Pattern Analysis & Machine Intelligence
Praktikum: MLPR-SS21

Martin Mundt, Dr. Iuliia Pliushch, Prof. Dr. Visvanathan Ramesh
Goethe Uni Frankfurt

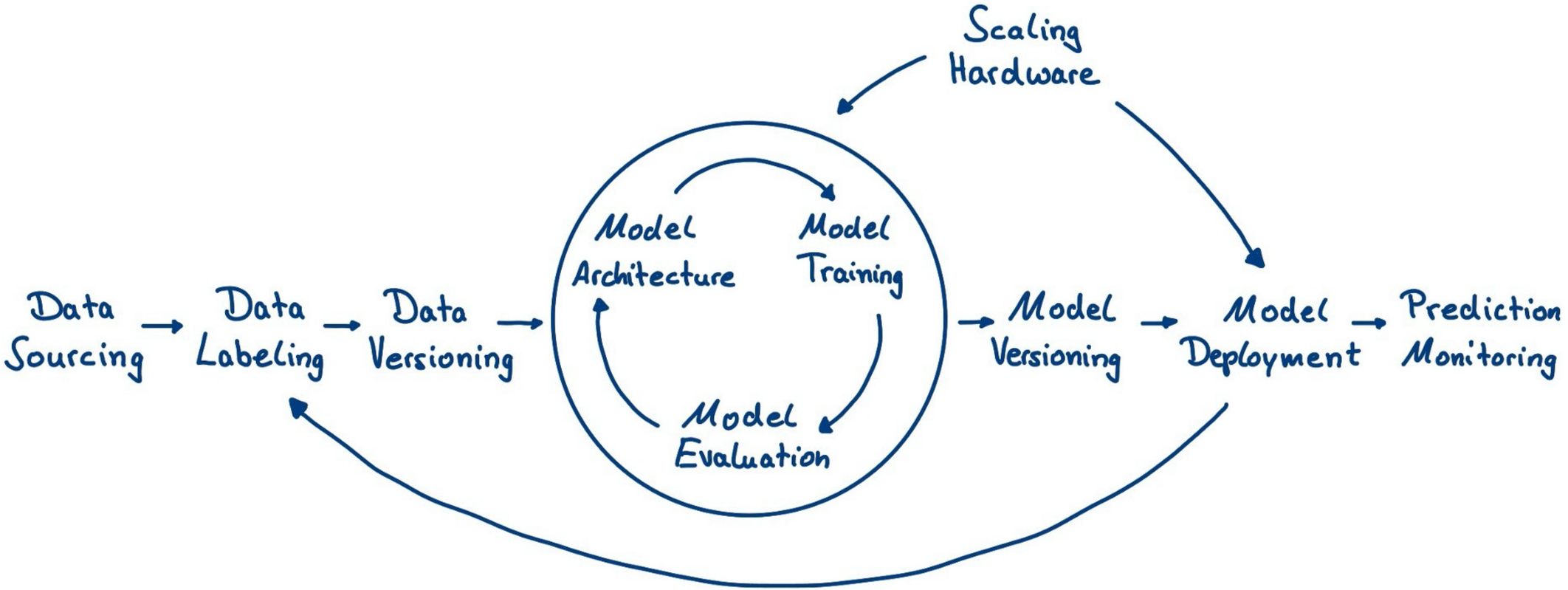
Week 5: Software Frameworks for AI & ML

Inspired by our previous lecture in Systems & Software Engineering II

“It is perhaps under appreciated how much machine learning frameworks shape ML research. They don’t just enable machine learning research. They enable and restrict the ideas that researchers are able to easily explore. How many nascent ideas are crushed simply because there is no easy way to express them in a framework?“

https://thegradient.pub/state-of-ml-frameworks-2019-pytorch-dominates-research-tensorflow-dominates-industry/

The machine learning workflow



https://medium.com/luminovo/the-deep-learning-toolset-an-overview-b71756016c06

AI & ML Software Frameworks

Bosch et al. 2019 – AI Engineering Concept

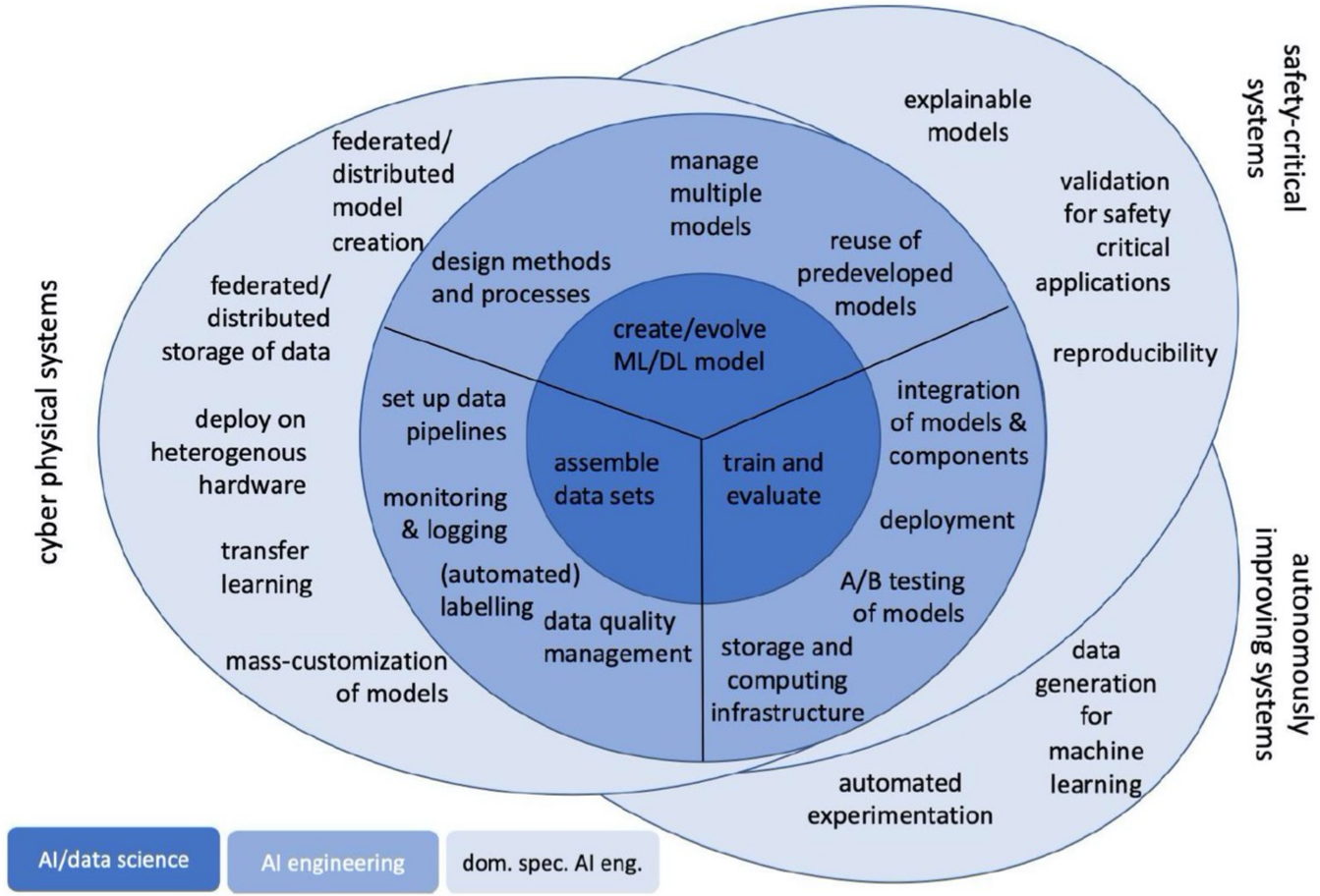


Figure 3: Conceptualization of AI engineering

- The inner to outer circles are reflected in, and to an extent even driven by, the historical development of software tools and corresponding ties to hardware advances.
- The software requirements are analogously constantly reshaped by the general advances in AI engineering, data science and domain specific progress. In addition requirements commonly shared across all software, such as OS interoperability, ease of installation & use for non-experts etc. apply.

PART I – ML Model Training

Establishing the core

AI & ML Software Frameworks

Recall lecture 2: Bosch et al. 2019 – AI Engineering Concept

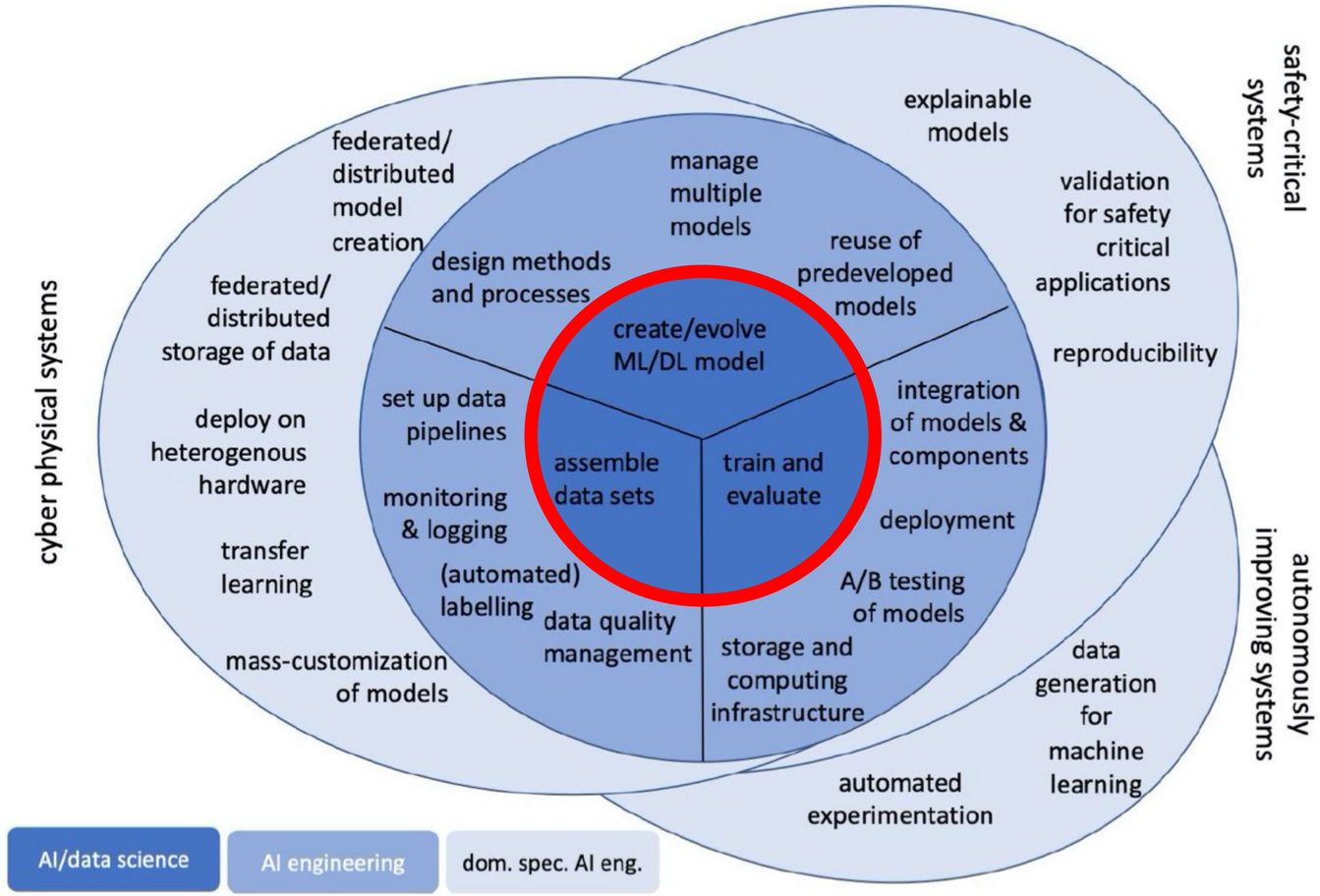


Figure 3: Conceptualization of AI engineering

- The inner to outer circles are reflected in, and to an extent even driven by, the historical development of software tools and corresponding ties to hardware advances.
- The software requirements are analogously constantly reshaped by the general advances in AI engineering, data science and domain specific progress. In addition requirements commonly shared across all software, such as OS interoperability, ease of installation & use for non-experts etc. apply.

Most core concepts have been well known for decades

Some key examples:

- **Automatic differentiation**. See e.g. Wengert (1964) or Rall, Louis B. (1981) for a review and software such as Maple (1982-today) or Mathematica (1988-today)

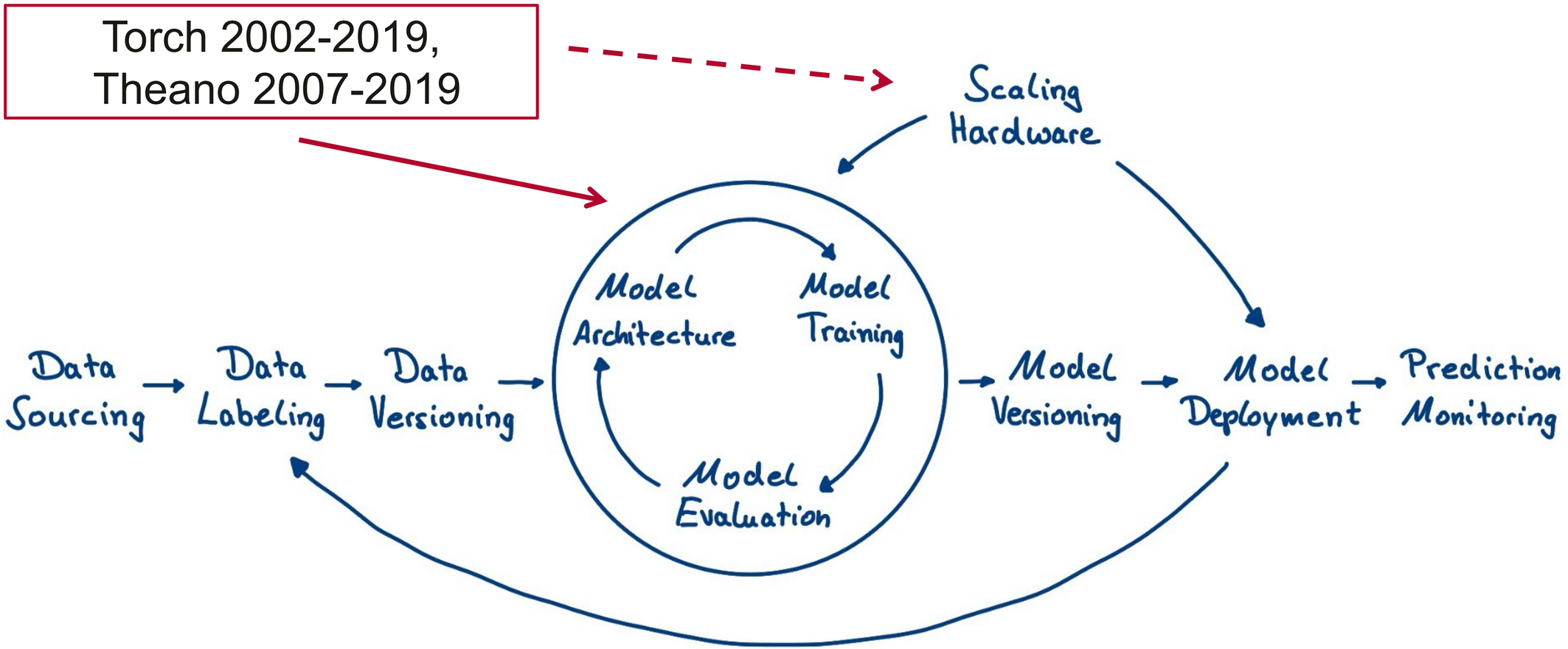
- Vast literature on **energy based numerical optimization** in natural sciences, and algorithmic techniques at the heart of machine learning such as **expectation maximization** (Dempster 1977) or **backpropagation** (Werbos 1983, Rummelhart 1986)

- Specific models such as **neural networks** (Rosenblatt 1961, Fukushima 1979), concepts such as **regression** or **decision trees, random forests** date back at least as much, if not even much further.

What are the enablers for the current wave?

→ Availability of data, computational power, specialized hardware, software tools.

Torch, Theano and the machine learning workflow



https://medium.com/luminovo/the-deep-learning-toolset-an-overview-b71756016c06

Torch and Theano: autodiff, abstractions & hardware interface

- Make differentiation easy. Theano through symbolic programming, Torch through reverse mode accumulation. Significantly facilitates numerical optimization.

- Started including code building blocks for common models such as neural network layers, logistic regression, random forests, support vector machines, clustering etc.

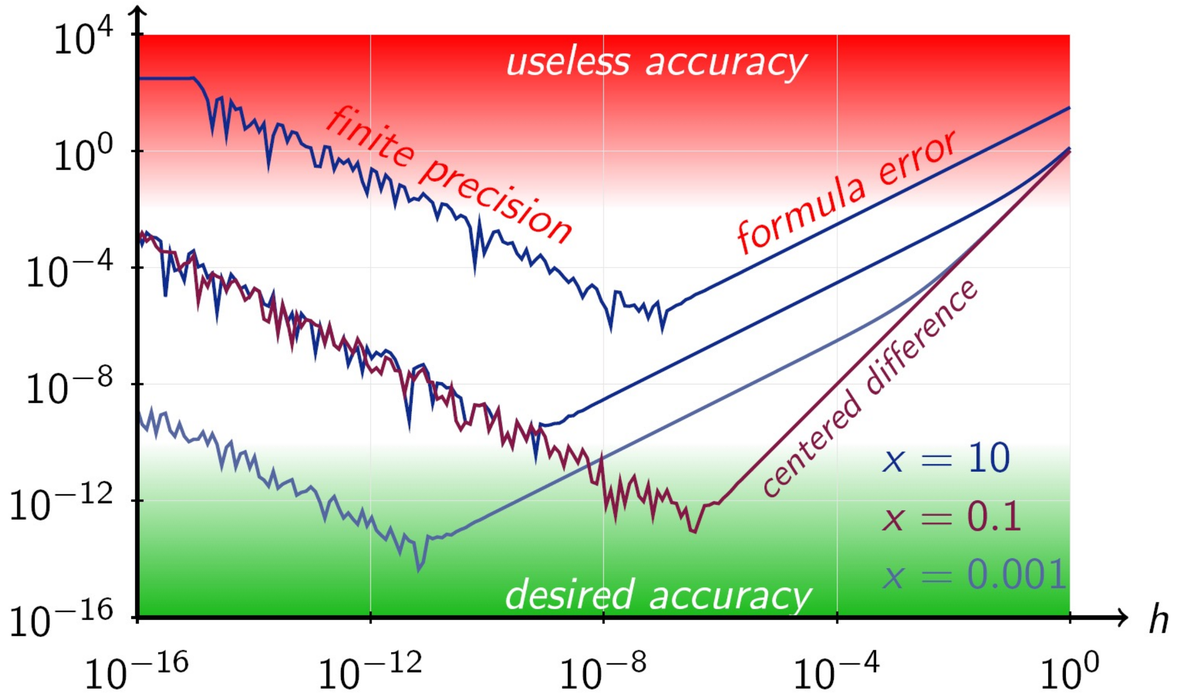
- Build on strong C99 matrix computation backend, starting to abstract away parallelization and hardware specific code from the developer to large degree. Integration with higher level programming languages such as Python or Lua.

Differentiation: from manual coding & finite differences to symbolic programming & automatic differentiation

Numerical derivatives

$$f'(x) = \frac{-f(x+2h) + 8f(x+h) - 8f(x-h) + f(x-2h)}{12h} + \frac{h^4}{30} f^{(5)}(c)$$

$$\frac{f(x+h) - f(x)}{h}.$$



By Berland - Self-made using TikZ, Beamer and LaTeX, Public Domain, https://commons.wikimedia.org/w/index.php?curid=4062778

Differentiation: from manual coding & finite differences to symbolic programming & automatic differentiation

Symbolic programming: Theano example

```
>>> import numpy
>>> import theano.tensor as T
>>> from theano import function
>>> x = T.dscalar('x')
>>> y = T.dscalar('y')
>>> z = x + y
>>> f = function([x, y], z)
```

```
>>> f(2, 3)
array(5.0)
```

If you are following along and typing into an interpreter, you may have noticed that there was a slight delay in executing the `function` instruction. Behind the scene, *f* was being compiled into C code.

http://deeplearning.net/software/theano/tutorial/gradients.html

Differentiation: from manual coding & finite differences to symbolic programming & automatic differentiation

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Symbolic programming: Theano example

```
>>> import numpy
>>> import theano
>>> import theano.tensor as T
>>> from theano import pp
>>> x = T.dscalar('x')
>>> y = x ** 2
>>> gy = T.grad(y, x)
>>> pp(gy)  # print out the gradient prior to optimization
'((fill((x ** TensorConstant{2}), TensorConstant{1.0}) * TensorConstant{2}) * (x ** (TensorCons
>>> f = theano.function([x], gy)
>>> f(4)
array(8.0)
```

http://deeplearning.net/software/theano/tutorial/gradients.html

Differentiation: from manual coding & finite differences to symbolic programming & automatic differentiation

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Symbolic programming: Theano example

Note

The optimizer simplifies the symbolic gradient expression. You can see this by digging inside the internal properties of the compiled function.

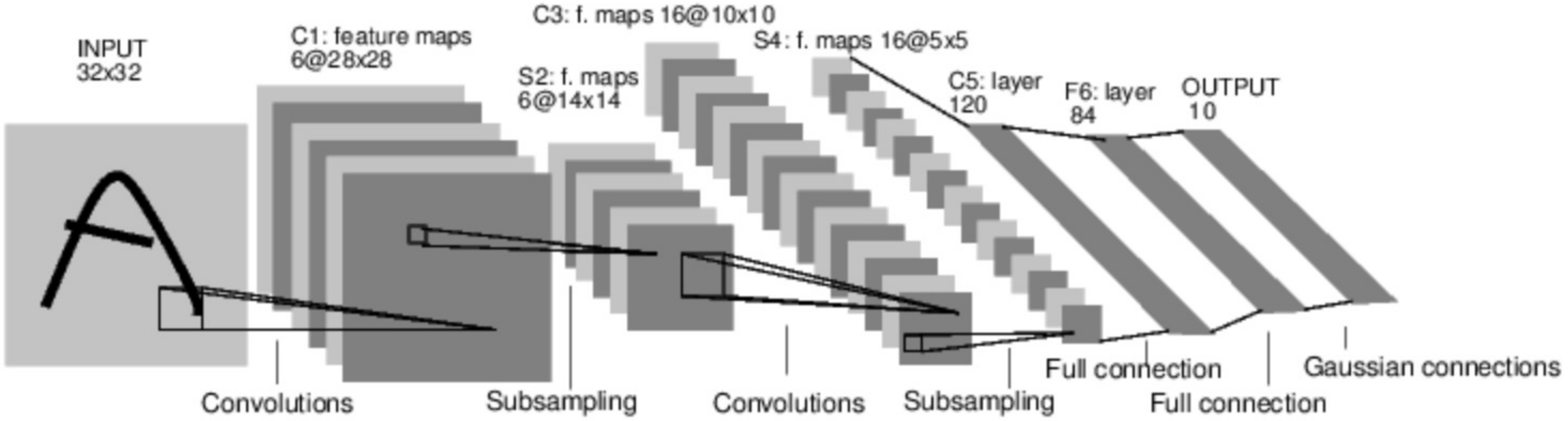
```
pp(f.maker.fgraph.outputs[0])
'(2.0 * x)'
```

After optimization there is only one Apply node left in the graph, which doubles the input.

http://deeplearning.net/software/theano/tutorial/gradients.html

Introduction of model building blocks and automatic differentiation

For example, look at this network that classfies digit images:



It is a simple feed-forward network.

It takes the input, feeds it through several layers one after the other, and then finally gives the output.

Theano and Torch started offering code building blocks for deep models, consisting of cascades of common operations. Here the so called “LeNet” (LeCun et al. 1989) that was used to recognize handwritten postal codes and later handwritten characters (1998), which was eventually adapted for processing in ATM machines

Introduction of model building blocks with automatic differentiation

```
net = nn.Sequential()
net:add(nn.SpatialConvolution(1, 6, 5, 5)) -- 1 input image channel, 6 output channels, 5x5 convol
ution kernel
net:add(nn.ReLU())                         -- non-linearity
net:add(nn.SpatialMaxPooling(2,2,2,2))     -- A max-pooling operation that looks at 2x2 windows an
d finds the max.
net:add(nn.SpatialConvolution(6, 16, 5, 5))
net:add(nn.ReLU())                         -- non-linearity
net:add(nn.SpatialMaxPooling(2,2,2,2))
net:add(nn.View(16*5*5))                   -- reshapes from a 3D tensor of 16x5x5 into 1D tensor
of 16*5*5
net:add(nn.Linear(16*5*5, 120))            -- fully connected layer (matrix multiplication betwee
n input and weights)
net:add(nn.ReLU())                         -- non-linearity
net:add(nn.Linear(120, 84))
net:add(nn.ReLU())                         -- non-linearity
net:add(nn.Linear(84, 10))                 -- 10 is the number of outputs of the network (in thi
s case, 10 digits)
net:add(nn.LogSoftMax())                   -- converts the output to a log-probability. Useful f
or classification problems

print('Lenet5\n' .. net:__tostring());
```

```
input = torch.rand(1,32,32) -- pass a random tensor as input to the network
```

```
output = net:forward(input)
```

https://github.com/soumith/cvpr2015/blob/master/Deep%20Learning%20with%20Torch.ipynb

Introduction of model building blocks with automatic differentiation

Why is this special?

In torch, loss functions are implemented just like neural network modules, and have automatic differentiation.
They have two functions - `forward(input, target)`, `backward(input, target)`

For example:

```
criterion = nn.ClassNLLCriterion() -- a negative log-likelihood criterion for multi-class classifi
cation
criterion:forward(output, 3) -- let's say the groundtruth was class number: 3
gradients = criterion:backward(output, 3)
```

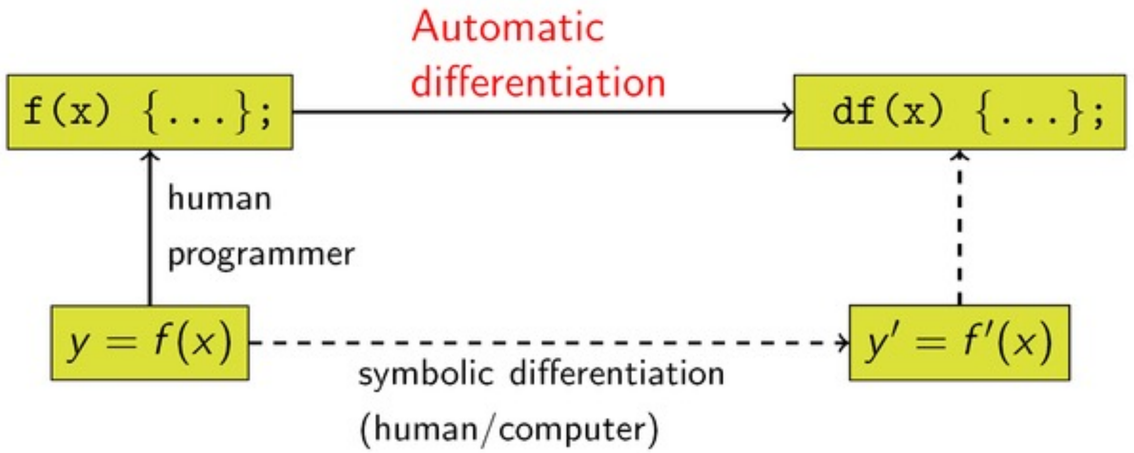
```
gradInput = net:backward(input, gradients)
```

https://github.com/soumith/cvpr2015/blob/master/Deep%20Learning%20with%20Torch.ipynb

Introduction of model building blocks with automatic differentiation

The backbone of automatic differentiation is a technique called “forward” or “reverse mode” accumulation.

In contrast to symbolic programming that can create complicated expressions or numerical challenges in finite differences, automatic differentiation makes use of the fact that *every complicated operation is built from a small set of primitive operations such as addition, multiplication or trigonometric functions*.



https://commons.wikimedia.org/wiki/File:AutomaticDifferentiationNutshell.png#/media/File:AutomaticDifferentiationNutshell.png

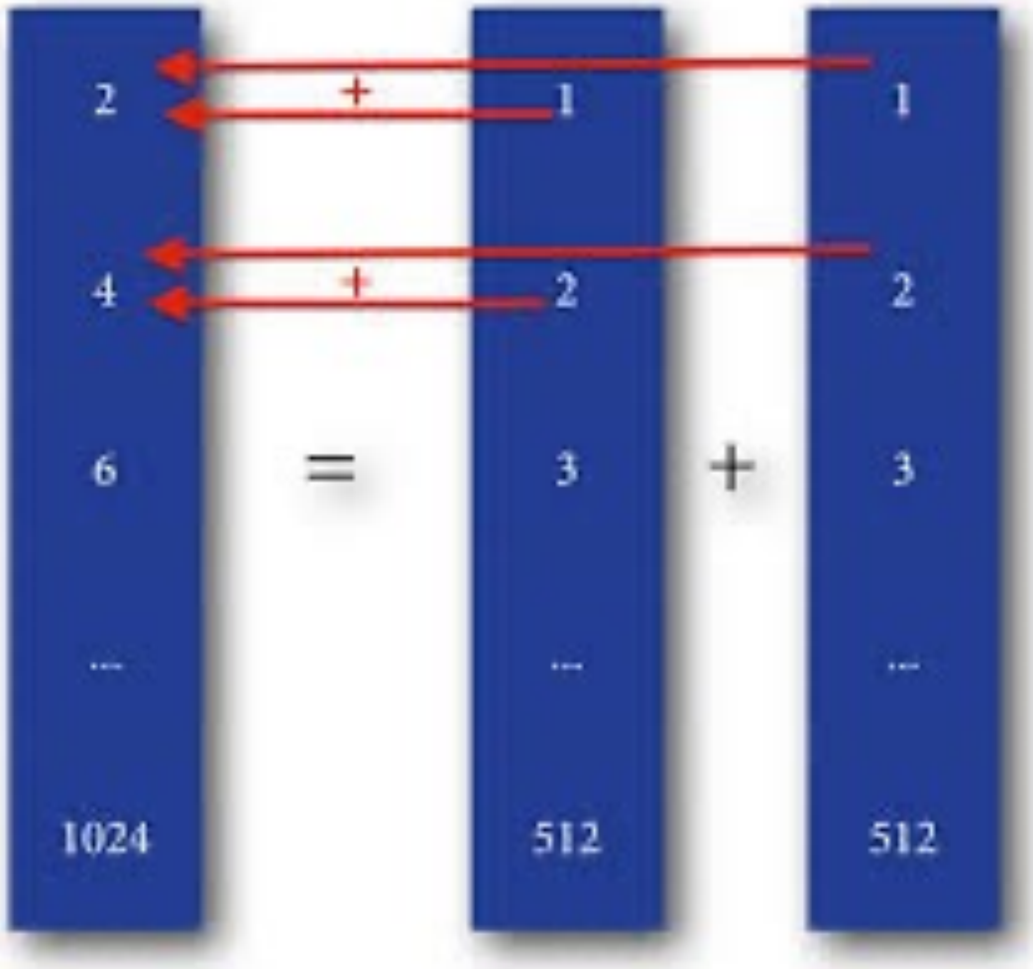
Automatic differentiation tracks operations (e.g. on a tape) and makes use of the chain rule of differentiation. Reverse or forward mode simply specifies the computational order.

An example of a good in-depth tutorial is https://rufflewind.com/2016-12-30/reverse-mode-automatic-differentiation

Importance of hardware: Graphics Processing Units (GPUs) towards general purpose computing (GPGPU)

Let's do a small tour de force in parallelization with an example of vector addition to better understand and appreciate what we are going to see next in the context of ML software frameworks

Think of something easy as vector addition or the respective Hadamard product. Each vector element is independent of the other. Ideally, we could calculate them all at the same time, in parallel!



https://www.quantstart.com/articles/Vector-Addition-Hello-World-Example-with-CUDA-on-Mac-OSX/

Importance of hardware: Graphics Processing Units (GPUs) towards general purpose computing (GPGPU)

This is fairly straightforward in standard C code executed on a CPU.

Here is an example as a refresher.

https://subscription.packtpub.com/book/programming/9781788996242/1/ch01lvl1sec04/vector-addition-using-cuda

```
#include<stdio.h>
#include<stdlib.h>

#define N 512

void host_add(int *a, int *b, int *c) {
    for(int idx=0;idx<N;idx++)
        c[idx] = a[idx] + b[idx];
}

//basically just fills the array with index.
void fill_array(int *data) {
    for(int idx=0;idx<N;idx++)
        data[idx] = idx;
}

void print_output(int *a, int *b, int*c) {
    for(int idx=0;idx<N;idx++)
        printf("\n %d + %d = %d", a[idx] , b[idx], c[idx]);
}
```

```
int main(void) {
    int *a, *b, *c;
    int size = N * sizeof(int);
   // Alloc space for host copies of a, b, c and setup input values
    a = (int *)malloc(size); fill_array(a);
    b = (int *)malloc(size); fill_array(b);
    c = (int *)malloc(size);
    host_add(a,b,c);
    print_output(a,b,c);
    free(a); free(b); free(c);
    return 0;
}
```

Importance of hardware: Graphics Processing Units (GPUs) towards general purpose computing (GPGPU)

When we use a GPU we now also need to worry about:

- **Managing the GPU memory as a separate device**
- **Transferring arrays back & forth**
- Writing the code to parallelize on GPU
- The memory layout of the GPU

Let's take a look at NVIDIA CUDA C

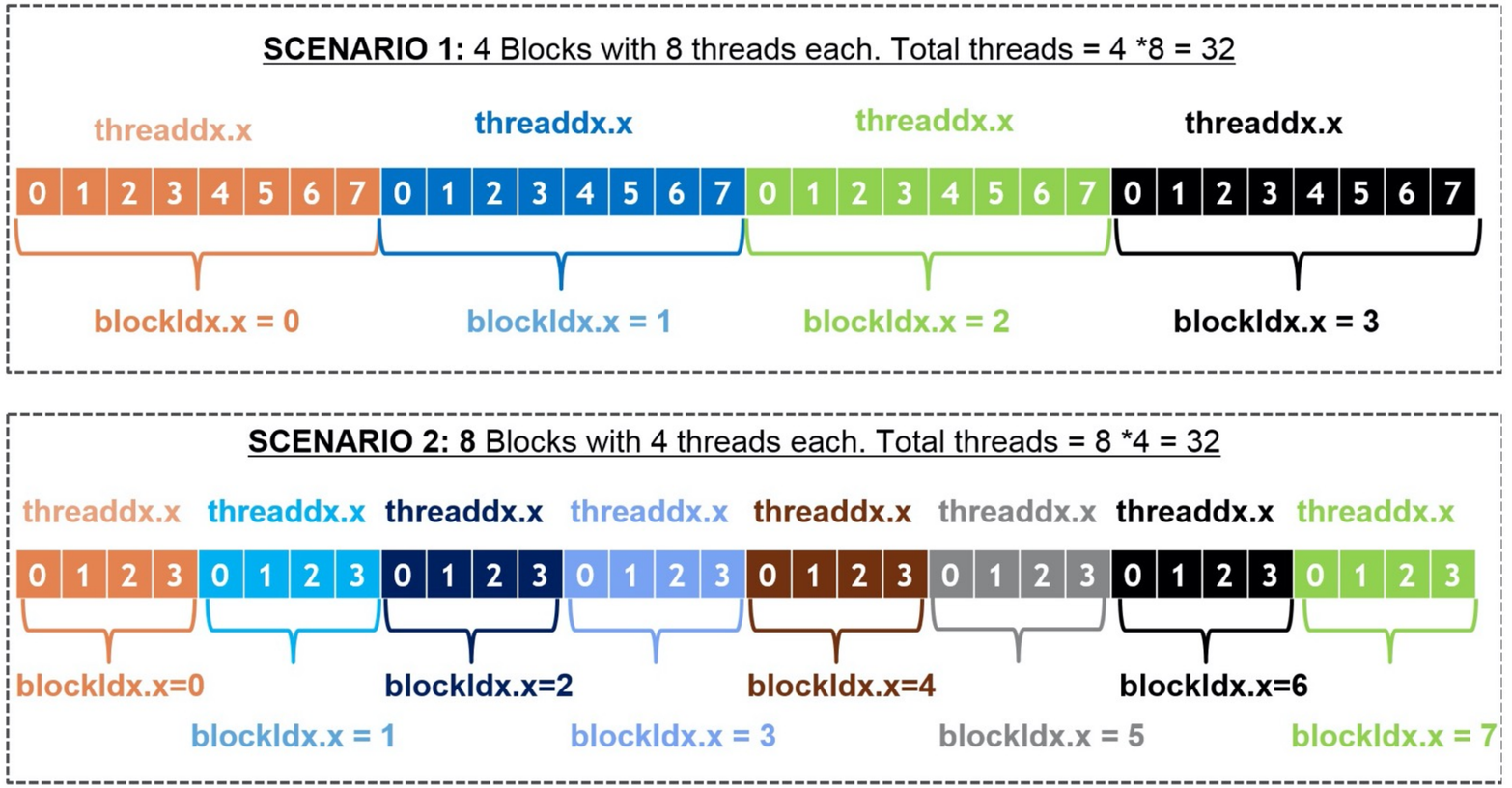
https://subscription.packtpub.com/book/programming/9781788996242/1/ch01lvl1sec04/vector-addition-using-cuda

```
int main(void) {
    int *a, *b, *c;
    int *d_a, *d_b, *d_c; // device copies of a, b, c
    int size = N * sizeof(int);

    // Alloc space for host copies of a, b, c and setup input values
    a = (int *)malloc(size); fill_array(a);
    b = (int *)malloc(size); fill_array(b);
    c = (int *)malloc(size);

    // Alloc space for device copies of vector (a, b, c)
    cudaMalloc((void *)&d_a, N * sizeof(int));
    cudaMalloc((void *)&d_b, N *sizeof(int));
    cudaMalloc((void *)&d_c, N * sizeof(int));

    // Copy from host to device
    cudaMemcpy(d_a, a, N * sizeof(int), cudaMemcpyHostToDevice);
    cudaMemcpy(d_b, b, N* sizeof(int), cudaMemcpyHostToDevice);

    device_add<<<1,1>>>(d_a,d_b,d_c);

    // Copy result back to host
    cudaMemcpy(c, d_c, N * sizeof(int), cudaMemcpyDeviceToHost);

    print_output(a,b,c);
    free(a); free(b); free(c);

    //free gpu memory
    cudaFree(d_a); cudaFree(d_b); cudaFree(d_c);

    return 0;
}
```

Importance of hardware: Graphics Processing Units (GPUs) towards general purpose computing (GPGPU)

When we use a GPU we now also need to worry about:

- Managing the GPU memory as a separate device
- Transferring arrays back & forth
- Writing the code to parallelize on GPU
- **The memory layout of the GPU**



https://subscription.packtpub.com/book/programming/9781788996242/1/ch01lvl1sec04/vector-addition-using-cuda

```
__global__ void device_add(int *a, int *b, int *c) {
     int index = threadIdx.x + blockIdx.x * blockDim.x;
     c[index] = a[index] + b[index];
}
```

Importance of hardware: parallelized, accelerated computing -> towards large scale!

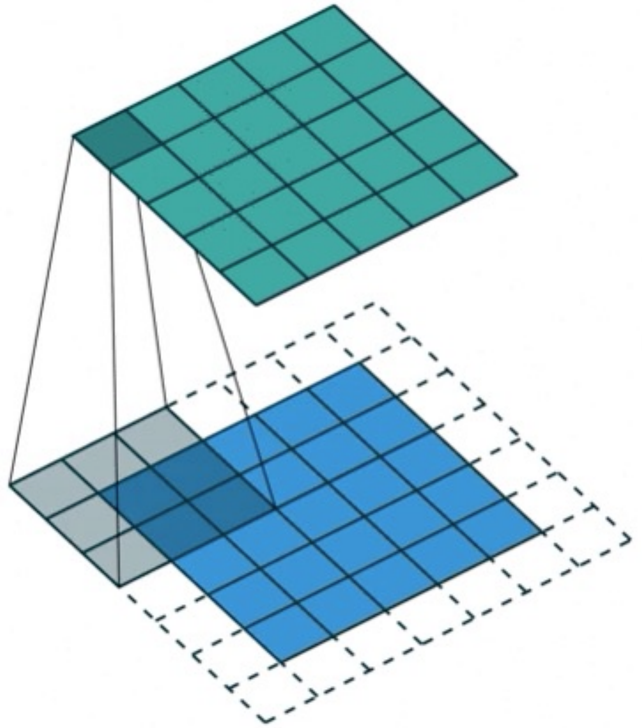
Remember that for the earlier seen neural network LeNet, we plug together modules consisting of convolutions & other operations and can then made use of autodiff.

A basic building block of current machine learning frameworks is the strided 2D convolution. Most frameworks provide a primitive operation that accepts N input images of size $H \times W$, where each pixel has a "depth" of C_i channels[2]. Informally, for a "kernel size" $K=3$ and "stride" $S=2$, conv2d computes a weighted sum of overlapping 3×3 patches of pixels centered at every other (x, y) coordinate, to produce N smaller images with pixel depth C_o (Figure 1). Mathematically, this can be expressed as follows:

$$\forall n, x, y, c_o : O^{n,c_o}_{x,y} = \sum_{k_x} \sum_{k_y} \sum_{c_i} I^{n,c_i}_{sx+k_x, sy+k_y} \cdot K^{c_i,c_o}_{k_x,k_y} \quad (1)$$

where $\cdot$ denotes scalar multiplication, and O, I, and K are all 4-dimensional arrays of scalars. The resulting code is little more than 7 nested loops around a multiply-accumulate operation, but array layout, vectorization,

https://dl.acm.org/doi/pdf/10.1145/3317550.3321441

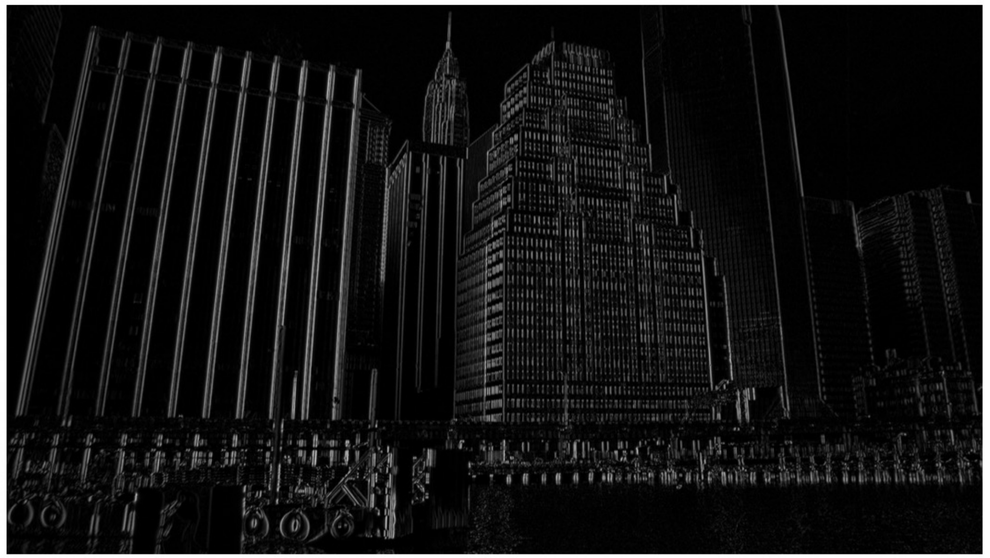


https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

Importance of hardware: parallelized, accelerated computing -> towards large scale!

Convolution is a prime example for the benefits of parallelization because it contains elementwise multiplication between the filter (independently of whether this is learned in ML or modelled explicitly) & a portion of the image, and each spatial position is entirely independent of the other.

Figure 11. The image used in the evaluation of the kernel function.

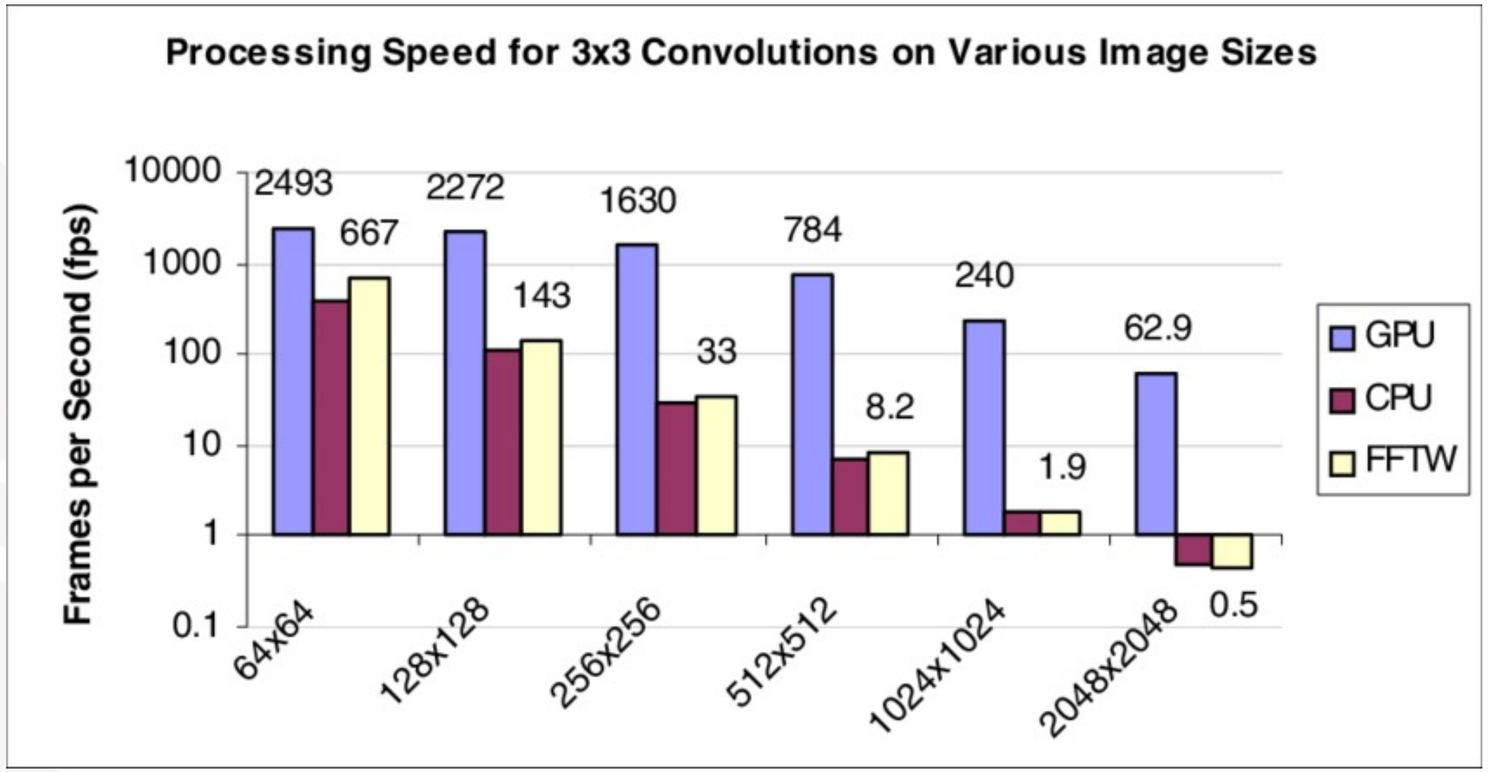


The result of the 3x3 filter.

https://qiita.com/naoyuki_ichimura/items/8c80e67a10d99c2fb53c

Importance of hardware: parallelized, accelerated computing -> towards large scale!

You can see a corresponding ”easy” CUDA implementation here: https://qiita.com/naoyuki_ichimura/items/8c80e67a10d99c2fb53c . It does not fully optimize for memory layout, but you can imagine that the code gets increasingly complicated.



https://www.researchgate.net/publication/220857904_Accelerated_2D_image_processing_on_GPUs

Importance of hardware: Graphics Processing Units (GPUs) towards general purpose computing (GPGPU)

Easy parallelized, accelerated computing for everyone. Tensors, models and optimization on GPU using a strong C backend with a single line Lua or Python interface!

cunn: neural networks on GPUs using CUDA

```
require 'cunn';
```

The idea is pretty simple. Take a neural network, and transfer it over to GPU:

```
net = net:cuda()
```

Also, transfer the criterion to GPU:

```
criterion = criterion:cuda()
```

Ok, now the data:

```
trainset.data = trainset.data:cuda()
trainset.label = trainset.label:cuda()
```

Okay, let's train on GPU :) #sosimple

https://github.com/soumith/cvpr2015/blob/master/Deep%20Learning%20with%20Torch.ipynb

PART II – Beyond Optimization

A high-speed framework race

AI & ML Software Frameworks

Recall lecture 2: Bosch et al. 2019 – AI Engineering Concept

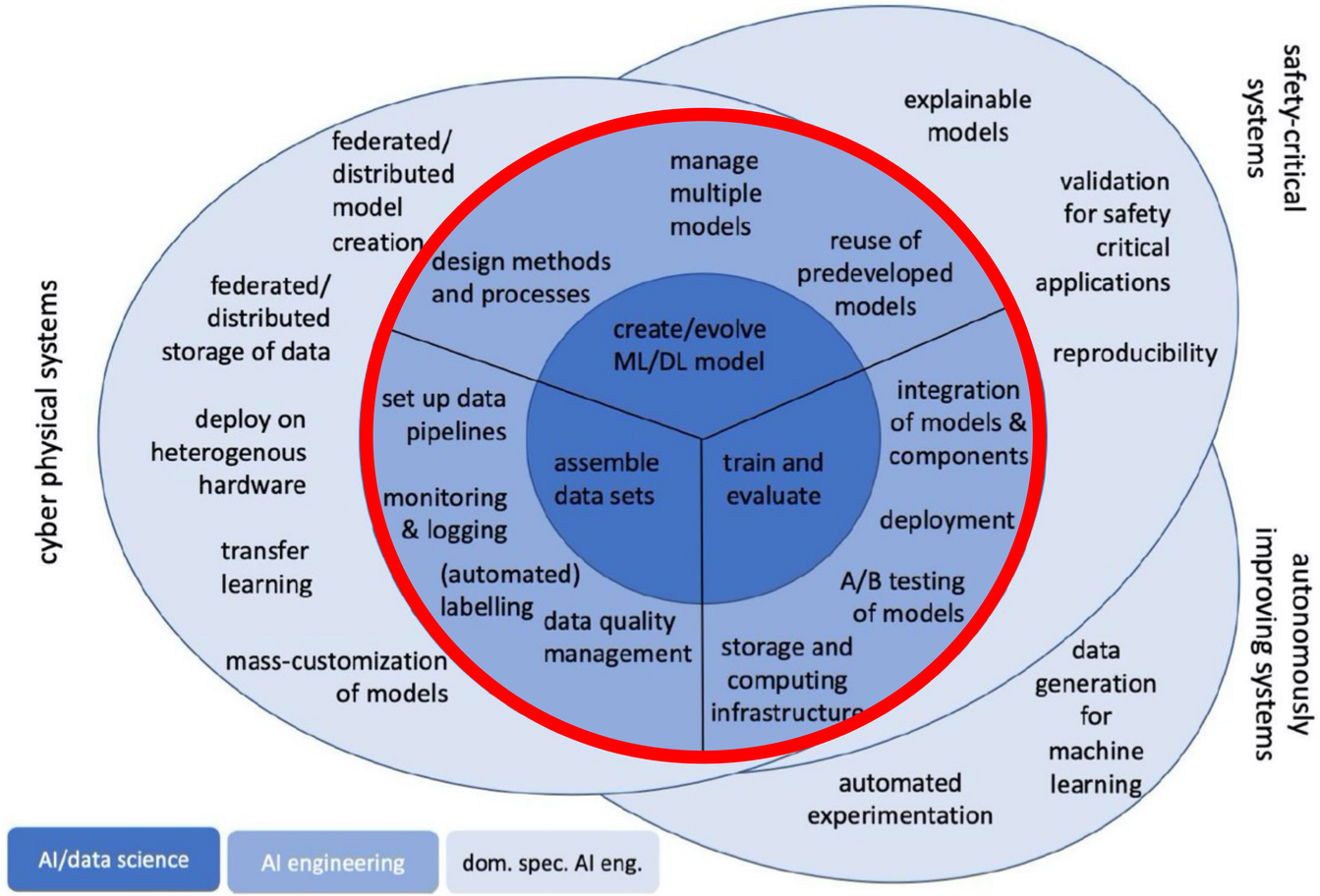


Figure 3: Conceptualization of AI engineering

- The inner to outer circles are reflected in, and to an extent even driven by, the historical development of software tools and corresponding ties to hardware advances.
- The software requirements are analogously constantly reshaped by the general advances in AI engineering, data science and domain specific progress. In addition requirements commonly shared across all software, such as OS interoperability, ease of installation & use for non-experts etc. apply.

The emergence of ML frameworks: start of a race

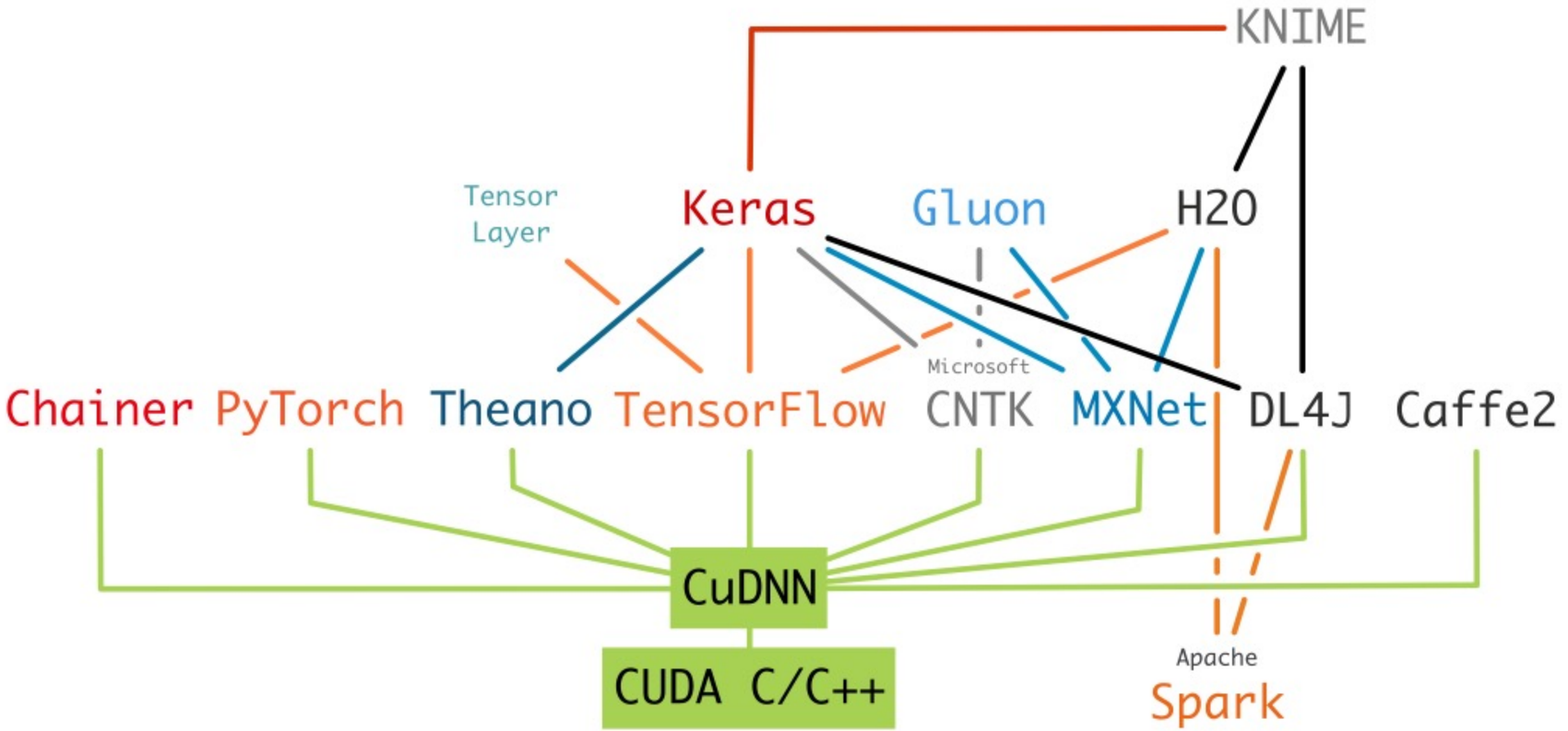


Fig. 3 The most popular Deep Learning frameworks and libraries layering in various abstraction implementation levels

Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey, Nguyen et al. 2019

The emergence of ML frameworks: start of a race

- Many frameworks appear and leverage the advances made by Torch and Theano
- The core remains: CUDA C backend and automatic differentiation
- More emphasis on clean interface to programming languages, mainly Python
- More layers for enhanced “ease of use” on top, abstracting away even more detail, e.g. Keras or Gluon
- More than just model optimization: extensions towards data pipelines, reuse of models , monitoring and logging convenience, tools and libraries

The emergence of ML frameworks: start of a race

We will not go into detail on individual advantages of the many existing software frameworks. Most are direct extensions to Theano or Torch, e.g. Chainer (Preferred Networks, 2015) or PyTorch (Facebook, 2016) continuing Torch's "define-by-run" or TensorFlow (Google, 2015) being the spiritual successor to Theano's static graphs.

As we will see in later parts of this presentation, the large majority of these has either converged to a common design, been discontinued or has never taken off in the first place.

We will instead focus on some selected significant advances that continue to influence and shape today's practices.

Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey (Nguyen et al, Artificial Intelligence Review 2019, Springer) provides an excellent review of the most popular frameworks between 2015-2019
https://link.springer.com/content/pdf/10.1007/s10462-018-09679-z.pdf

Static vs. dynamic graphs

```
import torch
matrix1 = torch.Tensor(3,3)
matrix2 = torch.Tensor(3,3)
product = torch.matmul(matrix1,matrix2)
print(product)
```

2016: PyTorch's graph is dynamically build

If you simply add another operation below above "print" statement with any mathematical operation making use of "product", this operation will be added as the next element in the graph.

```
import tensorflow as tf
sess = tf.Session()
matrix1 = tf.constant([[3.],[3.]])
matrix2 = tf.constant([[3.],[3.]])
product = tf.matmul(matrix1,matrix2)
result = sess.run(product)
print(result)
sess.close()
```

2016: TensorFlow's graph is static and needs to be predefined and is only executed when a "session is run".

It is not feasible to "drop in" an iteration after the graph has been built.

Static vs. dynamic graphs: neural network examples

```
from torch import nn as nn
import torch.nn.functional as F

class Model(nn.Module):
        def __init__(self):
                super(Model, self).__init__()
                self.input_size = 28*28
                self.conv1 = nn.Conv2d(1, 32, 5)
                self.mp1 = nn.MaxPool2d(2, 2)
                self.conv2 = nn.Conv2d(32, 64, 5)
                self.mp2 = nn.MaxPool2d(2,2)
                self.fc = nn.Linear(64*4*4, 10)

        def forward(self, x):
                x = self.mp1(F.relu(self.conv1(x)))
                x = self.mp2(F.relu(self.conv2(x)))
                x = x.view(-1, 64*4*4)
                x = self.fc1(x)
                return x

model = Model()
result = model(torch.rand(1,1,28,28))
print(result)
```

PYTORCH

```
    import tensorflow as tf
    x = tf.placeholder("float", [None, n_input])
    y = tf.placeholder("float", [None, n_classes])
    def multilayer_perceptron(_X, _weights, _biases):
        layer_1 = tf.nn.relu(tf.add(tf.matmul(_X, _weights['h1']), _biases['b1']))
        layer_2 = tf.nn.relu(tf.add(tf.matmul(layer_1, _weights['h2']),_biases['b2']))
        return tf.matmul(layer_2, weights['out']) + biases['out']
...
# Initialize variables
# Launch the graph
```

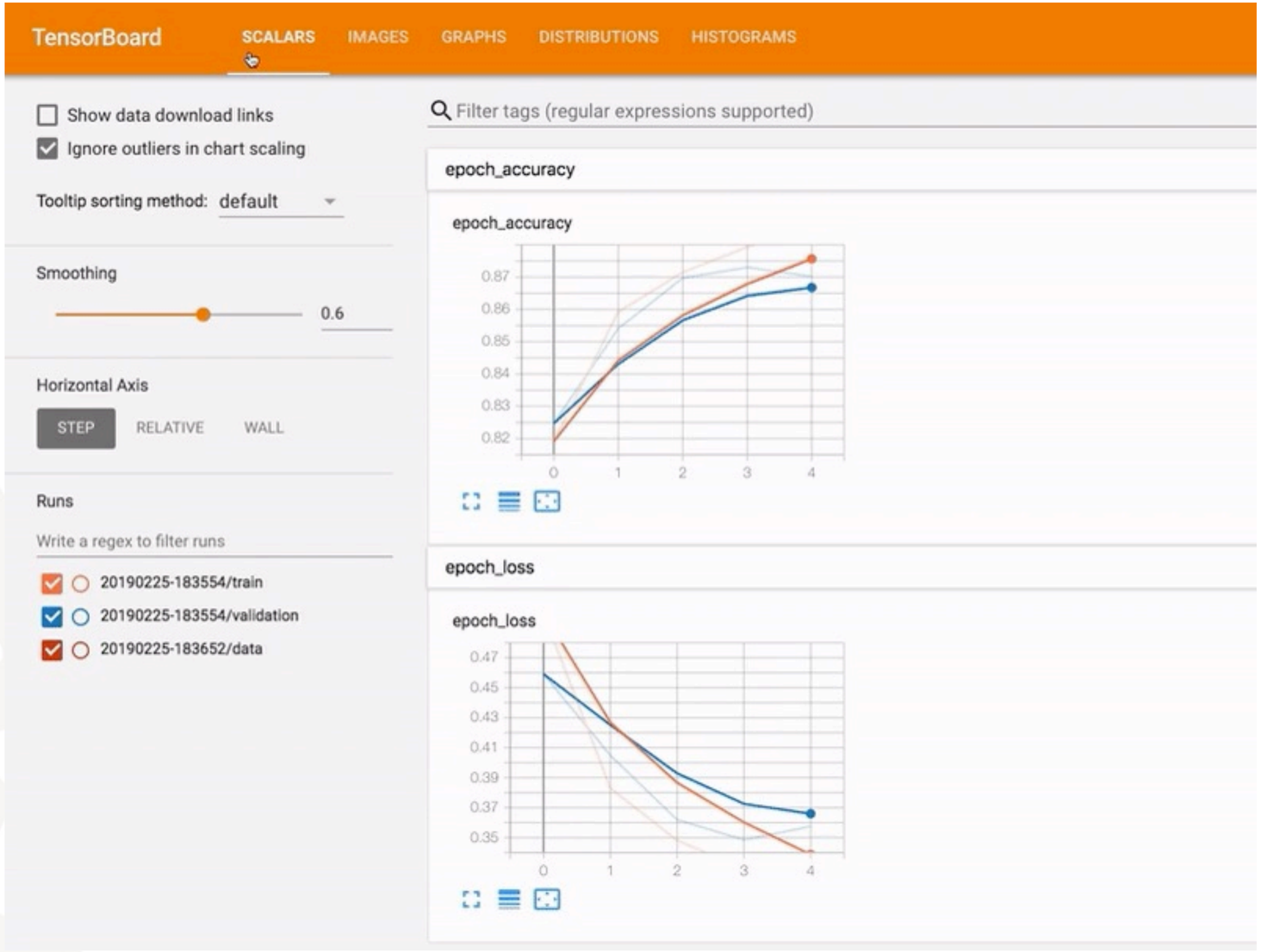
TensorFlow

Static vs. dynamic graphs: neural network examples

```
epochs = 5
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr = 0.01)
model = Model()

for e in range(epochs):
        for i, (inp, target) in enumerate(train_loader):
                output = model(inp)
                loss = criterion(output, target)

                optimizer.zero_grad()
                loss.backward()
                optimizer.step()

        # cross-validation, testing etc.
```

```
epochs = 5
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr = 0.01)

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = Model().to(device)

for e in range(epochs):
        for i, (inp, target) in enumerate(train_loader):
                inp = inp.to(device)
                target = target.to(device)

                output = model(inp)
                loss = criterion(output, target)

                optimizer.zero_grad()
                loss.backward()
                optimizer.step()
```

Community, tutorials and workflows



Community, tutorials and workflows

Getting Started

Deep Learning with PyTorch: A 60 Minute Blitz
Data Loading and Processing Tutorial
Learning PyTorch with Examples
Transfer Learning Tutorial
Deploying a Seq2Seq Model with the Hybrid Frontend
Saving and Loading Models
What is *torch.nn really*?

Image

TorchVision 0.3 Object Detection Finetuning Tutorial
Finetuning Torchvision Models
Spatial Transformer Networks Tutorial
Neural Transfer Using PyTorch
Adversarial Example Generation
Transfering a Model from PyTorch to Caffe2 and Mobile using ONNX

Text

Chatbot Tutorial
Generating Names with a Character-Level RNN
Classifying Names with a Character-Level RNN
Deep Learning for NLP with Pytorch
Translation with a Sequence to Sequence Network and Attention

Generative

DCGAN Tutorial

Reinforcement Learning

Reinforcement Learning (DQN) Tutorial

https://pytorch.org/tutorials/index.html

For beginners

Your first neural network

Train a neural network to classify images of clothing, like sneakers and shirts, in this fast-paced overview of a complete TensorFlow program.

For experts

Generative adversarial networks

Train a generative adversarial network to generate images of handwritten digits, using the Keras Subclassing API.

For experts

Neural machine translation with attention

Train a sequence-to-sequence model for Spanish to English translation using the Keras Subclassing API.

from: https://www.tensorflow.org/

Extending the framework to include data and deployment

The `torchvision` package consists of popular datasets, model architectures, and common image transformations for computer vision.

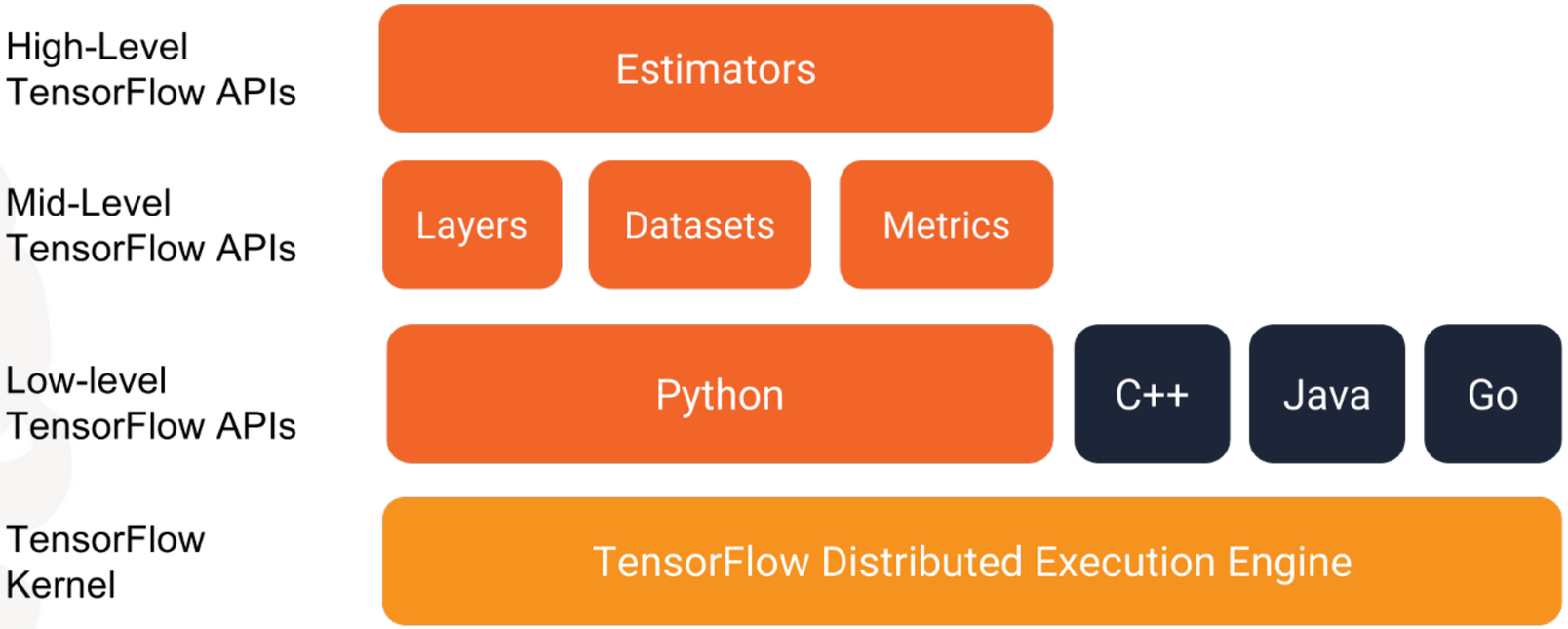
Package Reference

- torchvision.datasets
  - MNIST
  - Fashion-MNIST
  - KMNIST
  - EMNIST
  - QMNIST
  - FakeData
  - COCO
  - LSUN
  - ImageFolder
  - DatasetFolder
  - ImageNet
  - CIFAR
  - STL10
  - SVHN
  - PhotoTour
  - SBU
  - Flickr
  - VOC
  - Cityscapes

https://pytorch.org/docs/stable/torchvision/index.html

Extending the framework to include data and deployment

Multiple modular API layers. Deployment with estimators



https://www.tensorflow.org/get_started/premade_estimators

PART III – Towards Systems

Software Convergence, Limitations & the Future?

AI & ML Software Frameworks

Bosch et al. 2019 – AI Engineering Concept

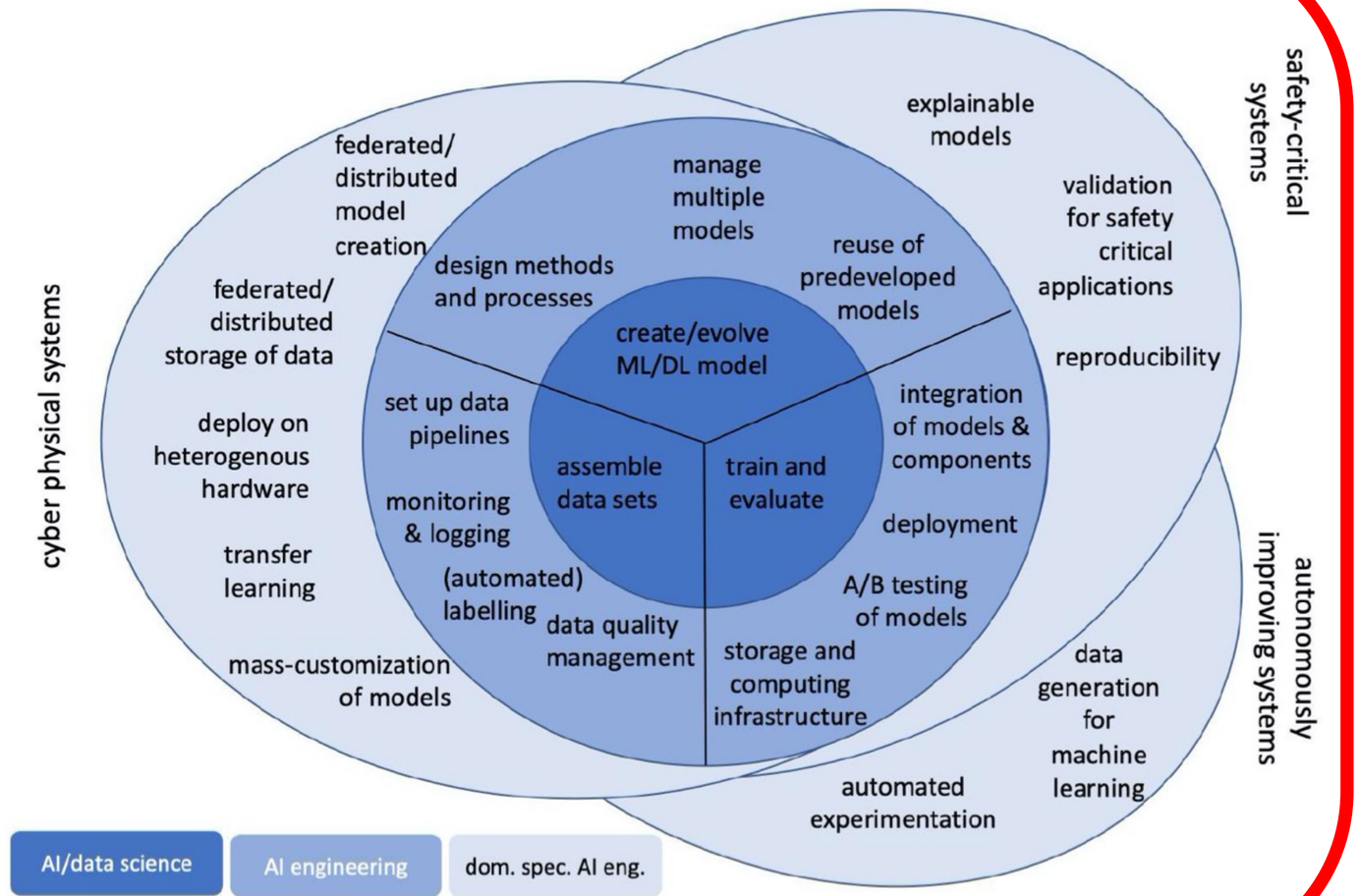


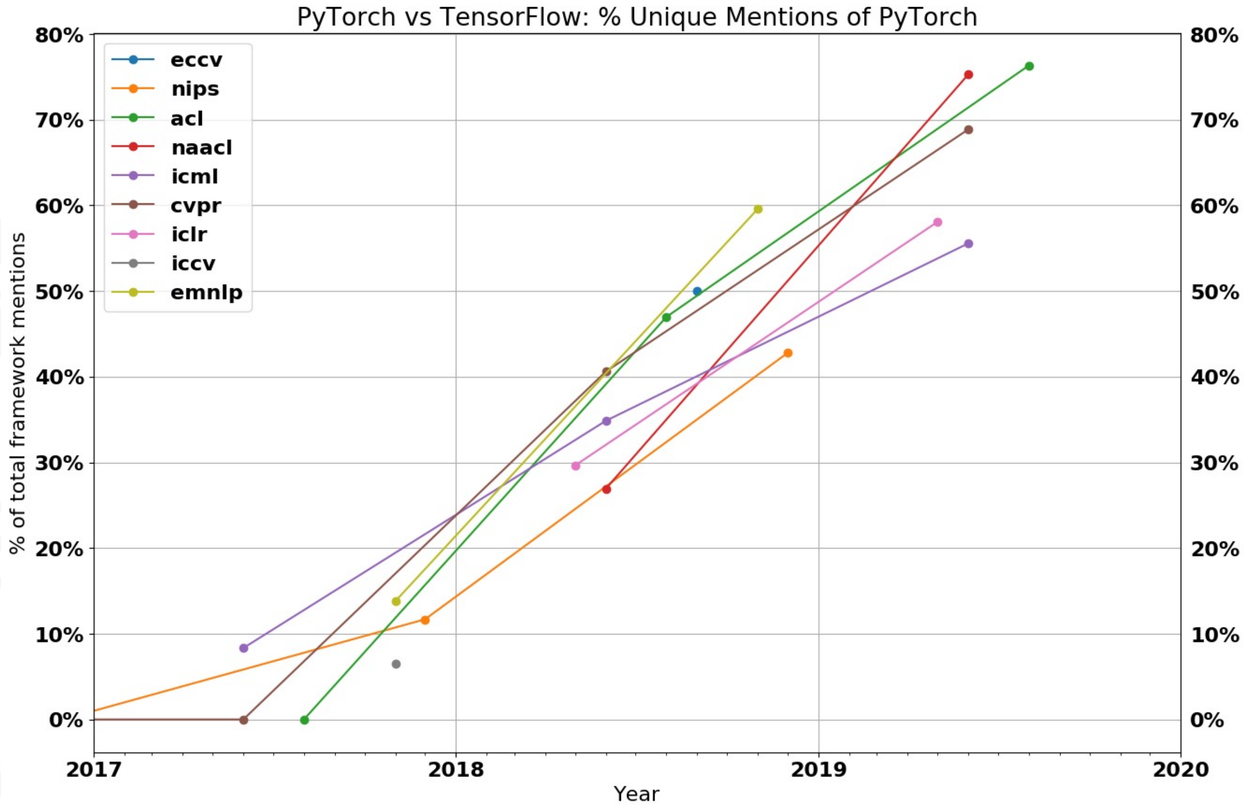
Figure 3: Conceptualization of AI engineering

- The inner to outer circles are reflected in, and to an extent even driven by, the historical development of software tools and corresponding ties to hardware advances.
- The software requirements are analogously constantly reshaped by the general advances in AI engineering, data science and domain specific progress. In addition, requirements commonly shared across all software, such as OS interoperability, ease of installation & use for non-experts etc. apply.

2019 ML framework convergence in academia & industry

Industry → TensorFlow 2.0 (2019)

Academia → PyTorch 1.0 (2018)



Researchers care about how fast they can iterate on their research, which is typically on relatively small datasets (datasets that can fit on one machine) and run on <8 GPUs. This is not typically gated heavily by performance considerations, but by their ability to quickly implement new ideas. On the other hand, industry considers performance to be of the utmost priority. While 10% faster runtime means nothing to a researcher, that could directly translate to millions of savings for a company.

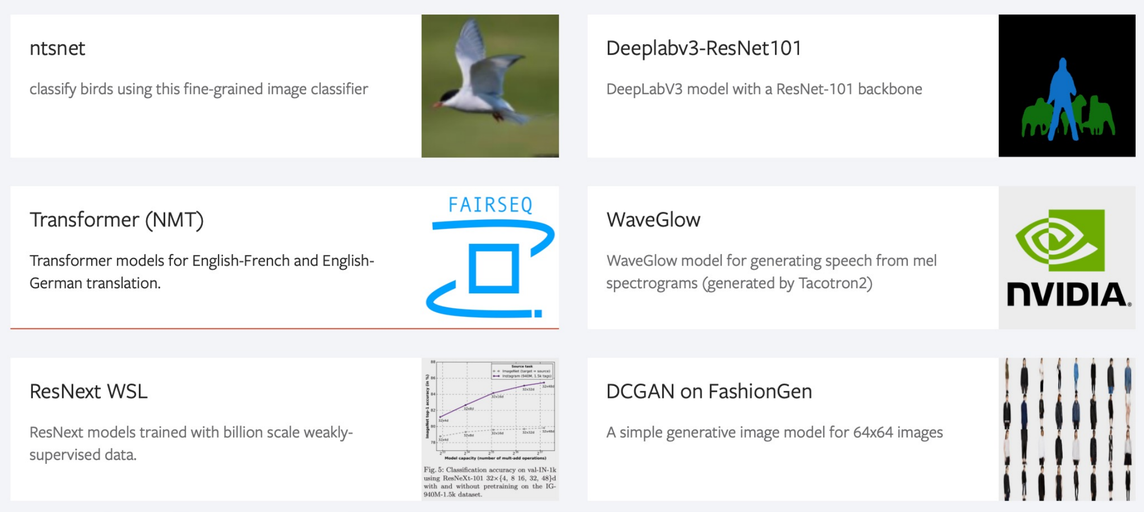
https://thegradient.pub/state-of-ml-frameworks-2019-pytorch-dominates-research-tensorflow-dominates-industry/

2019 ML framework convergence in academia & industry

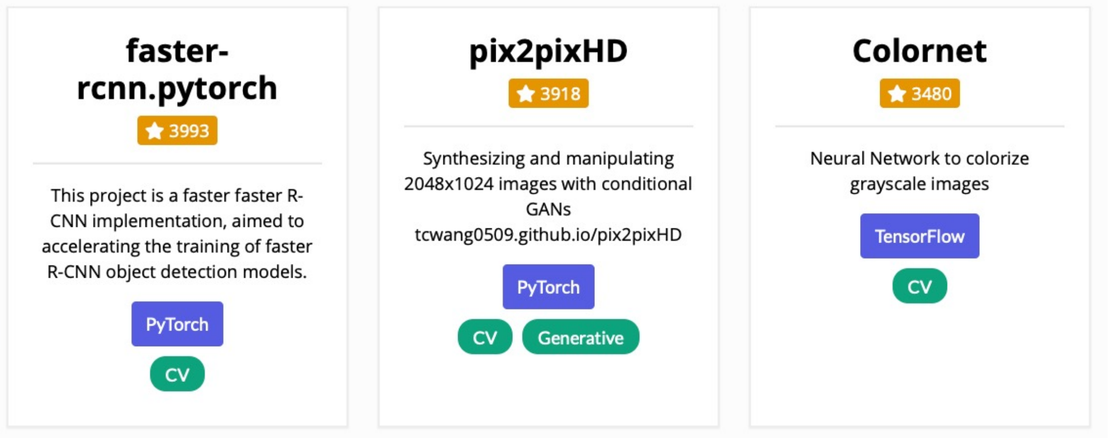
Frameworks keep growing, but are losing uniqueness

- PyTorch 1.0 has introduced a static graph mode to improve deployment
- TensorFlow 2.0 has now defaulted to “eager mode”, i.e. introduced dynamic graphs
- Both frameworks introduce remarkably similar features or even share many feature/backends (like TensorBoard, autodiff, CUDA C code …)
- Many other frameworks such as Torch (v7, 2019), Theano (v1.0, 2019), Chainer (v6.3, 2019), Microsoft CNTK (2.7, 2019) have officially announced their last release or have been swallowed, e.g. Caffee into PyTorch, Keras into TensorFlow by the two main contenders.

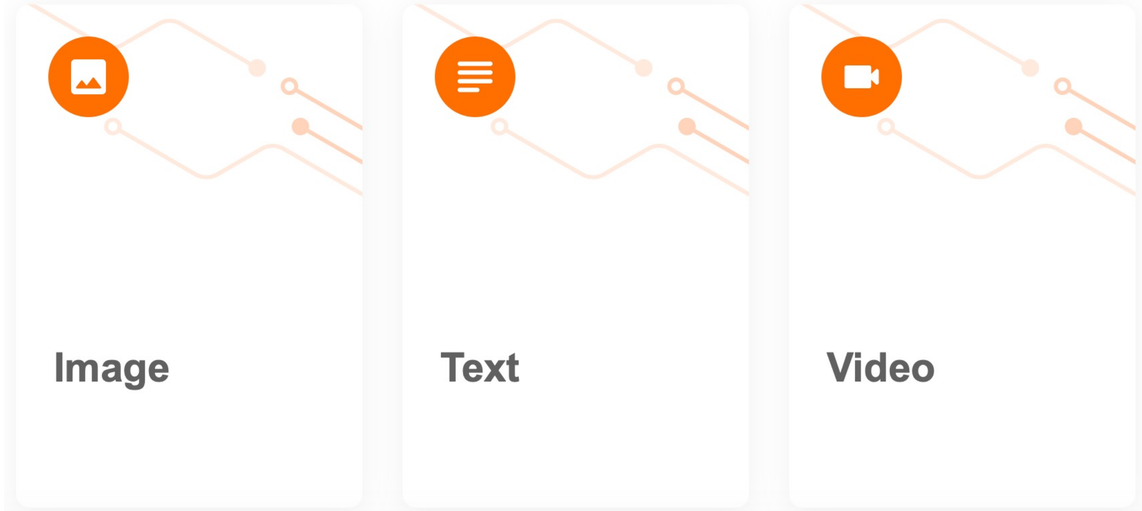
Model sharing, transfer learning, reproducibility, applications



https://pytorch.org/hub/



https://modelzoo.co



https://tfhub.dev

TensorFlow Model Garden

Model sharing, transfer learning, reproducibility, applications

- ONNX – the open neural network exchange ecosystem and effort has been launched to introduce an open format for machine learning models. This essentially allows for interoperability between programming languages, software frameworks and allows to better leverage certain hardware optimizations present in select frameworks.

https://onnx.ai

Model sharing, transfer learning, reproducibility, applications

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Frameworks & Converters

Use the frameworks you already know and love.

Caffe2 · Chainer · dmlc XGBoost · K · LibSVM

MATLAB · Cognitive Toolkit · mxnet · ML · 🐱

Neural Network Libraries · PaddlePaddle · PyTorch · sas · SIEMENS

SINGA · learn · Tengine · TensorFlow

Visualize

Better understand your model by visualizing its computational graph.

NETRON · Visual DL

Cloud Services

Leverage services from cloud providers to build, train and inference your models.

Inference

Deploy your ONNX model using runtimes designed to accelerate inferencing.

BITMAIN · CEVA · habana · intel AI · NVIDIA

ONNX RUNTIME · MACE Mobile AI Compute Engine · Qualcomm · Rockchip · skymizer

SYNOPSYS · Tencent · tvm · TwinCAT 3 · vespa

Windows

https://onnx.ai/supported-tools.html

Heterogeneous hardware

- PyTorch and TensorFlow have introduced mobile support, support for TPUs (tensor processing units) and have adapted their code towards device agnostic programming by introducing generic commands such as “.to(device)”

- TensorFlow has introduced native swift, javascript versions and a set of browser based applications

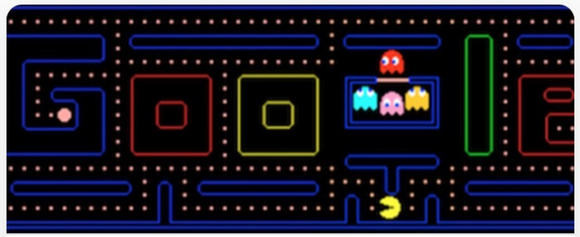
Demos

See examples and live demos built with TensorFlow.js.

Emoji Scavenger Hunt

Use your phone’s camera to identify emojis in the real world. Can you find all the emojis before time expires?

Explore demo ↗ View code



Webcam Controller

Play Pac-Man using images trained in your browser.

Explore demo ↗ View code

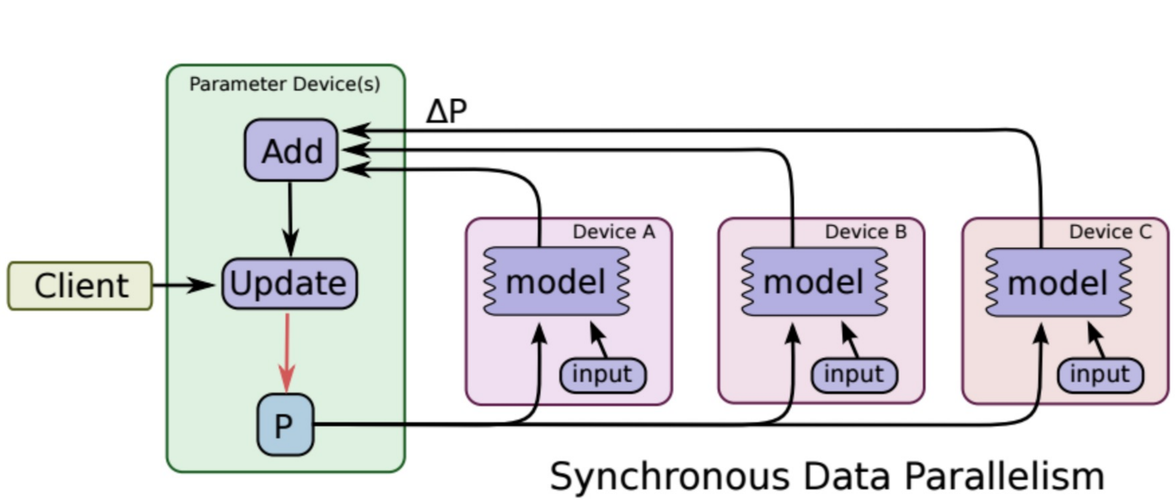
Teachable Machine

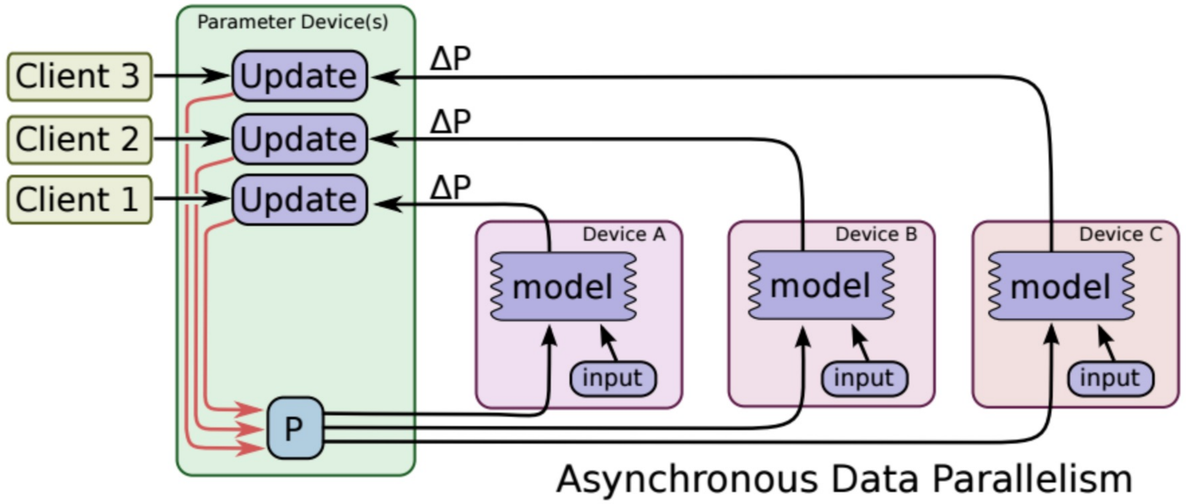
No coding required! Teach a machine to recognize images and play sounds.

Explore demo ↗ View code

Scaling not only horizontally, but vertically

Beyond data parallelism, towards model parallelism, distributed asynchronous computing, HPC.



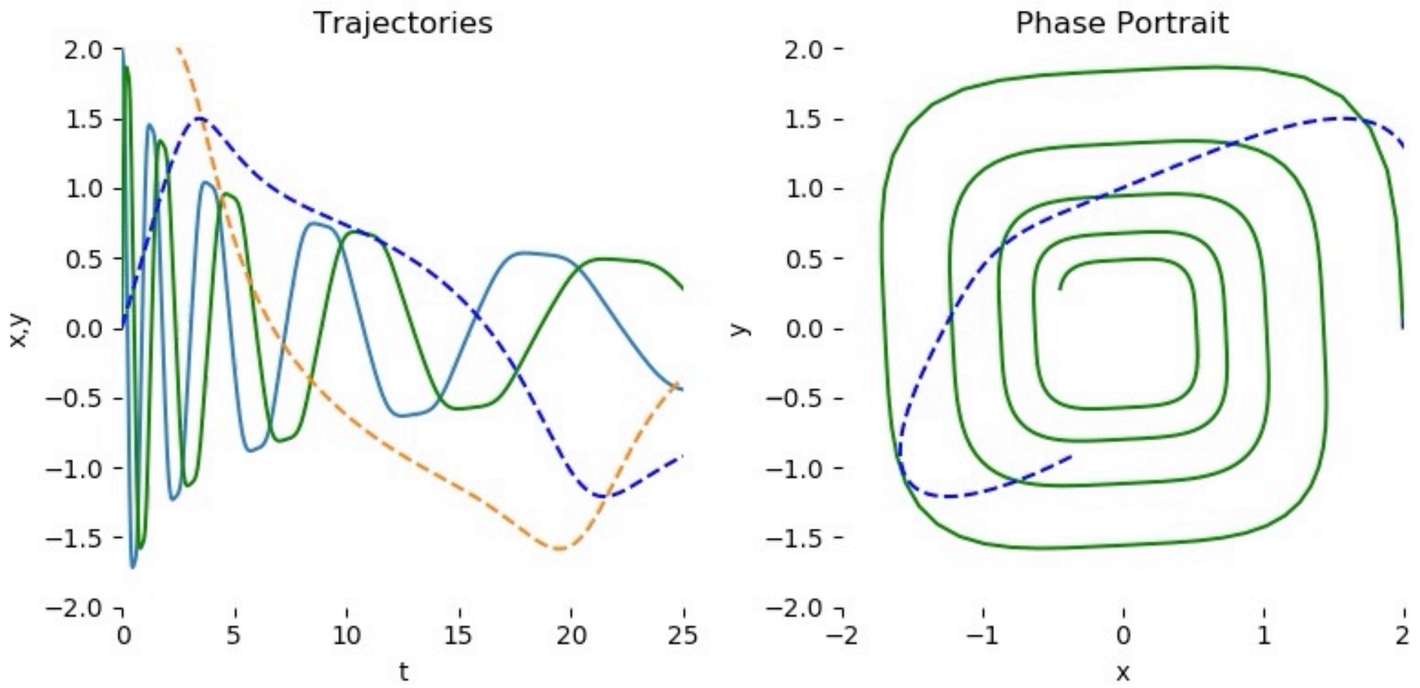


https://github.com/tensorflow/models/tree/master/research/inception

TensorFlow and PyTorch introduced native support for distributed applications, other frameworks such as Horovod provide additional layers on top

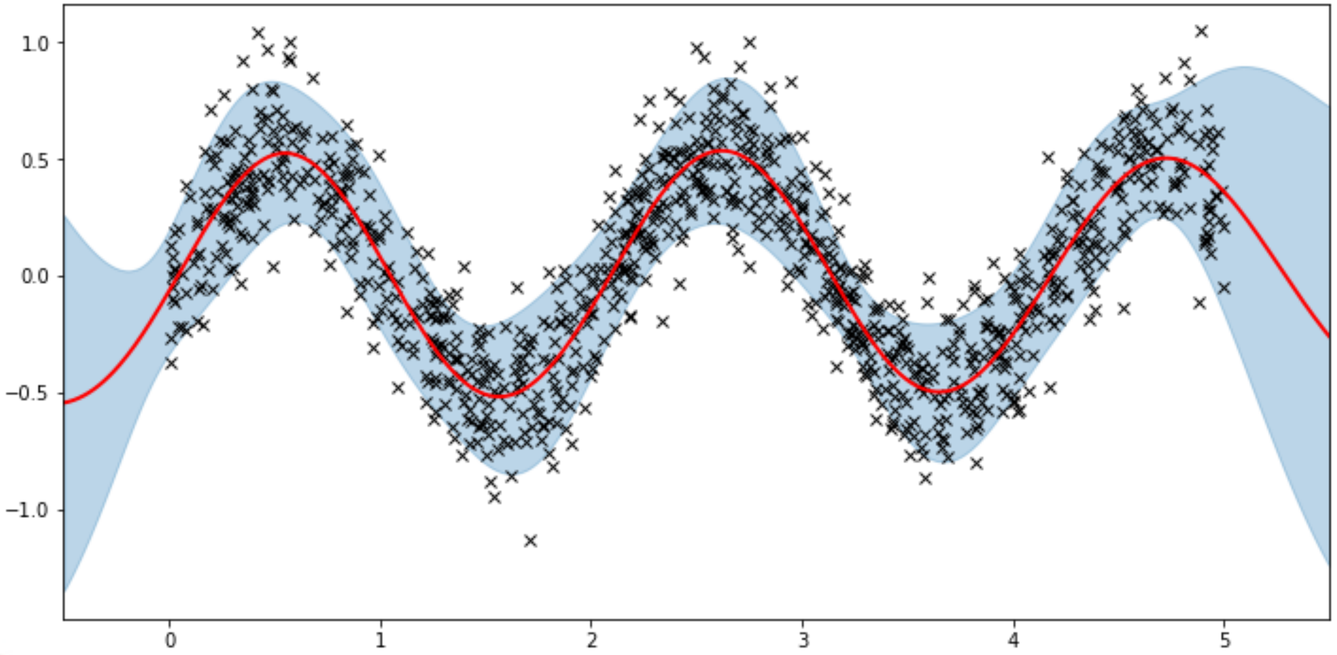
Beyond black-box models

- A variety of visualization toolboxes to gain introspection into features, activations and other visual explorations. See e.g. https://distill.pub for a range of interactive tools.
- Software is generic enough for projects to emerge beyond common black box applications, e.g. differentiable ODE solvers (Chen et al. 2018)



Beyond black-box models

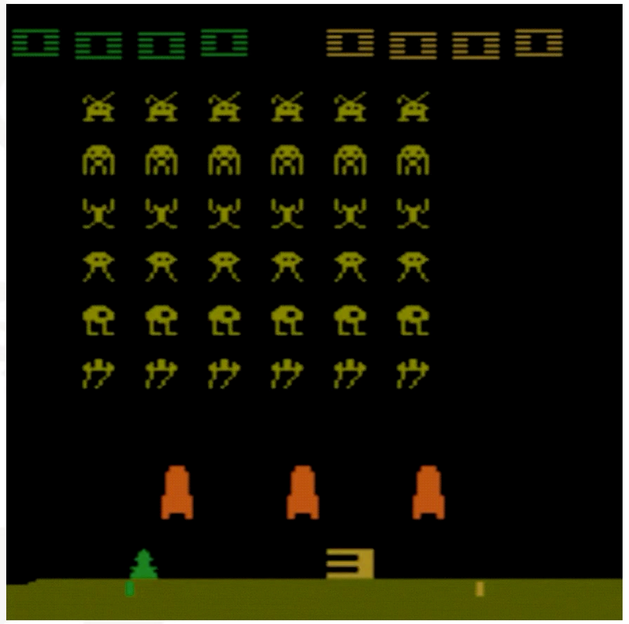
- Probabilistic programming languages towards explainable machine learning models: Edward and Pyro software on top of TensorFlow and PyTorch.
- Includes a wide range of techniques and methods such as tools for Bayesian optimization, graphical models, Gaussian processes, deep latent variable models, Dirichlet processes, hidden Markov models, sampling techniques …

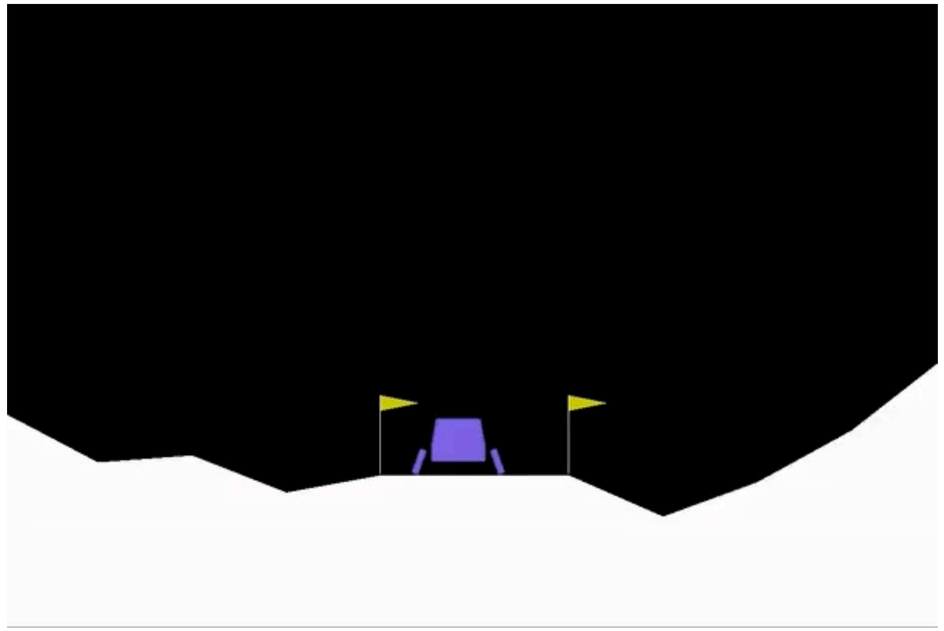


https://pyro.ai/examples/gp.html

Learning environments & graphics simulation

- Software such as OpenAI Gym now provides seamless integration of ATARI games or physics-based environments such as a pendulum with ML software frameworks like TensorFlow and PyTorch. Reinforcement learning, evolutionary algorithms or other approaches can be trained and tested in these environments.





https://gym.openai.com

Learning environments & graphics simulation

- There is many machine learning projects that make use of 3-D simulation using Unreal Engine, Unity, Blender etc. but the majority of these still treats the graphics pipeline as a tool to “dump data”.
- A full software stack that integrates the (graphics) simulation pipeline for data generation, model training and particularly validation remains open.

http://animalaiolympics.com/AAI/

AI & ML Software Frameworks

Bosch et al. 2019 – AI Engineering Concept

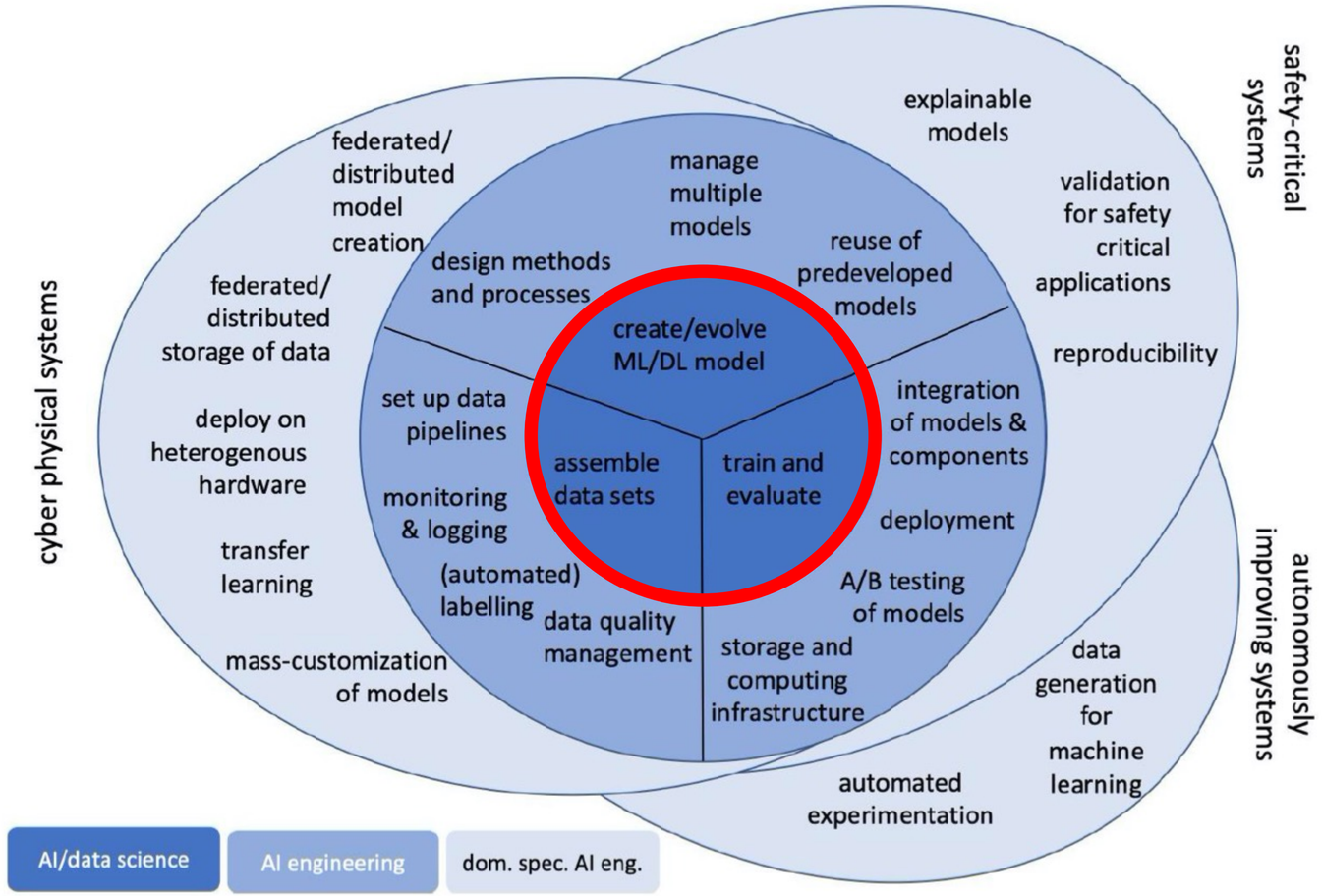


Figure 3: Conceptualization of AI engineering

What if we want or need to revisit the center?

Are our frameworks converging too much?

Limitations induced through current ML practices & its software

Machine Learning Systems are Stuck in a Rut, Workshop on Hot Topics in Operating Systems, Barham and Isard 2019, https://dl.acm.org/doi/10.1145/3317550.3321441

“Despite impressive and sometimes heroic efforts on some of the sub-problems, we as a community should recognize that we aren’t doing a great job of tackling the end-to-end problem in an integrated way. “

“We do not want to minimize the thought and engineering that has gone into current machine learning tool chains, and clearly they are valuable to many. Our main concern is that the inflexibility of languages and back ends is a real brake on innovative research, that risks slowing progress in this very active field”

Limitations induced through current ML practices & its software

Machine Learning Systems are Stuck in a Rut, Workshop on Hot Topics in Operating Systems, Barham and Isard 2019, https://dl.acm.org/doi/10.1145/3317550.3321441

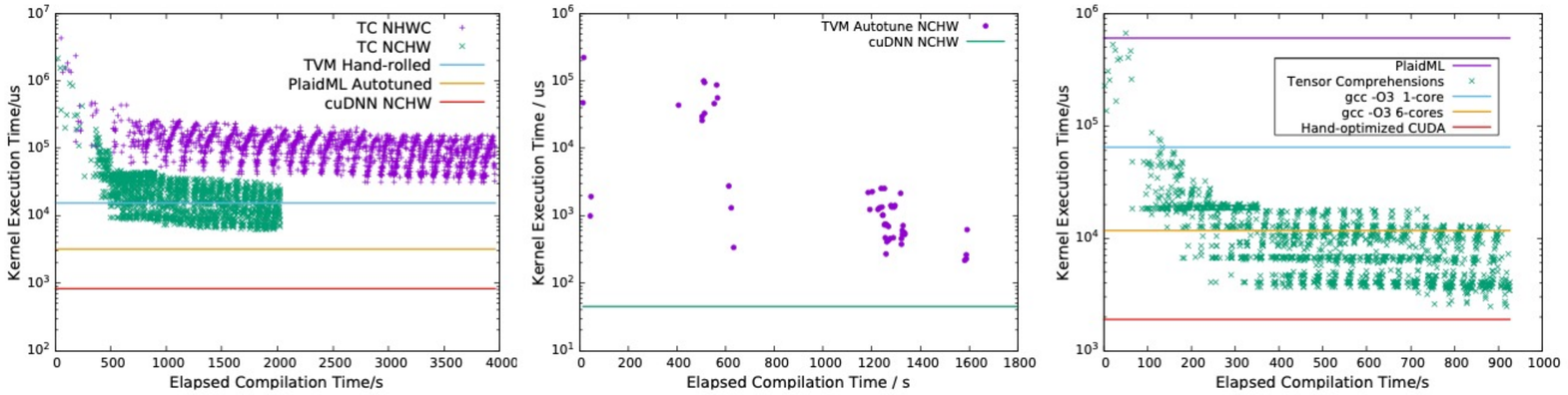
There is recent work typically referred to as “capsule networks” (Sabour et al. 2017) that tries to address some open issues with current neural networks, such as needing to see a variety of data from different viewpoints, illumination etc. in order to built up some sort of invariance.

At this point it is not important to understand what the internals of a capsule are or how the model works. For us it is enough to know that it might be an advance over existing models and that the basic computation happening in capsules is very similar to that of a convolution in a conventional CNN as we have seen on the slides before.

Limitations induced through current ML practices & its software

Machine Learning Systems are Stuck in a Rut, Workshop on Hot Topics in Operating Systems, Barham and Isard 2019, https://dl.acm.org/doi/10.1145/3317550.3321441

- It is not trivial to optimize operations over multiple dimensions and there is a hardware preference for specific memory layouts such as batch-channel-width-height (BCWH). This is counterintuitive & non-optimal, especially since operations such as e.g. convolutions are polymorphic over the rank (number of dimensions) -> what if we want to use more dimensions



(a) Strided conv2d (batch=32) **(b)** TVM autotuned conv2d (batch=1) **(c)** Capsule Kernel

Figure 3. Performance comparison of autotuned kernels

Compiler	Device	Compilation	Execution
gcc	x86 (1 core)	500ms	64.3ms
gcc -fopenmp	x86 (6 cores)	500ms	11.7ms
PlaidML	GTX1080	560ms	604ms
Tensor Comp.	GTX1080	3.2s	225ms
Tensor Comp.	GTX1080	64s	18.3ms
Tensor Comp.	GTX1080	1002s	1.8ms
CUDA	GTX1080	48h	1.9ms

Table 1. Convolutional Capsules Microbenchmark

Limitations induced through current ML practices & its software

Machine Learning Systems are Stuck in a Rut, Workshop on Hot Topics in Operating Systems, Barham and Isard 2019, https://dl.acm.org/doi/10.1145/3317550.3321441

- Code optimization happens at function and not system level, e.g. individual convolutions being the subject. The end-to-end pipeline is not considered.

- We are relying on the same old backend: *"There are frameworks, such as Julia, which nominally use the same language to represent both the graph of operators and their implementations, but back-end designs can diminish the effectiveness of such a front end. In Julia, while 2D convolution is provided as a native Julia library, there is an overloaded conv2d function for GPU inputs which calls NVidia's cuDNN kernel . Bypassing this custom implementation in favor of the generic code essentially hits a "not implemented" case and falls back to a path that is many orders of magnitude slower."*

“It is perhaps under appreciated how much machine learning frameworks shape ML research. They don’t just enable machine learning research. They enable and restrict the ideas that researchers are able to easily explore. How many nascent ideas are crushed simply because there is no easy way to express them in a framework?“

https://thegradient.pub/state-of-ml-frameworks-2019-pytorch-dominates-research-tensorflow-dominates-industry/