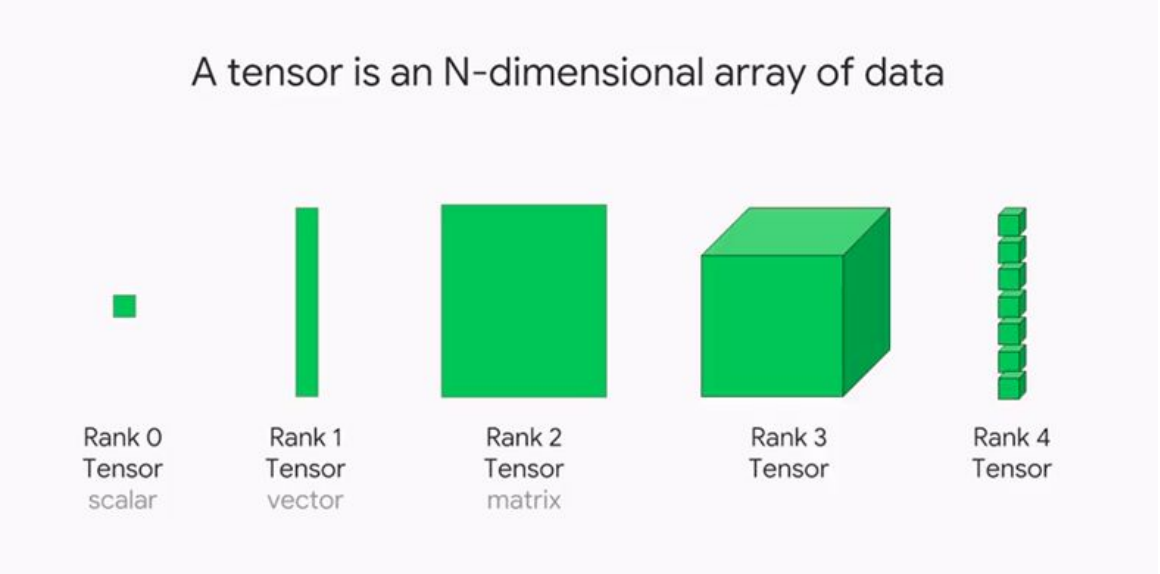


Ease of use

How to make a modern DL framework



Abstraction to higher order data



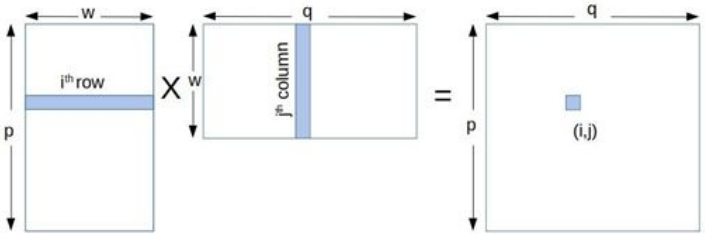
How to make a modern DL framework

CPU

GPU

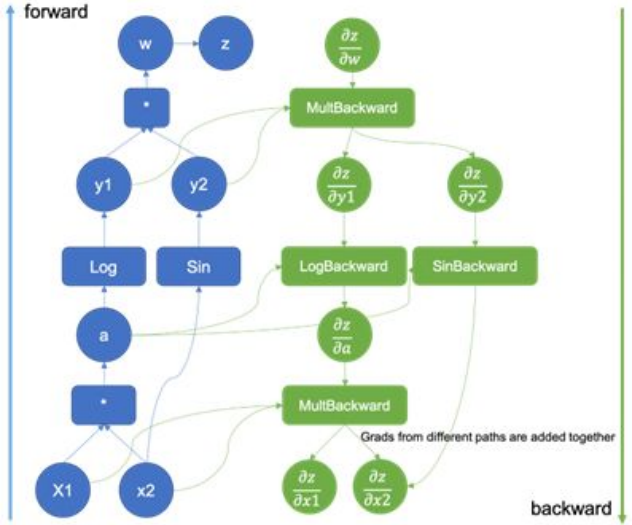
TPU

Hardware acceleration



Faster computations

How to make a modern DL framework



Automatic differentiation

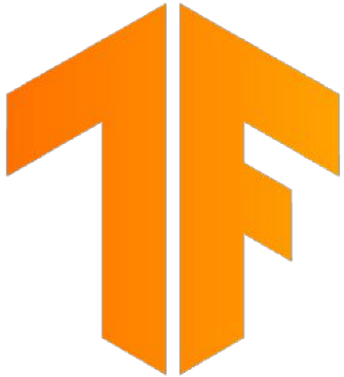
```
(base) C:\Users\nsde>python
Python 3.8.5 (default, Sep  3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> 2*torch.ones(5, requires_grad=True)
tensor([2., 2., 2., 2., 2.], grad_fn=<MulBackward0>)
>>>
```

Ease of use

The current landscape

Pytorch / Tensorflow / Jax all supply the same

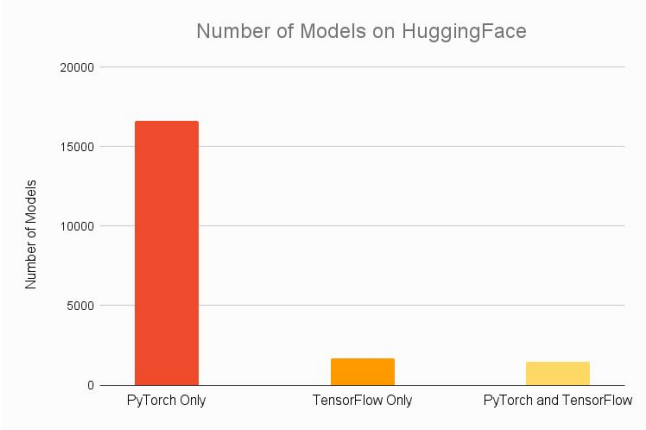
- Python interface
- Hardware acceleration
- Research and industry specific features



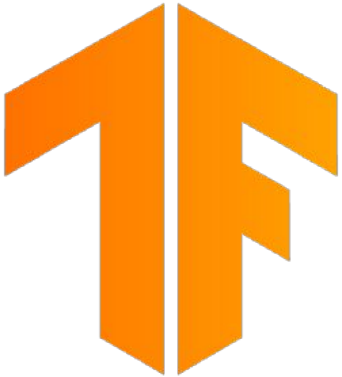
The current landscape

In this course we work with Pytorch because

- Absolutely dominant framework (#models, #papers, #competitions winners etc.)
- What we use in our research

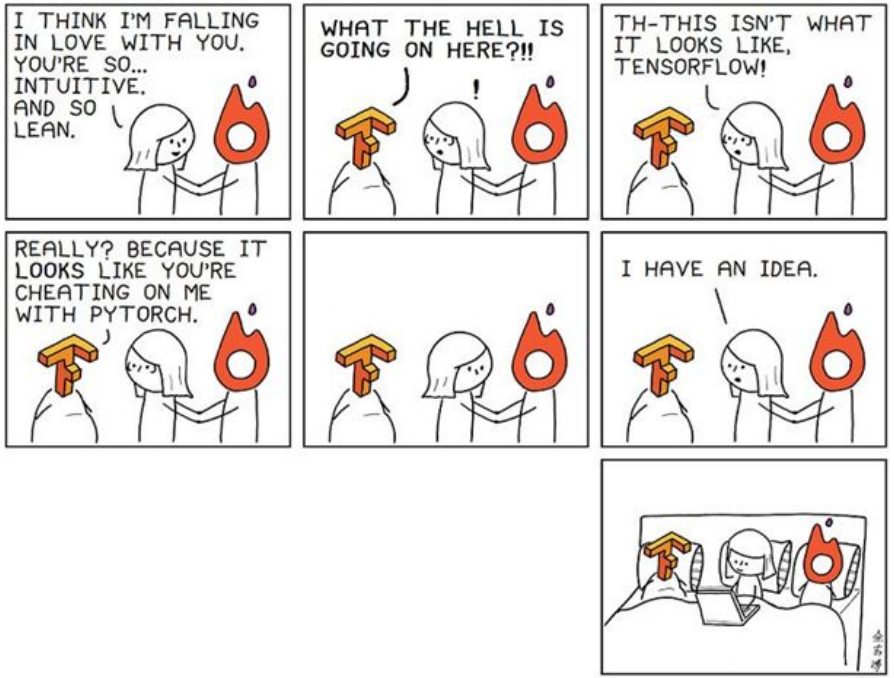


Reference



We highly recommend...

If you have the time, learn the basic of them all :)

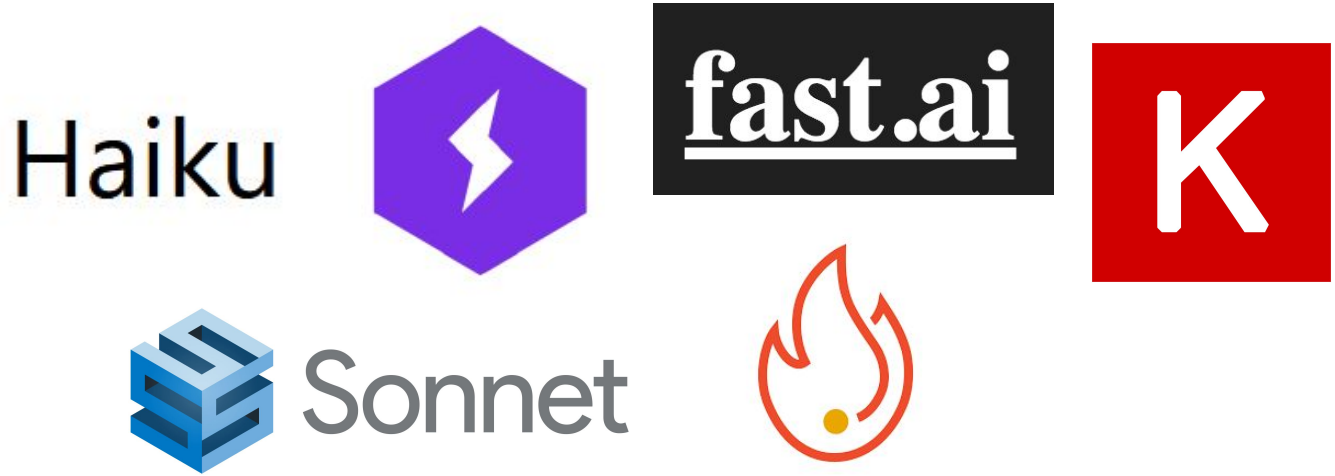


In practice, people often use high-level frameworks

Makes a lot of coding much easier.

Recommend, to only use these if you understand the underlying framework.

We get back top one of these.



Meme of the day