Open World Lifelong Learning
A Continual Machine Learning Course

Teacher

Dr. Martin Mundt,

hessian.AI-DEPTH junior research group leader on Open World Lifelong Learning (OWLL)
& researcher in the Artificial Intelligence and Machine Learning (AIML) group at TU Darmstadt

Time

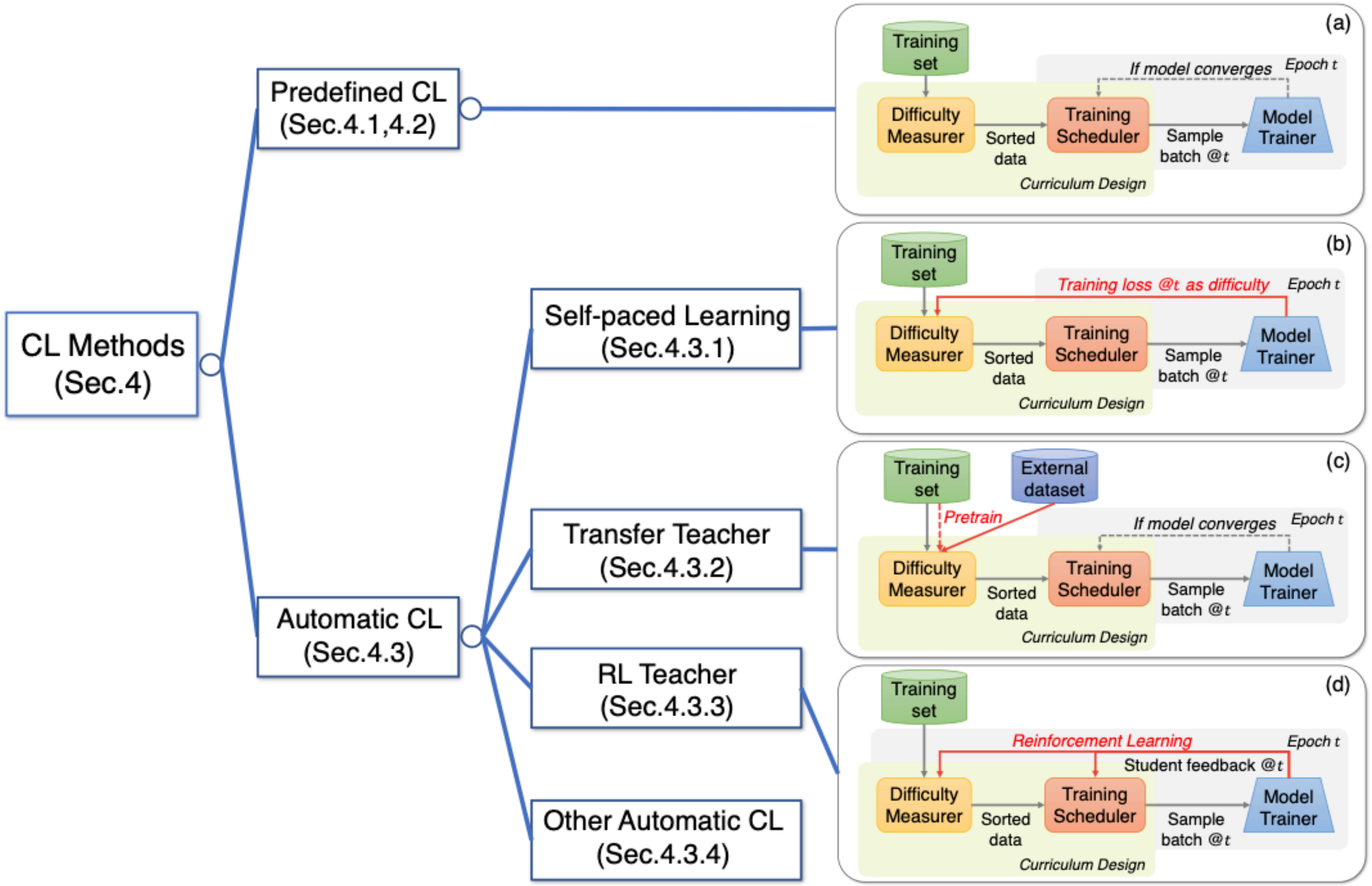
Every Tuesday 17:30 - 19:00 CEST

Course Homepage

http://owll-lab.com/teaching/cl_lecture

https://www.youtube.com/playlist?list=PLm6QXeaB-XkA5-lVBB-h7XeYzFzgSh6sk

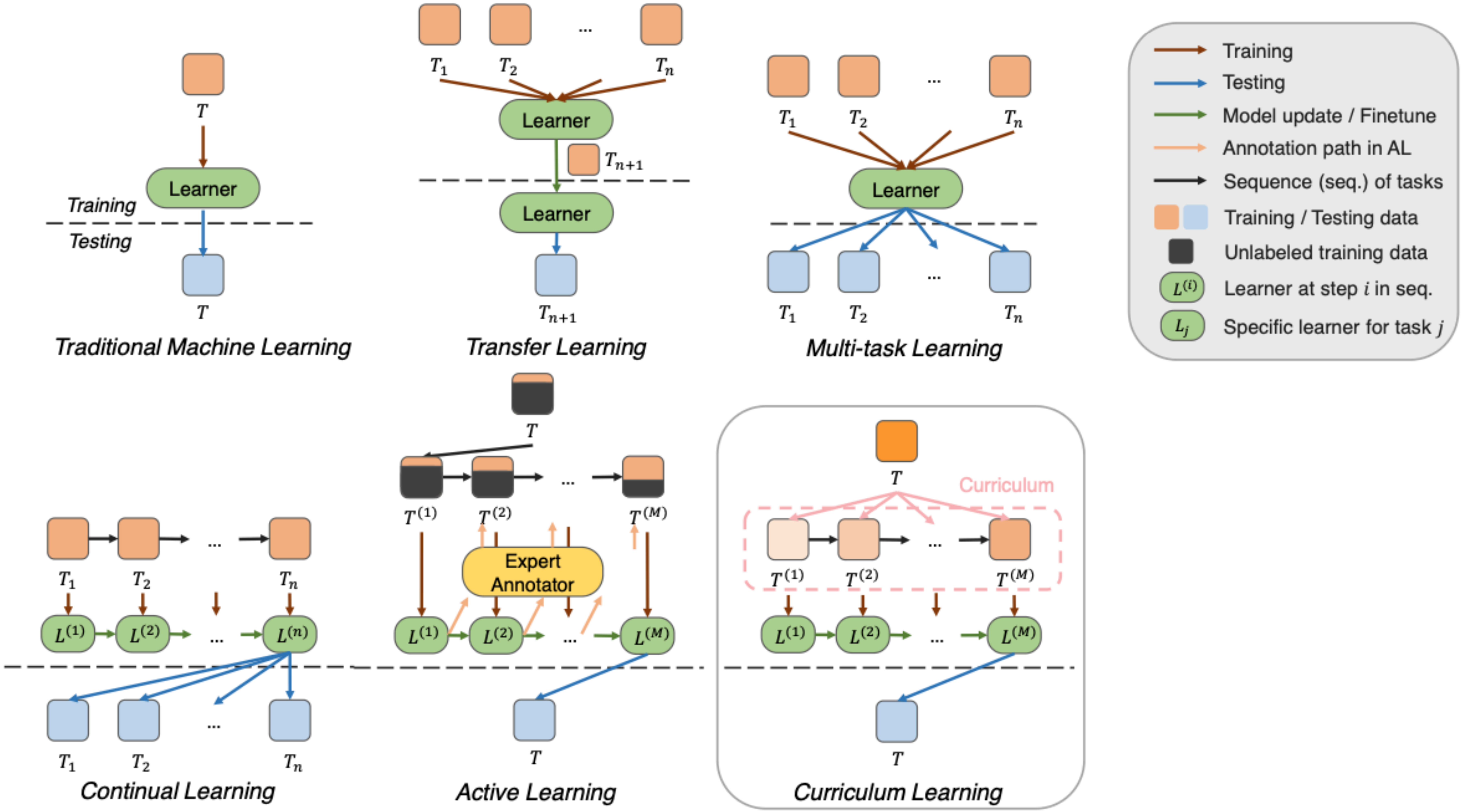
Recall: curriculum learning



Wang et al, “A Survey on Curriculum Learning”, TPAMI 2021

What we talked about last week

Recall: various paradigms



What we've covered in the course (+more like open world learning)

Wang et al, "A Survey on Curriculum Learning", TPAMI 2021

Week 10: The influence and role of soft + hardware

“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/

AI & ML Software Frameworks

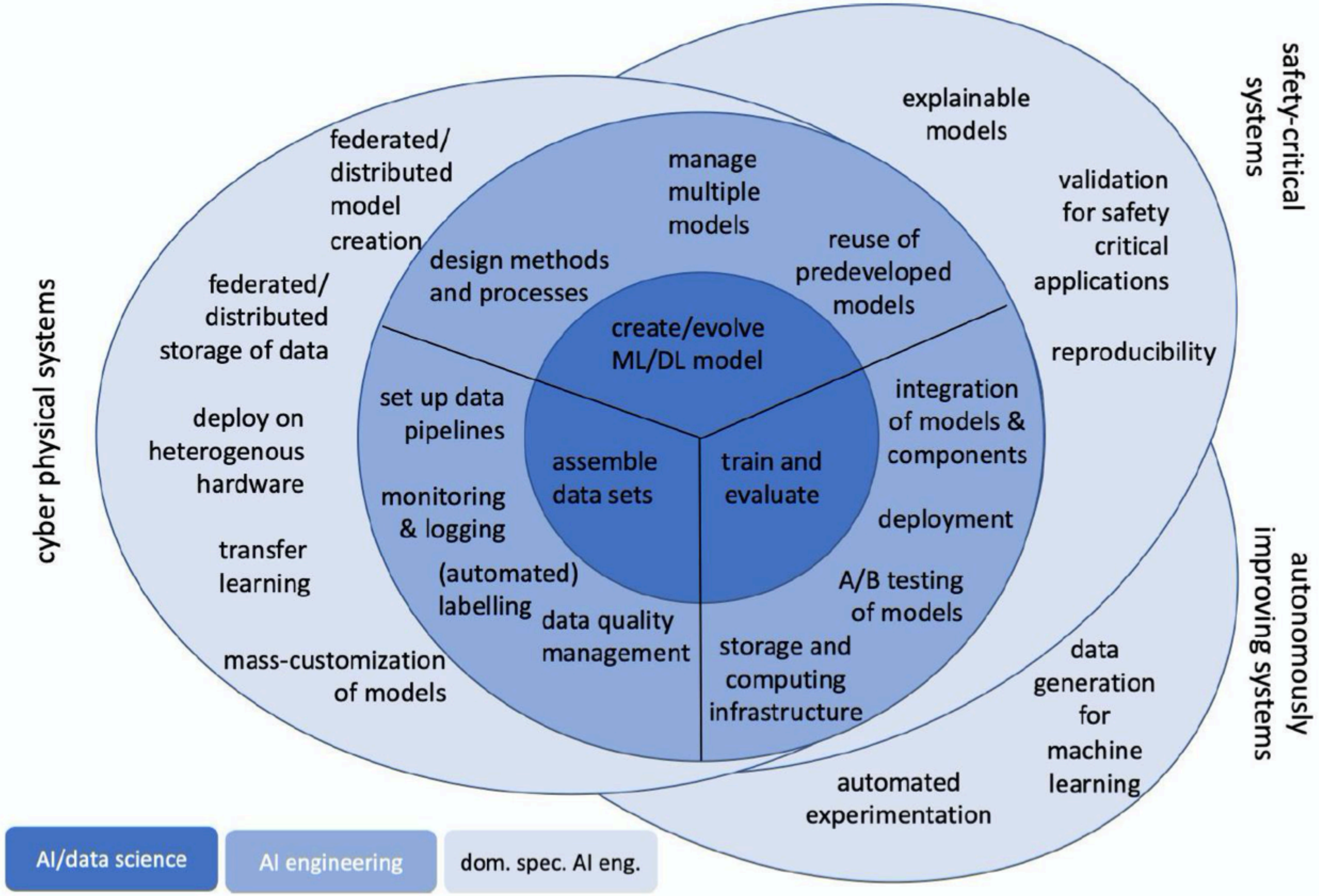


Figure 3: Conceptualization of AI engineering

Bosch et al, "Engineering AI Systems: A Research Agenda", in Artificial Intelligence Paradigms for Smart Cyber-Physical Systems

Inner to outer circles are reflected in/ driven by development of software tools & hardware advances

Software requirements are constantly being reshaped

AI & ML Software Frameworks

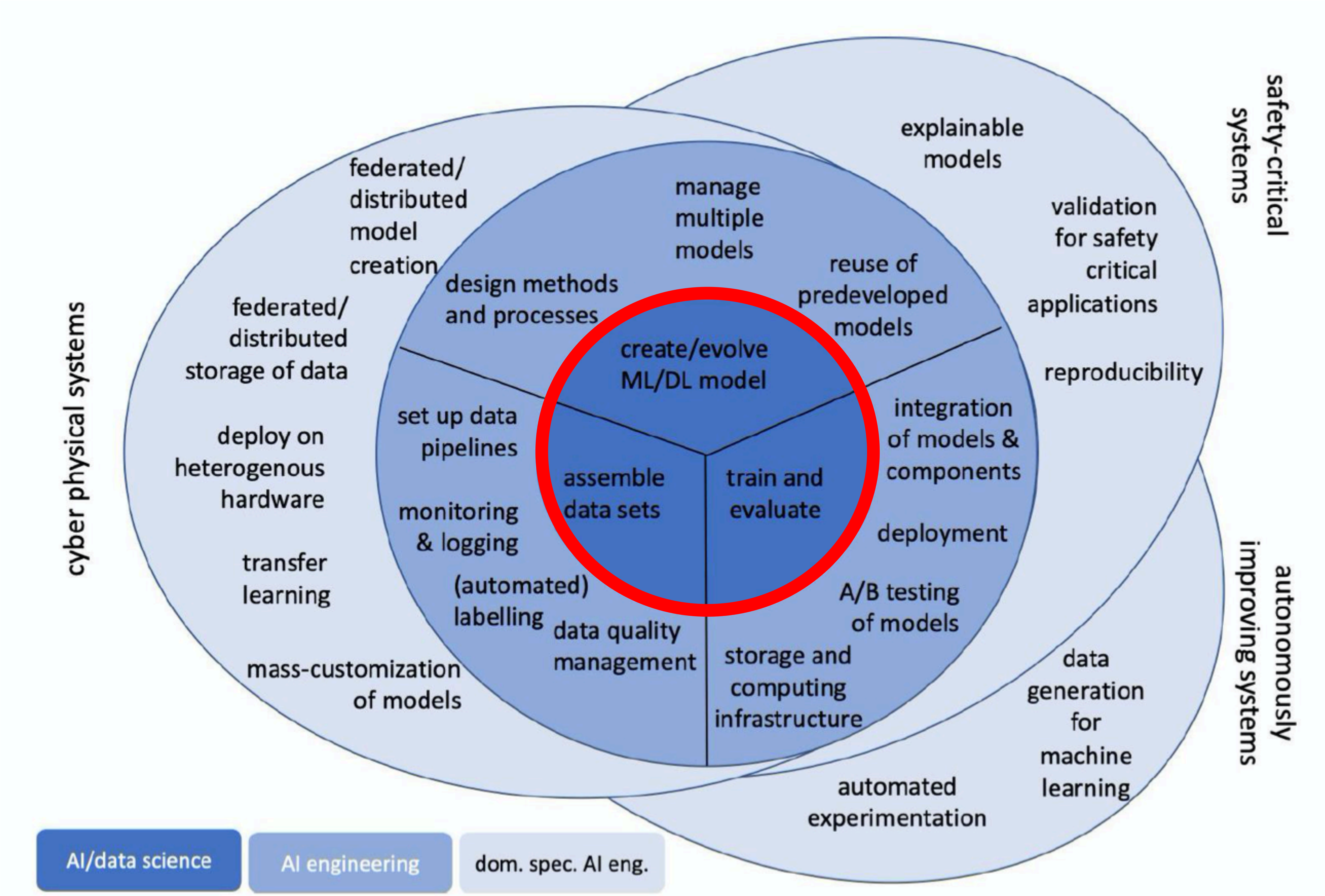


Figure 3: Conceptualization of AI engineering

Bosch et al, "Engineering AI Systems: A Research Agenda", in Artificial Intelligence Paradigms for Smart Cyber-Physical Systems

Inner to outer circles are reflected in/ driven by development of software tools & hardware advances

Software requirements are constantly being reshaped

Well-known long-term ideas

Some key examples:

1. **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)

2. **Numerical optimization** in natural sciences & algorithmic techniques at the heart of machine learning: **expectation maximization** (Dempster 1977) or **backpropagation** (Werbos 1983, Rummelhart 1986)

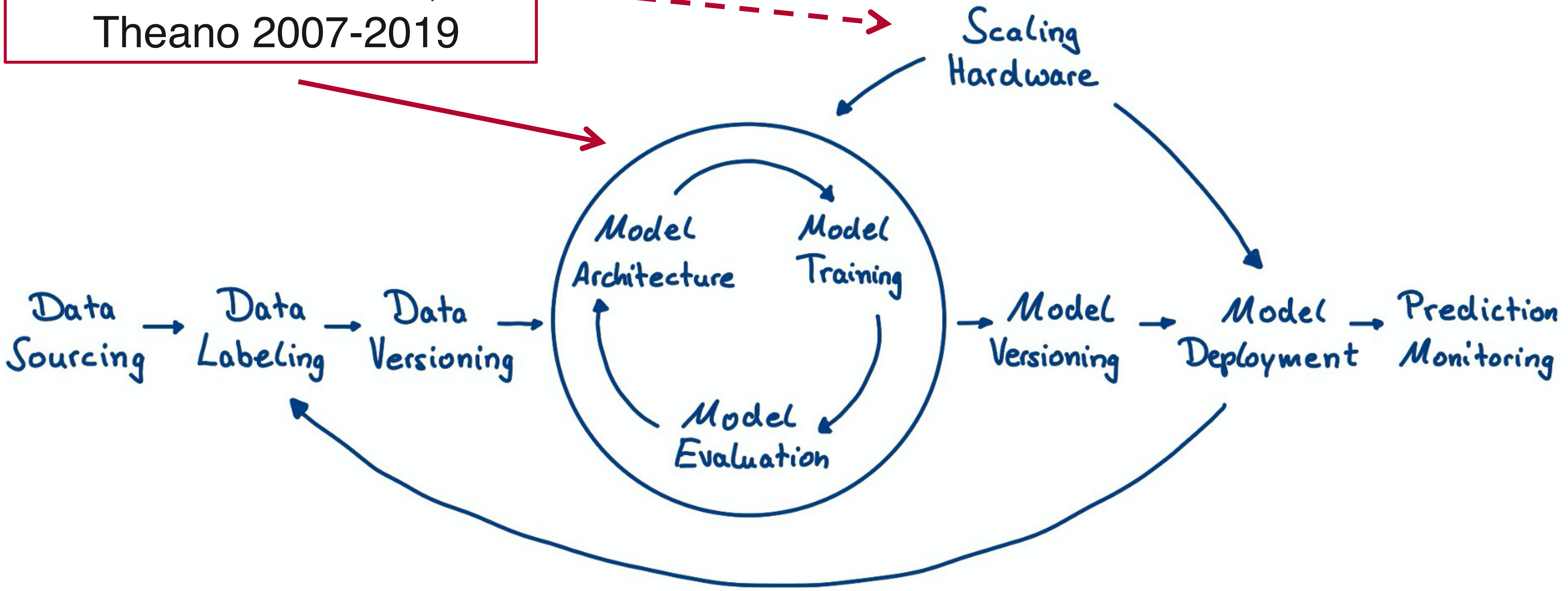
3. Specific models such as **neural networks** (Rosenblatt 1961, Fukushima 1979), **decision trees & random forests** date back at least as much, if not even further.

What are the enablers for the current wave?

[?] Availability of data, computational power, hardware, software tools.

Torch, Theano and the "core"

Torch 2002-2019,
Theano 2007-2019



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

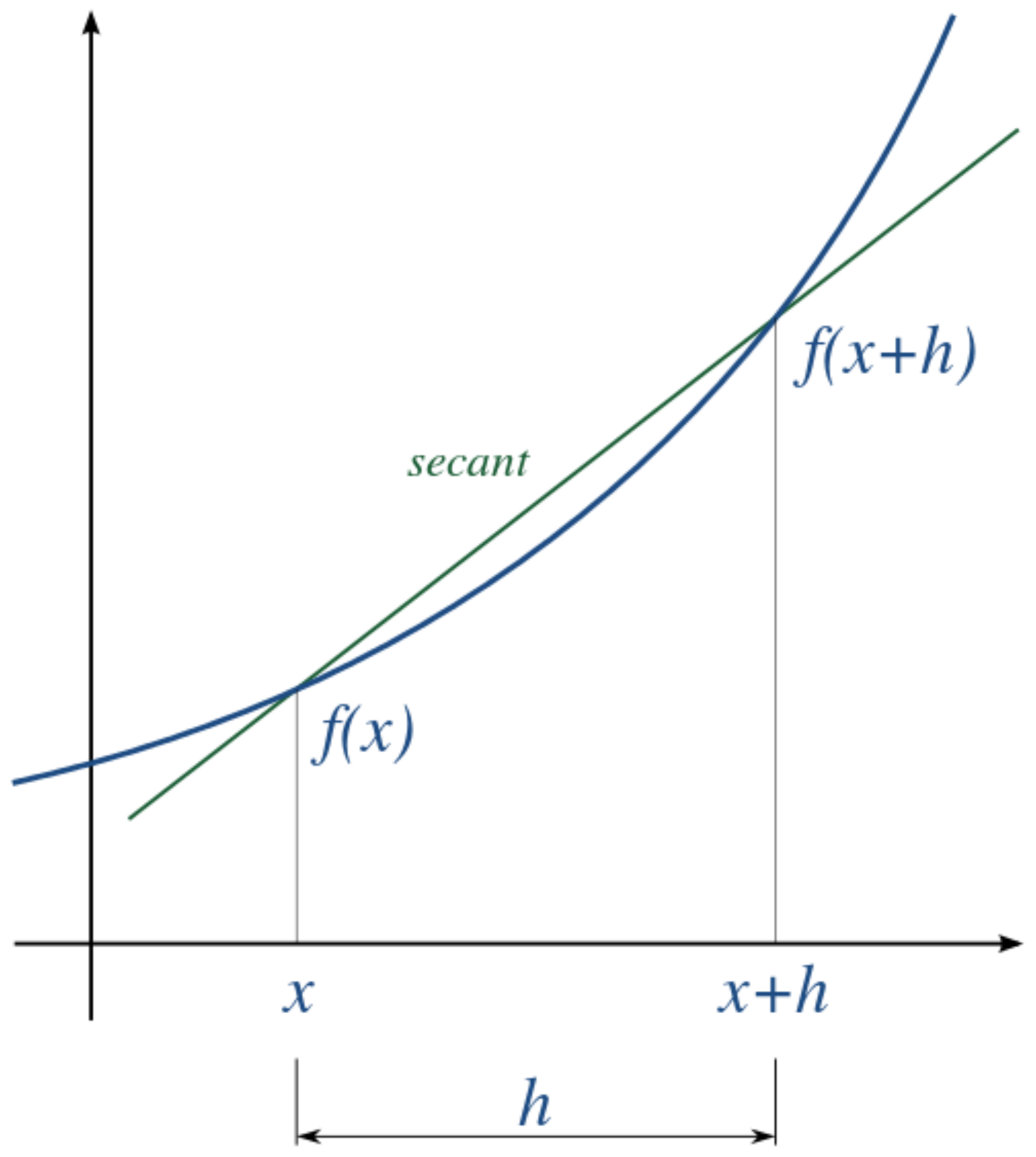
Torch, Theano and the “core”

- **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…

- Build on **strong matrix computation backend** (in C), 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: finite differences



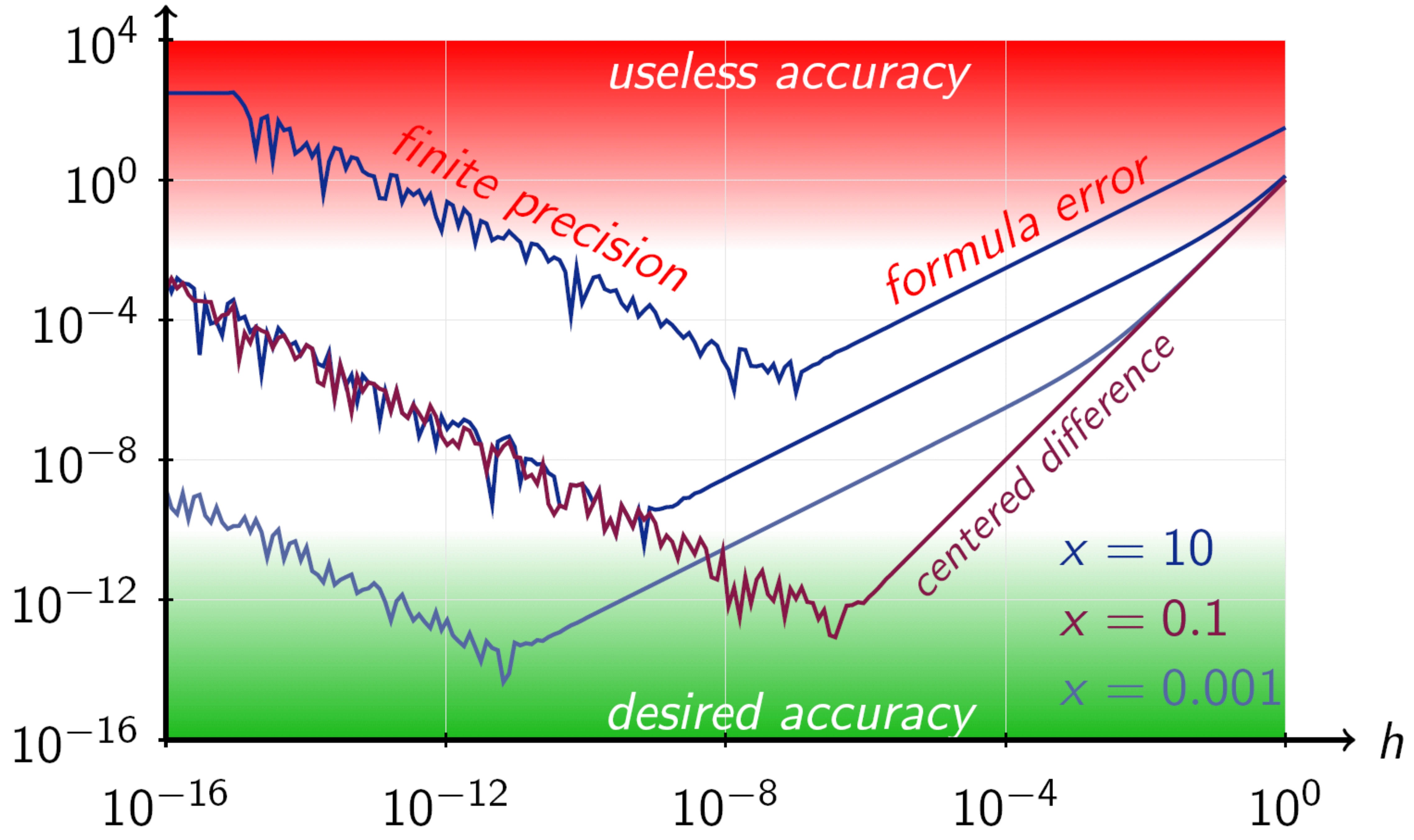
$$\frac{f(x+h)-f(x)}{h}$$

Numerical optimization:

Pick two points and compute slope of nearby secant line through points [x,f(x)] and [x+h, f(x+h)]

The derivative of *f* at *x* is the limit of the value of the difference quotient as the secant lines get closer to being a tangent

Differentiation: finite differences



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

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

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

Numerical optimization:

- *If h is too small*: subtraction yields large rounding error
- *If h is too large*: estimate of the secant becomes more accurate, but estimate for slope of the tangent gets worse

Symbolic programming: Theano

And beyond pure numerics: **symbolic programming** & automatic differentiation

```
>>> 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

Symbolic programming: Theano

```
>>> 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)
```

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)'
```

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

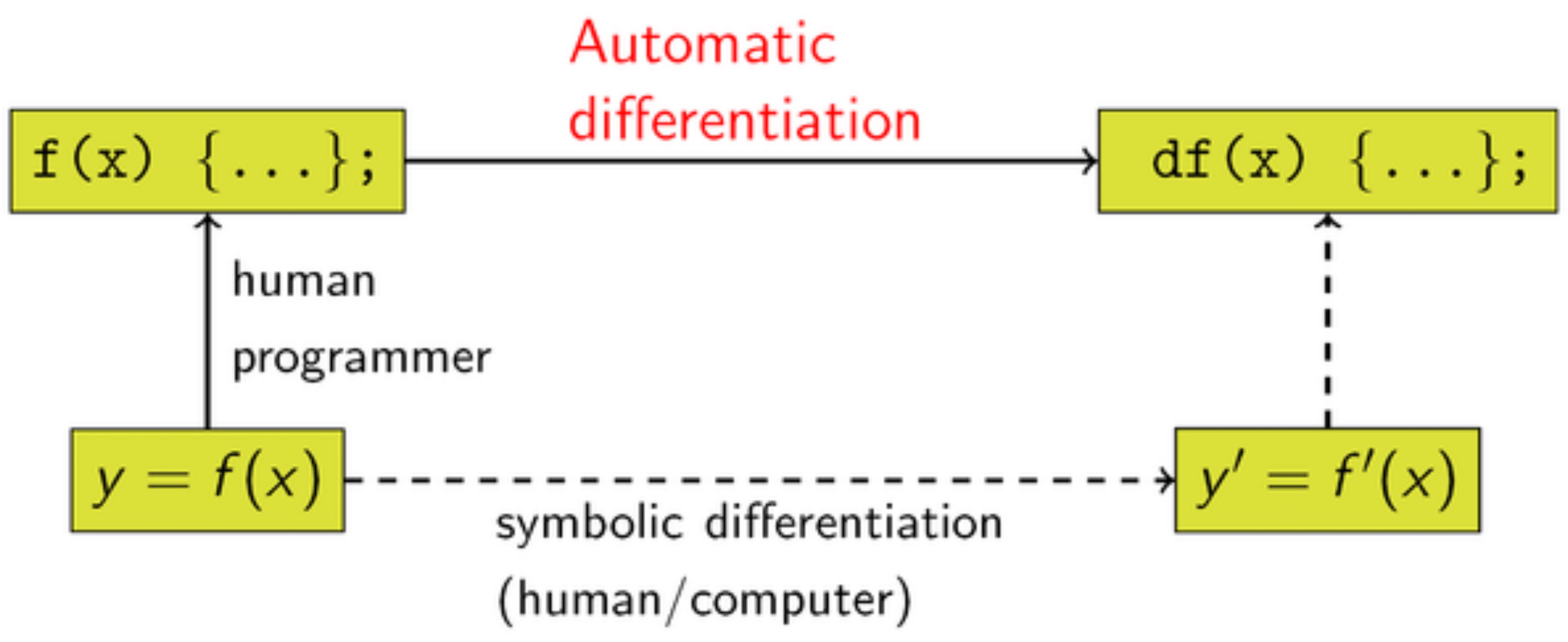
Building blocks + autodiff

The key idea of **automatic differentiation** is called "forward" &"reverse mode" accumulation.

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.

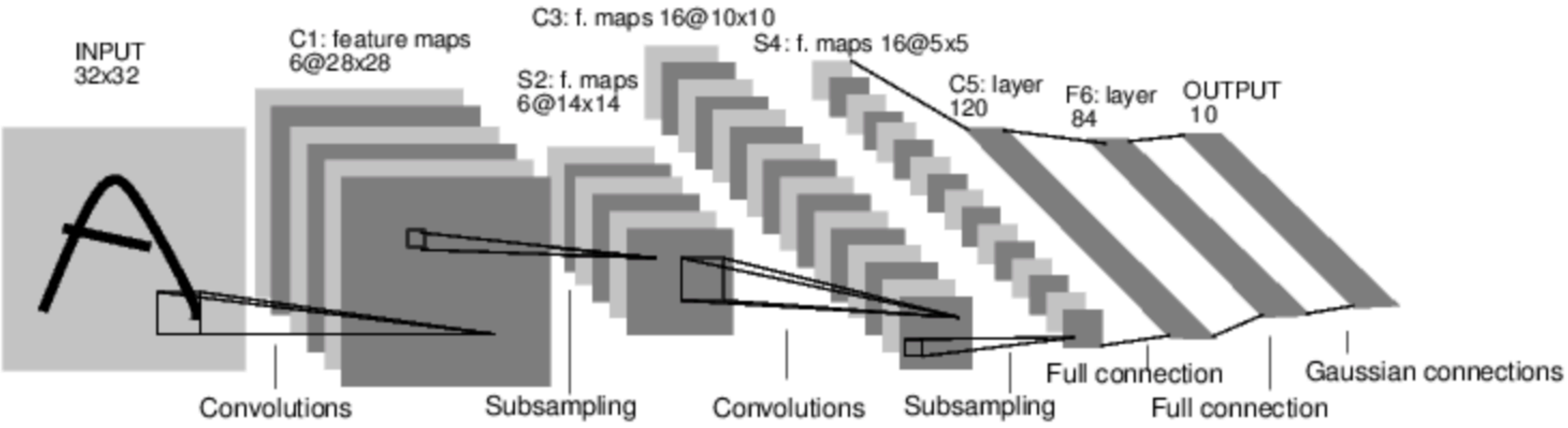
Automatic differentiation tracks operations & makes use of the **chain rule of differentiation**.

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



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

Introducing model building blocks



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)

Building blocks + autodiff

```
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

Building blocks + autodiff

Why is this special? Putting both together

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)
```

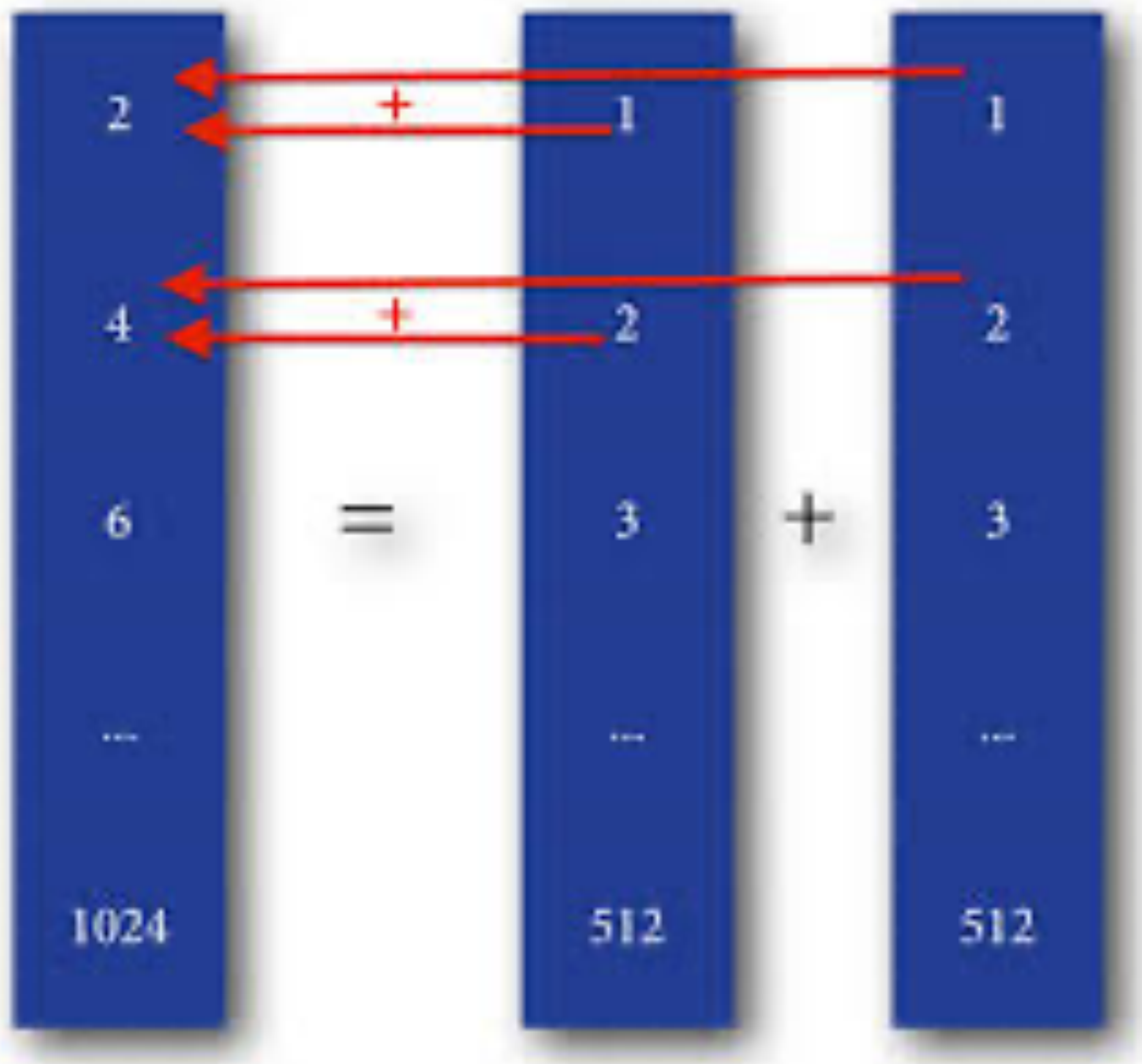
https://github.com/soumith/cvpr2015/blob/master/Deep%20Learning%20with%20Torch.ipynb

Importance of hardware: GPUs

Let's do a small tour de force in **parallelization** to fully appreciate the presented software frameworks.

Think of **vector addition** or the respective **Hadamard product**.

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: GPUs

```
#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;
}
```

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

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

Here is an example as a refresher.

Importance of hardware: GPUs

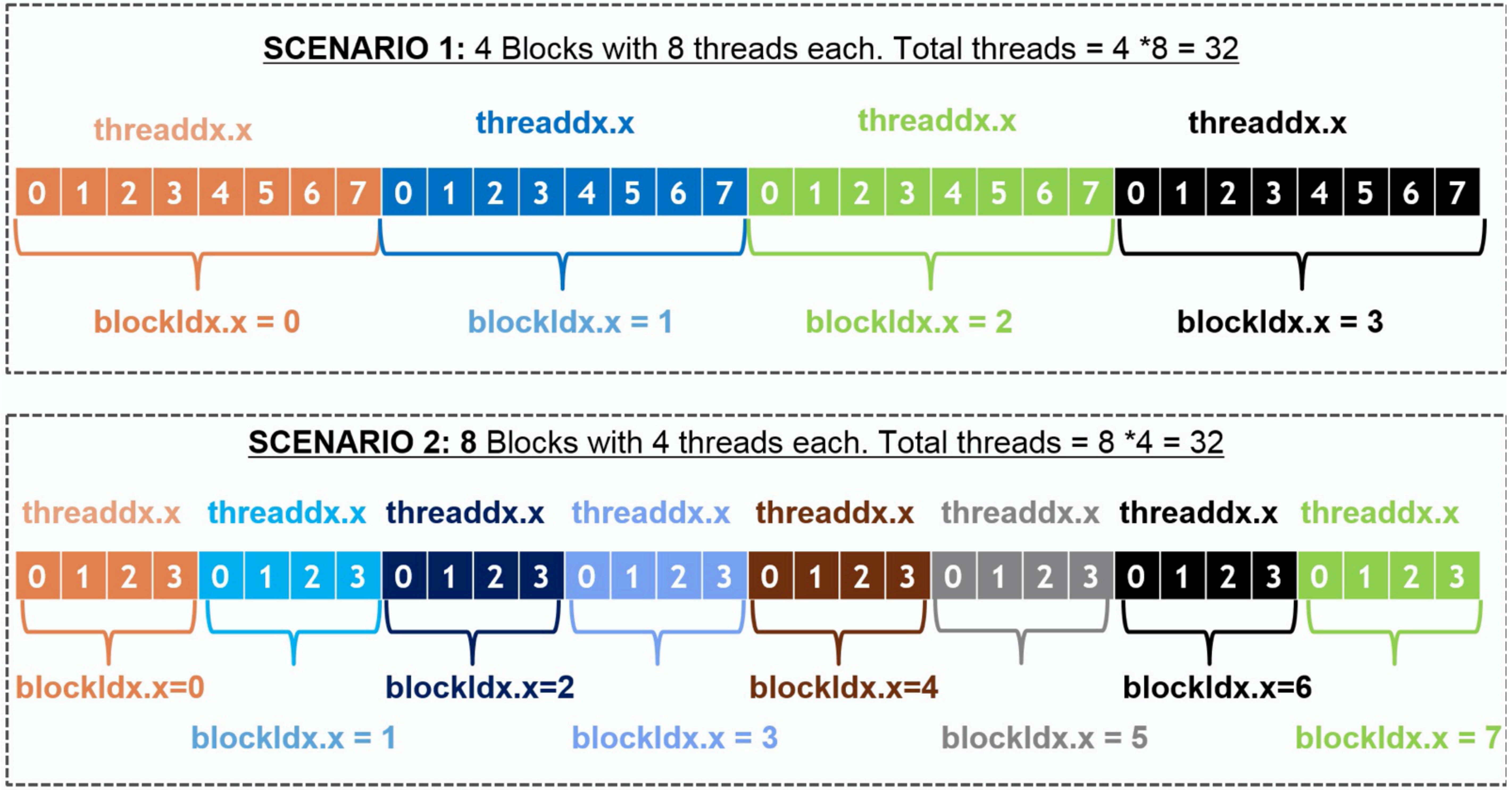
```c
int main(void) {
    int *a, *b, *c;
    int *d_a, *d_b, *d_c; // device copies of a, b, c
    int size = N * sizeof(int);

    // 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;
}
```

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

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

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

Importance of hardware: GPUs



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

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

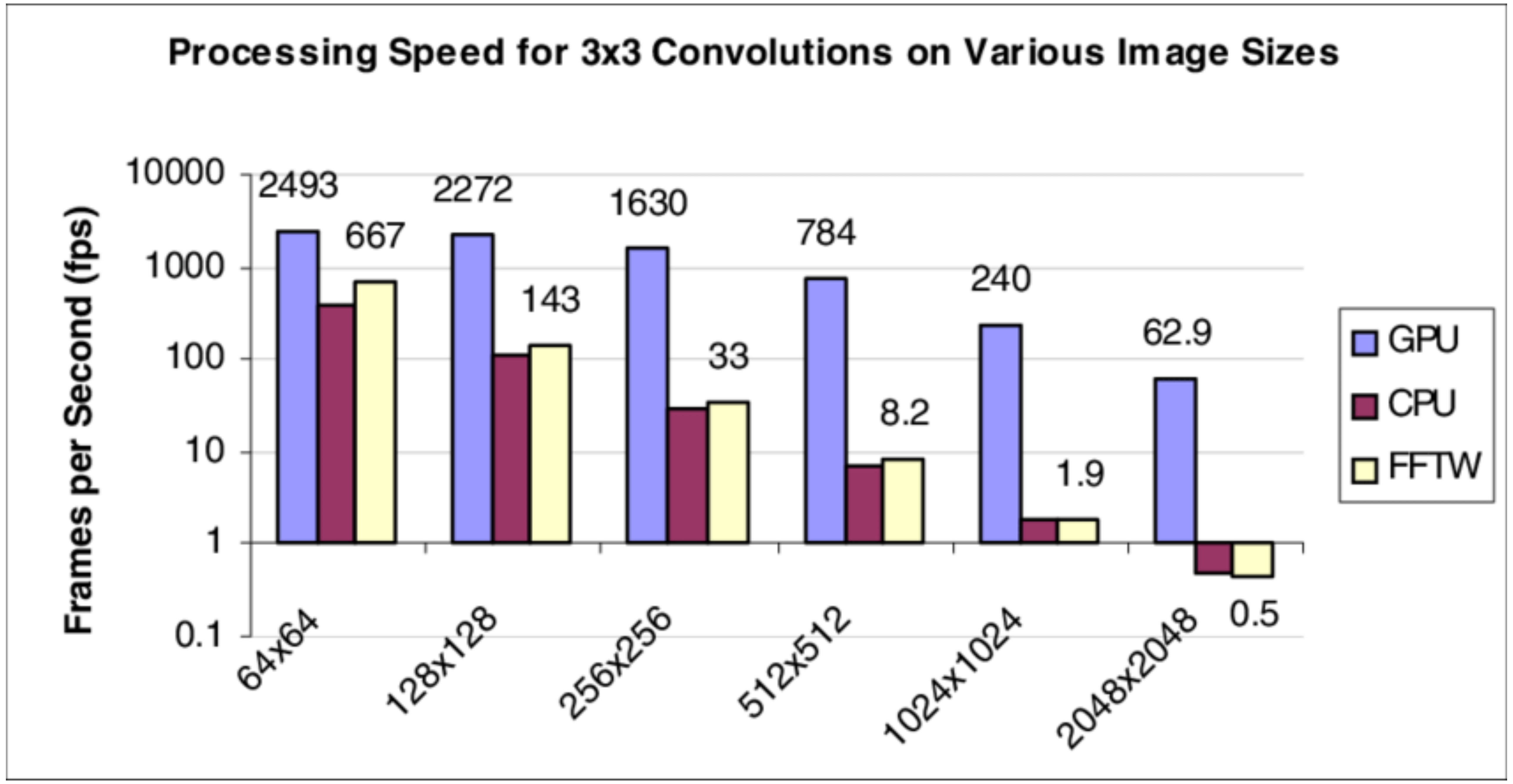
```
__global__ void device_add(int *a, int *b, int *c) {
    int index = threadIdx.x + blockIdx.x * blockDim.x;
    c[index] = a[index] + b[index];
}
```

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

Importance of hardware: GPUs

Convolution is a good example:

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: GPUs

(ML) software abstracts such hardware acceleration away.

We now get autodiff + model blocks + hardware acceleration

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

AI & ML Software Frameworks

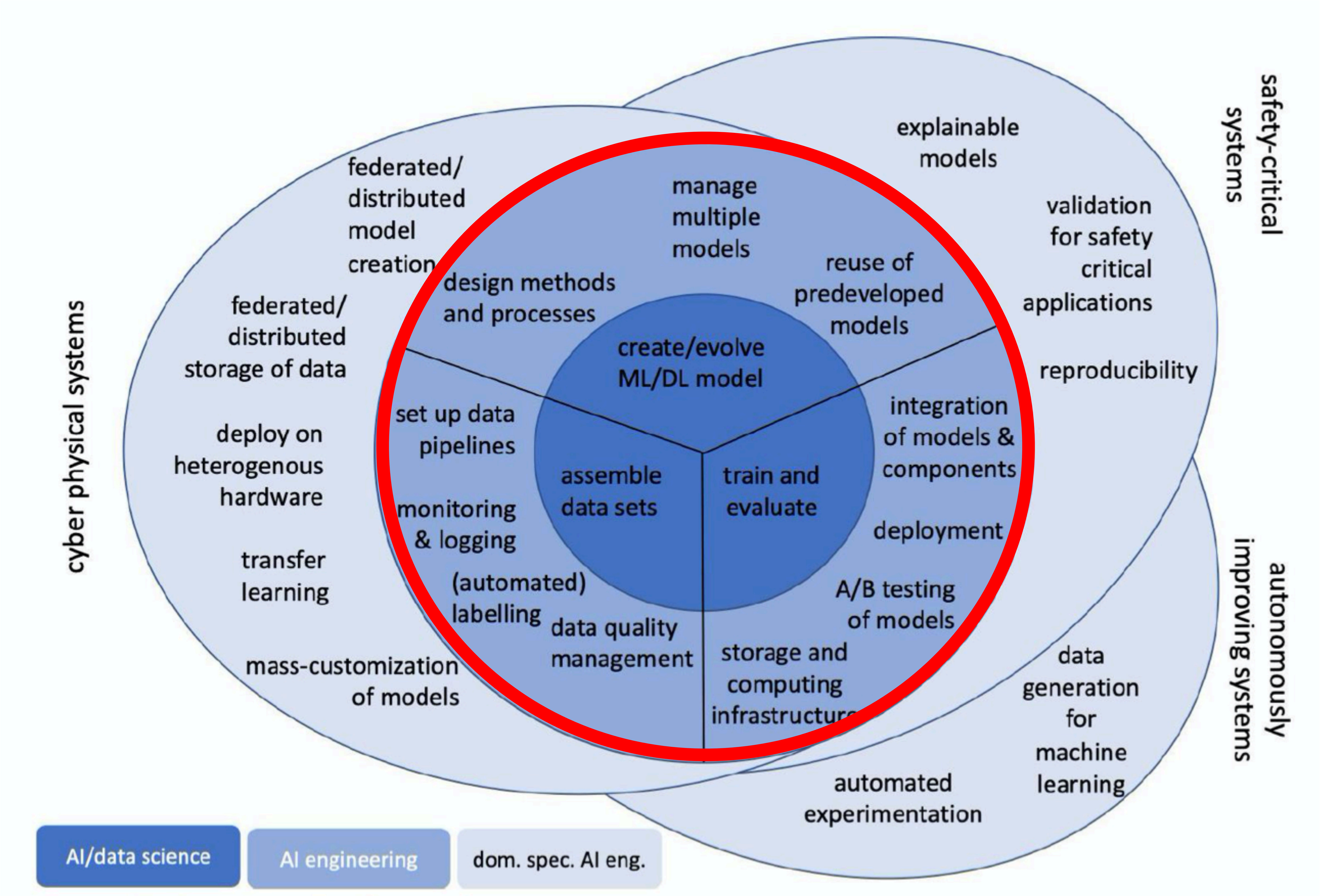


Figure 3: Conceptualization of AI engineering

Bosch et al, "Engineering AI Systems: A Research Agenda", in Artificial Intelligence Paradigms for Smart Cyber-Physical Systems

Inner to outer circles are reflected in/ driven by development of software tools & hardware advances

Software requirements are constantly being reshaped

The ML frameworks competition

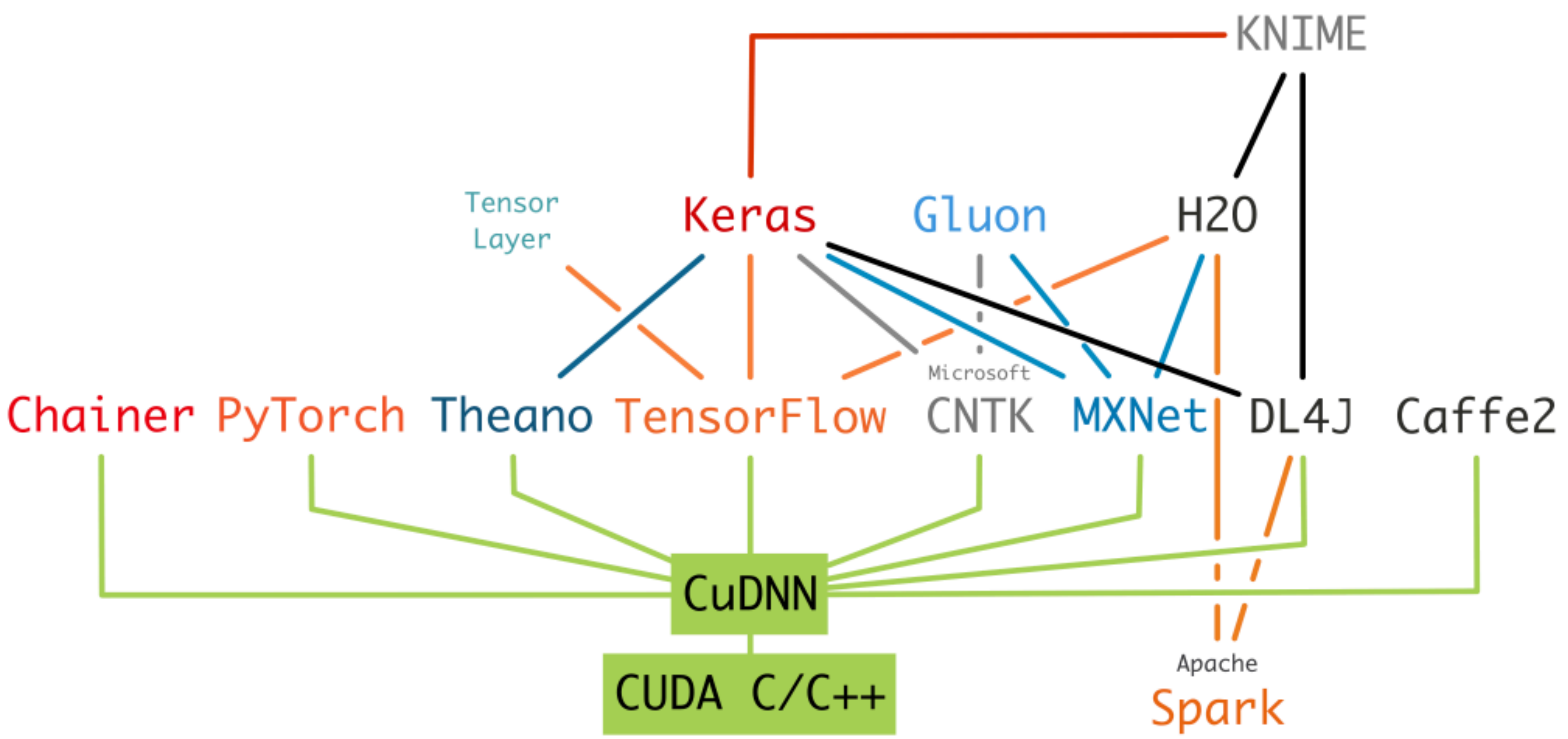


Fig. 3 The most popular Deep Learning frameworks and libraries layering in various abstraction implementation levels

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

- Many frameworks appear (we won't go in detail today, see reference below)
- The core remains: CUDA + autodiff
- More layers for "ease of use" on top
- **More than just model optimization**: data pipelines, reuse of models, monitoring, logging convenience …

Check the following for a good overview: Machine Learning and Deep Learning frameworks and libraries
for large-scale data mining: a survey (Nguyen et al, Artificial Intelligence Review 2019, Springer)

The ML frameworks competition

2016: PyTorch's **graph is dynamically build**. If you simply add another operation, it will be added as the next element in the graph.

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

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

```
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()
```

Static versus dynamic graphs

Continuing to build on top of predecessors

```
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)
```

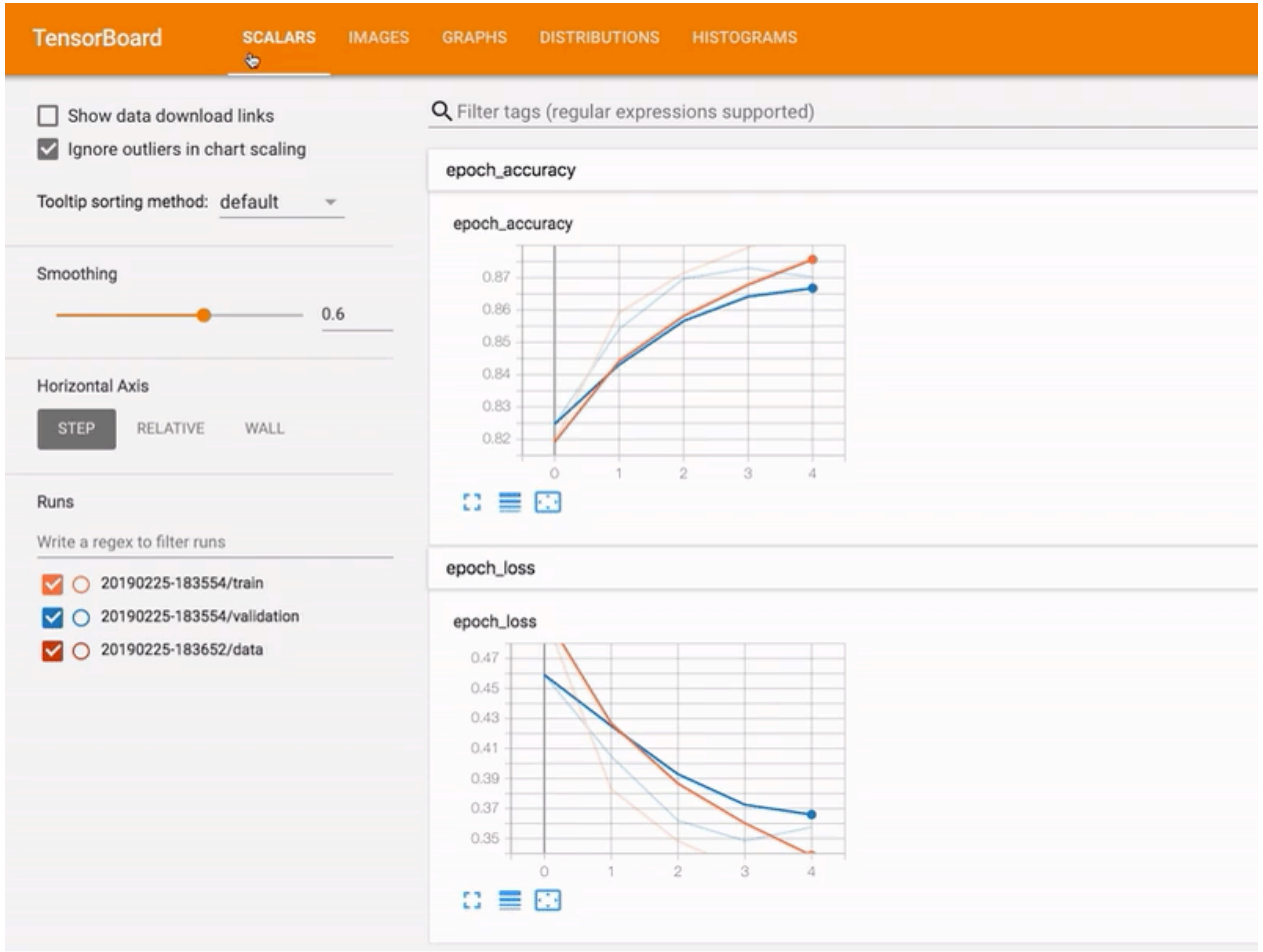
PYTORCH

```
    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

Community, tutorials & workflows

Large focus on improving ease of use & accessibility



Community, tutorials & workflows

Large focus on improving ease of use & accessibility

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

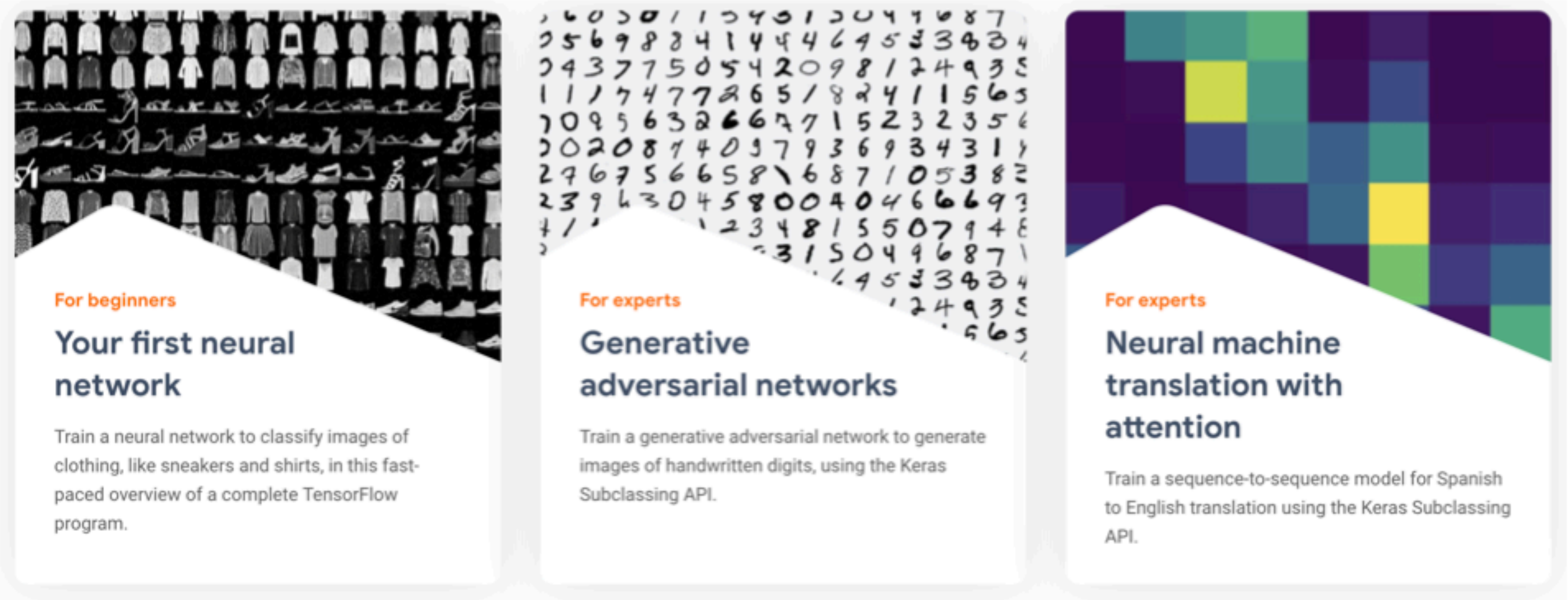
DCGAN Tutorial

Reinforcement Learning

Reinforcement Learning (DQN) Tutorial

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

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



Inclusion of data & deployment

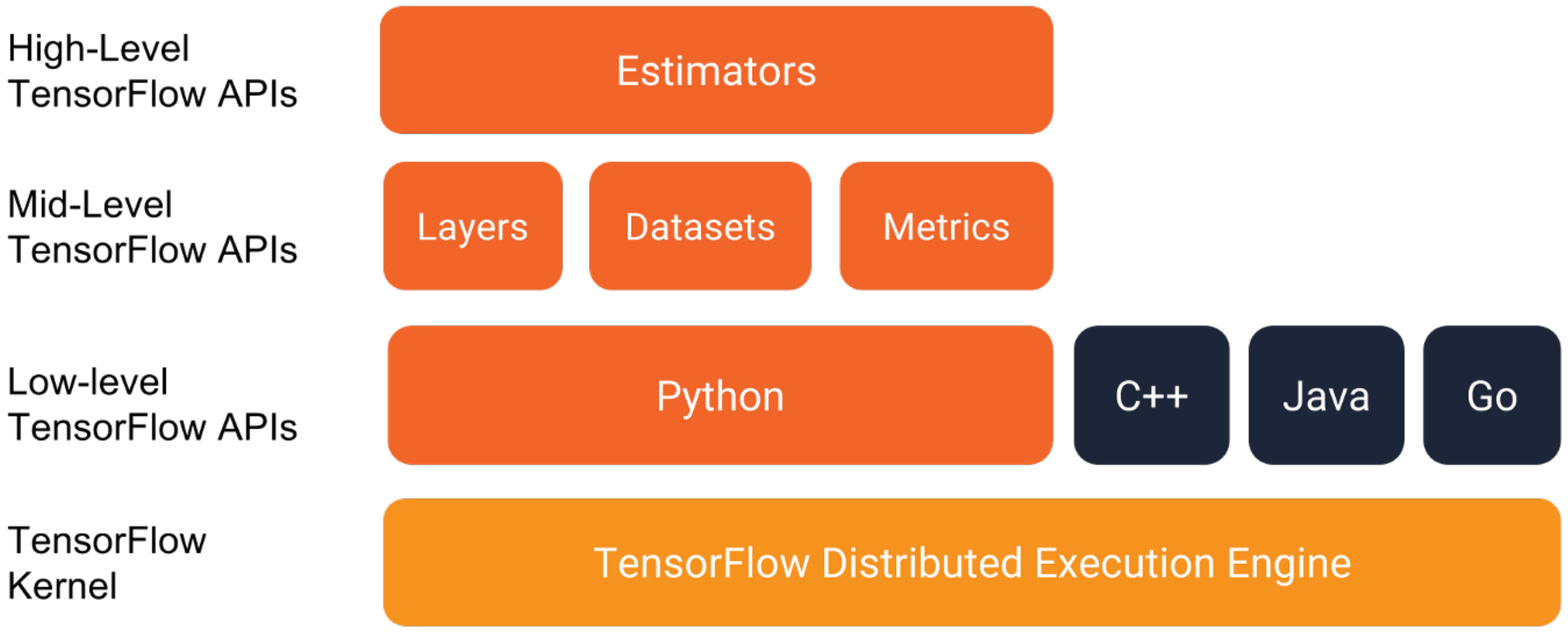
Large focus on improving ease of use & accessibility

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



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

AI & ML Software Frameworks

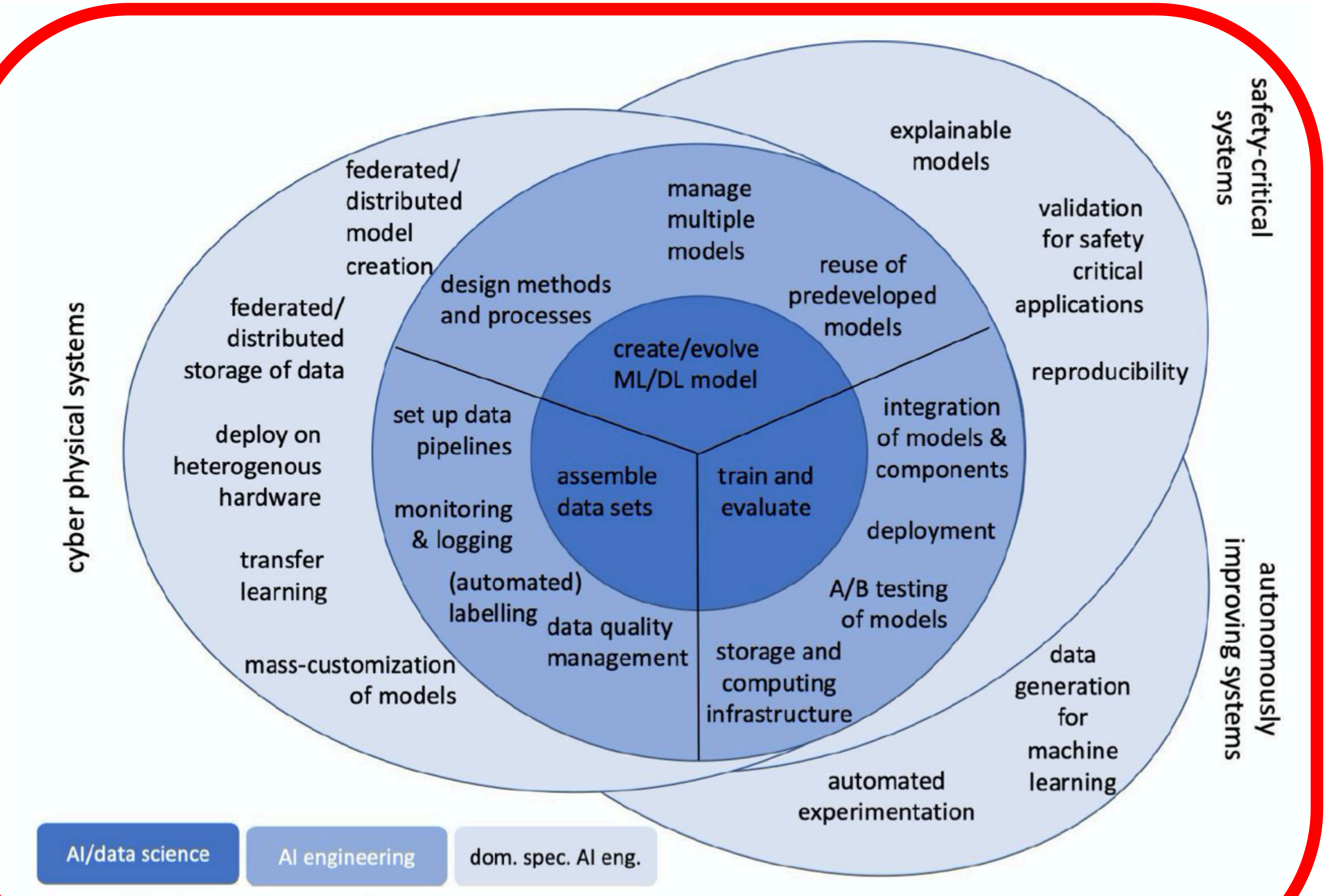


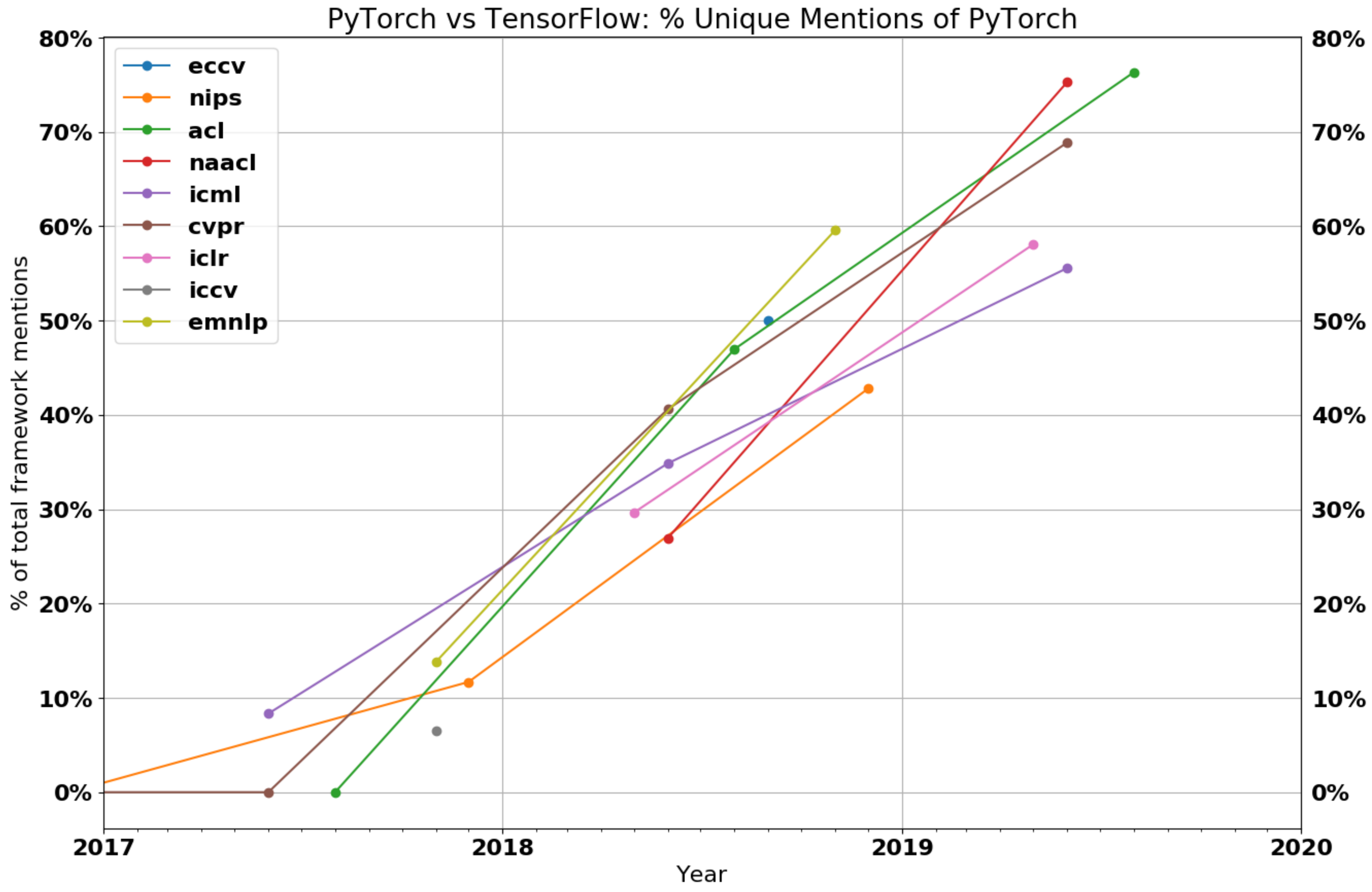
Figure 3: Conceptualization of AI engineering

Bosch et al, "Engineering AI Systems: A Research Agenda", in Artificial Intelligence Paradigms for Smart Cyber-Physical Systems

Inner to outer circles are reflected in/ driven by development of software tools & hardware advances

Software requirements are constantly being reshaped

2019 ML framework convergence



Industry -> TensorFlow 2.0 (2019)?

Academia -> PyTorch 1.0?

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

2019 ML framework convergence

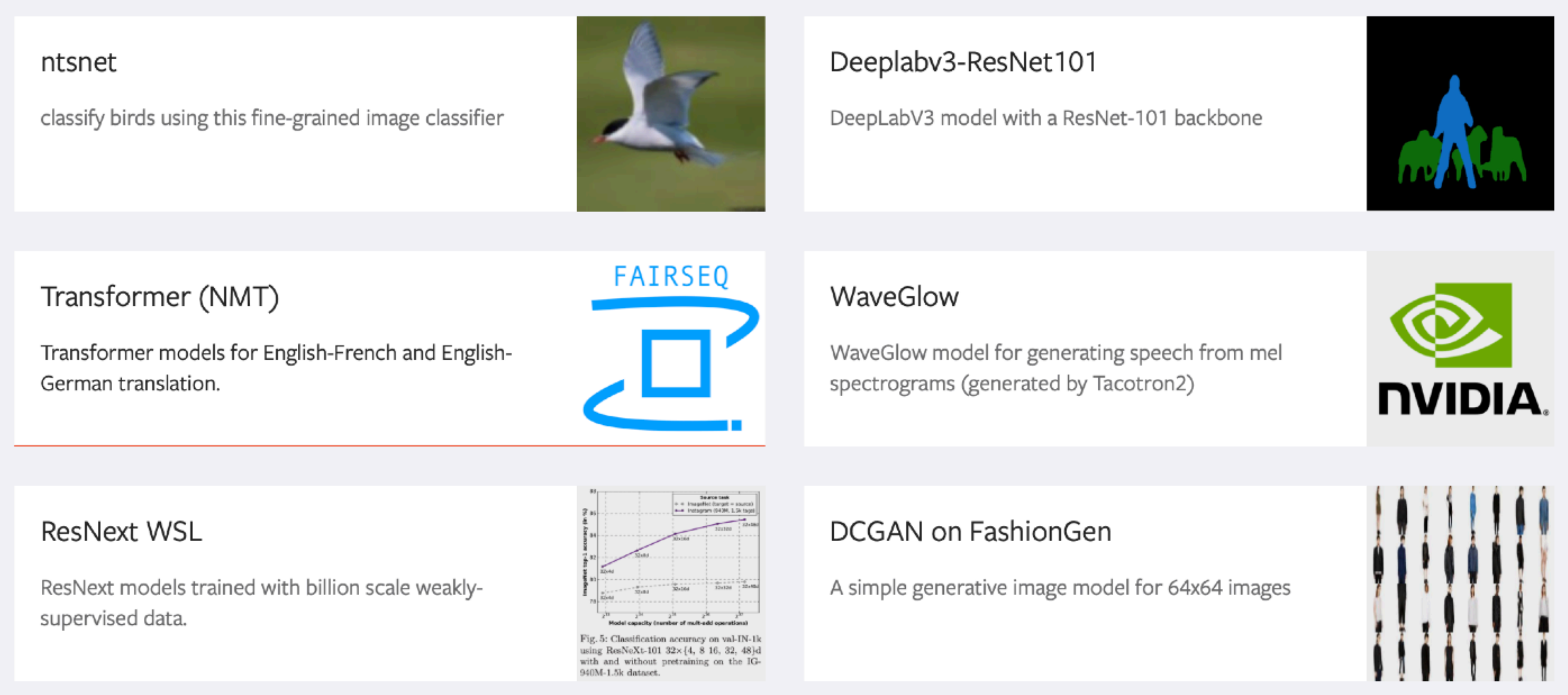
Frameworks keep growing, but are perhaps losing uniqueness? "Core" convergence

- 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
- Remarkably similar features (like TensorBoard, autodiff, CUDA C code …)

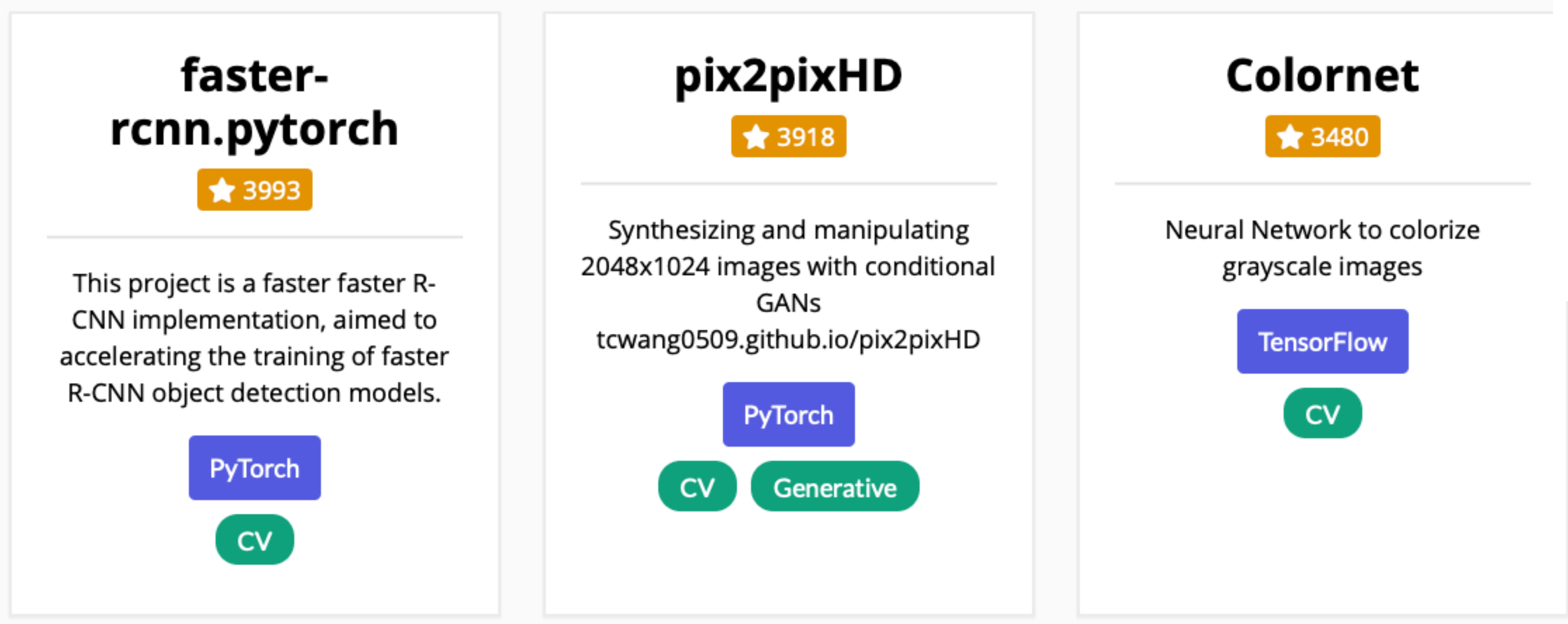
Many other frameworks such as Torch (v7, 2019),
Theano (v1.0, 2019), Chainer (v6.3, 2019),
Microsoft CNTK (2.7, 2019) have officially announced
their last release or have been swallowed.

Sharing, transfer, reproducibility…

OWLL · ContinualAI · hessian.AI · TECHNISCHE UNIVERSITÄT DARMSTADT



https://pytorch.org/hub/



https://modelzoo.co

TensorFlow Model Garden

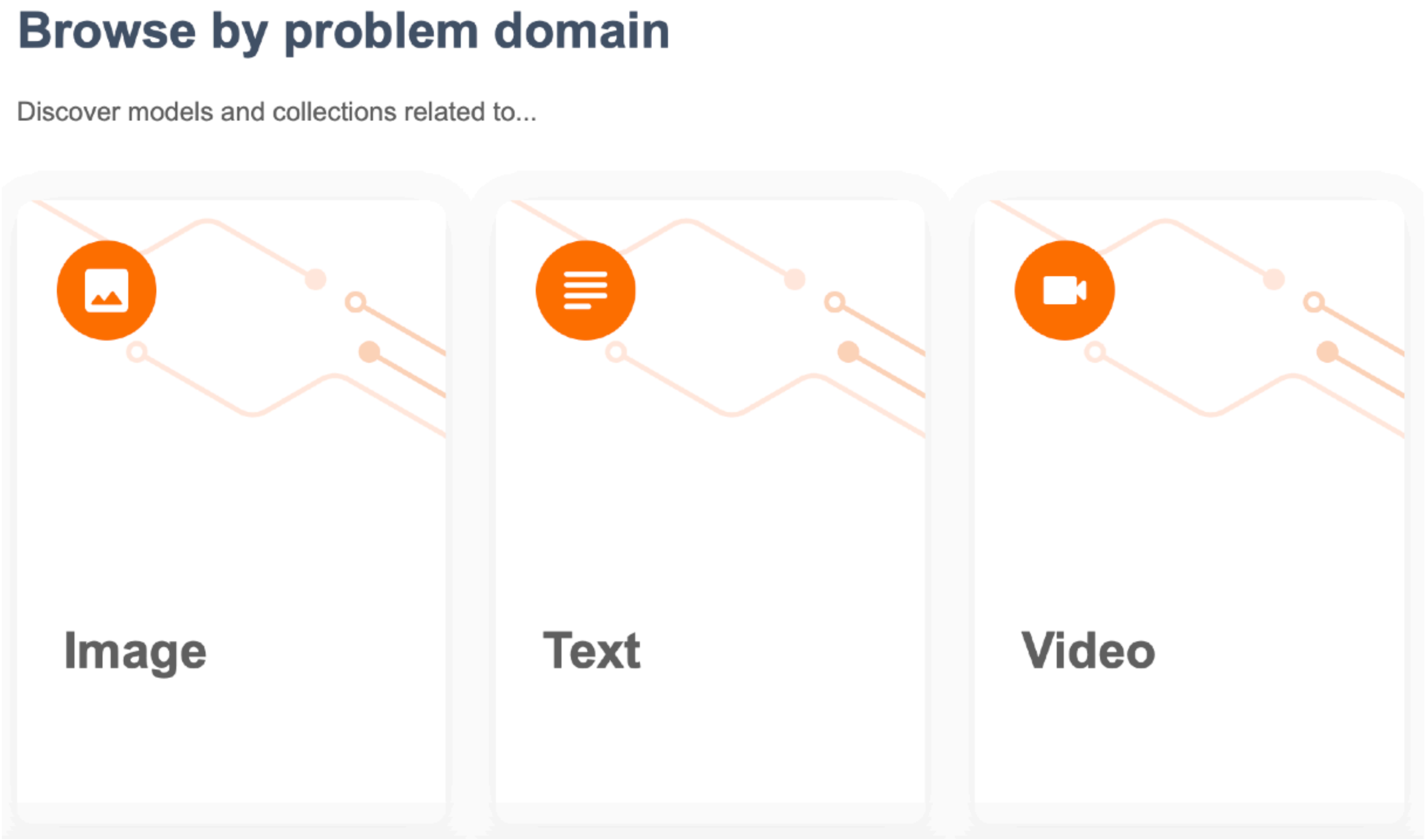
We are starting to emphasize **sharing, transferring & reproducing**

The AI community building the future.

Build, train and deploy state of the art models powered by the reference open source in machine learning.

Star 66,356

https://huggingface.co



https://tfhub.dev

ONNX

https://onnx.ai

Heterogeneous hardware

- Introduced **mobile support**, support for TPUs (tensor processing units)
- Adapted code towards **device agnostic programming** by introducing generic commands such as “.to(device)”
- Native swift, javascript versions and a set of **browser based applications**

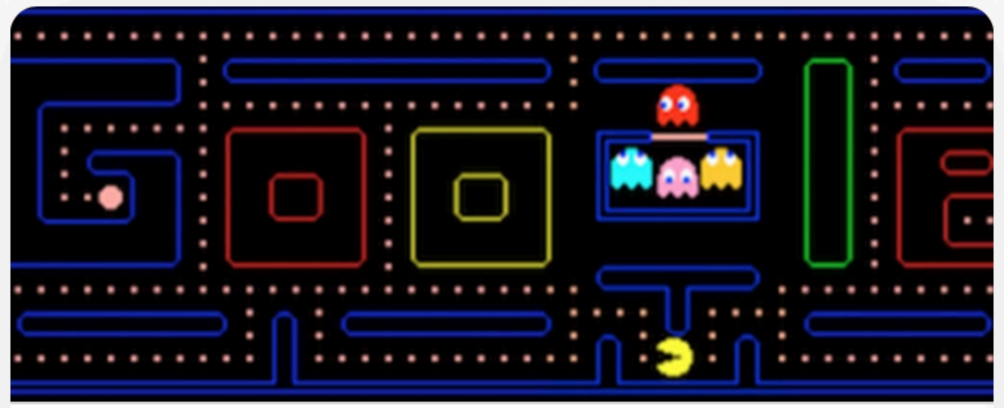
Demos

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?

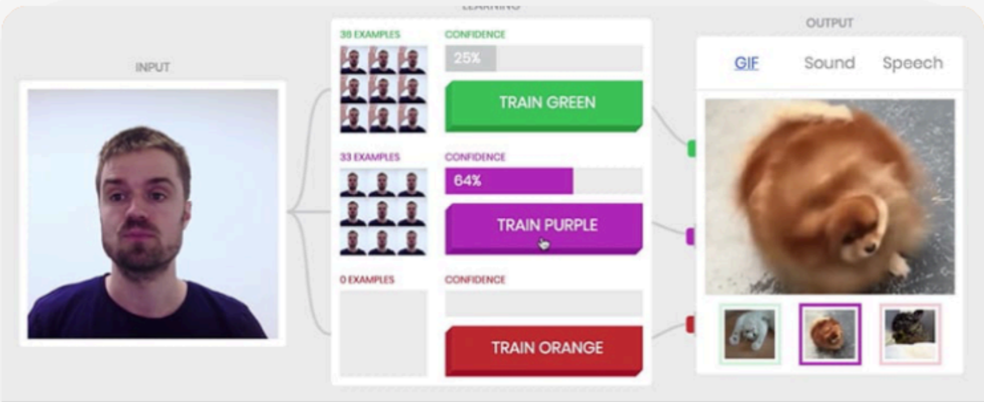
Explore demo ↗ View code



Webcam Controller

Play Pac-Man using images trained in your browser.

Explore demo ↗ View code



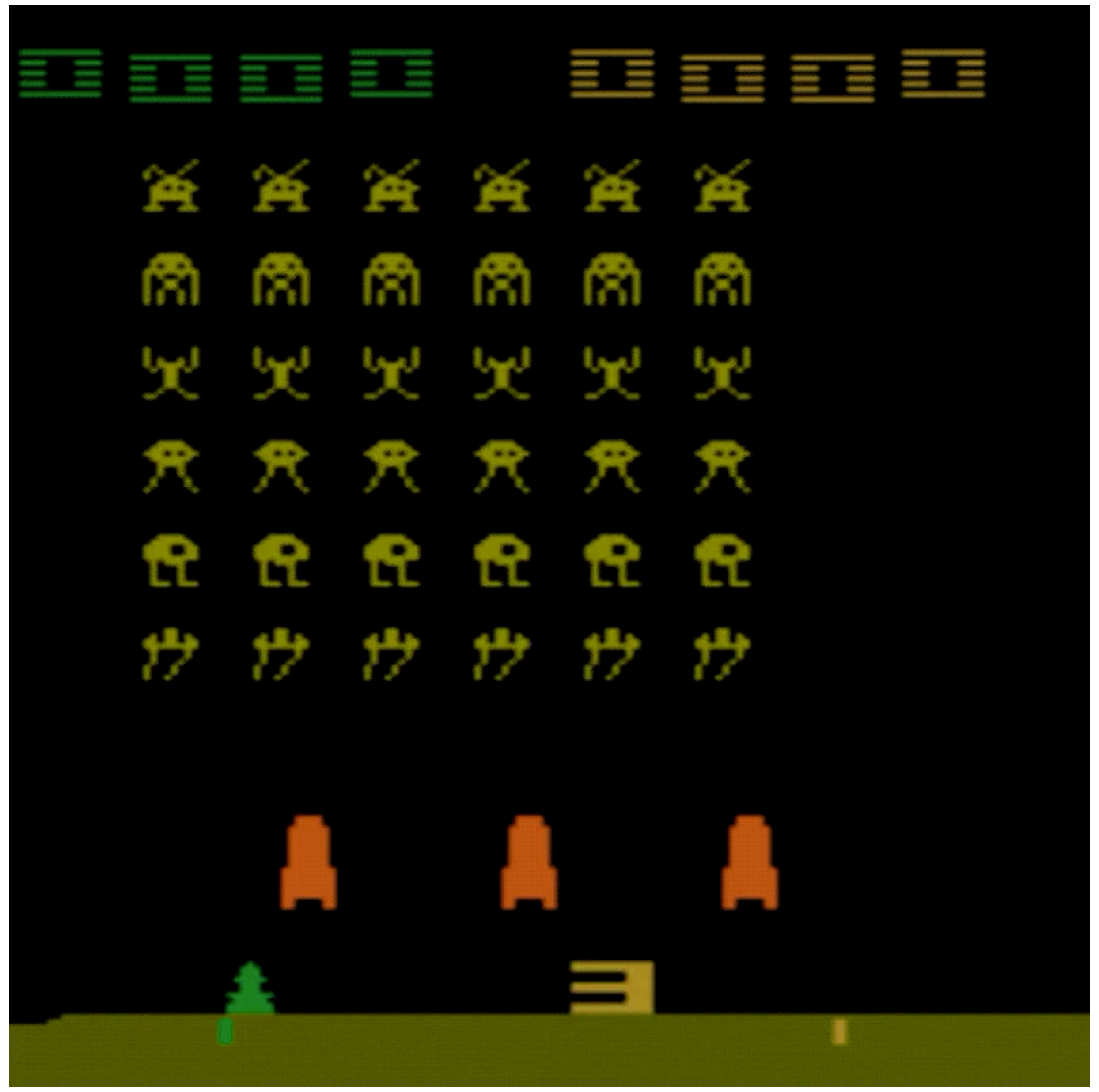
Teachable Machine

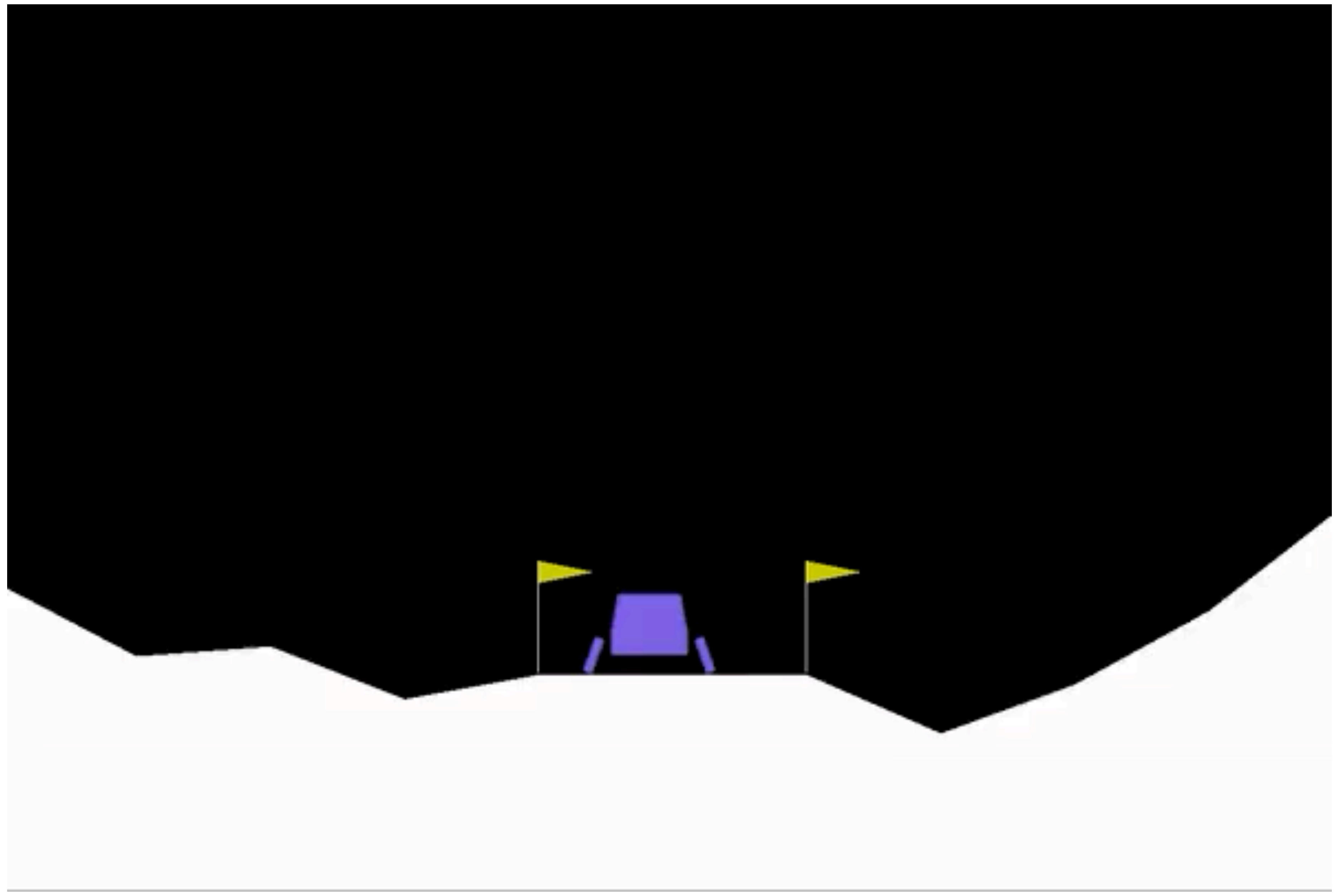
No coding required! Teach a machine to recognize images and play sounds.

Explore demo ↗ View code

Learning environments

We are starting to see inclusion of simulation environments, e.g. "gyms" for reinforcement learning, & various other 3-D graphics simulators





https://gym.openai.com

Learning environments

We are starting to see inclusion of simulation environments, e.g. “gyms” for reinforcement learning, & various other 3-D graphics simulators

http://animalaiolympics.com/AAI/

AI & ML Software Frameworks

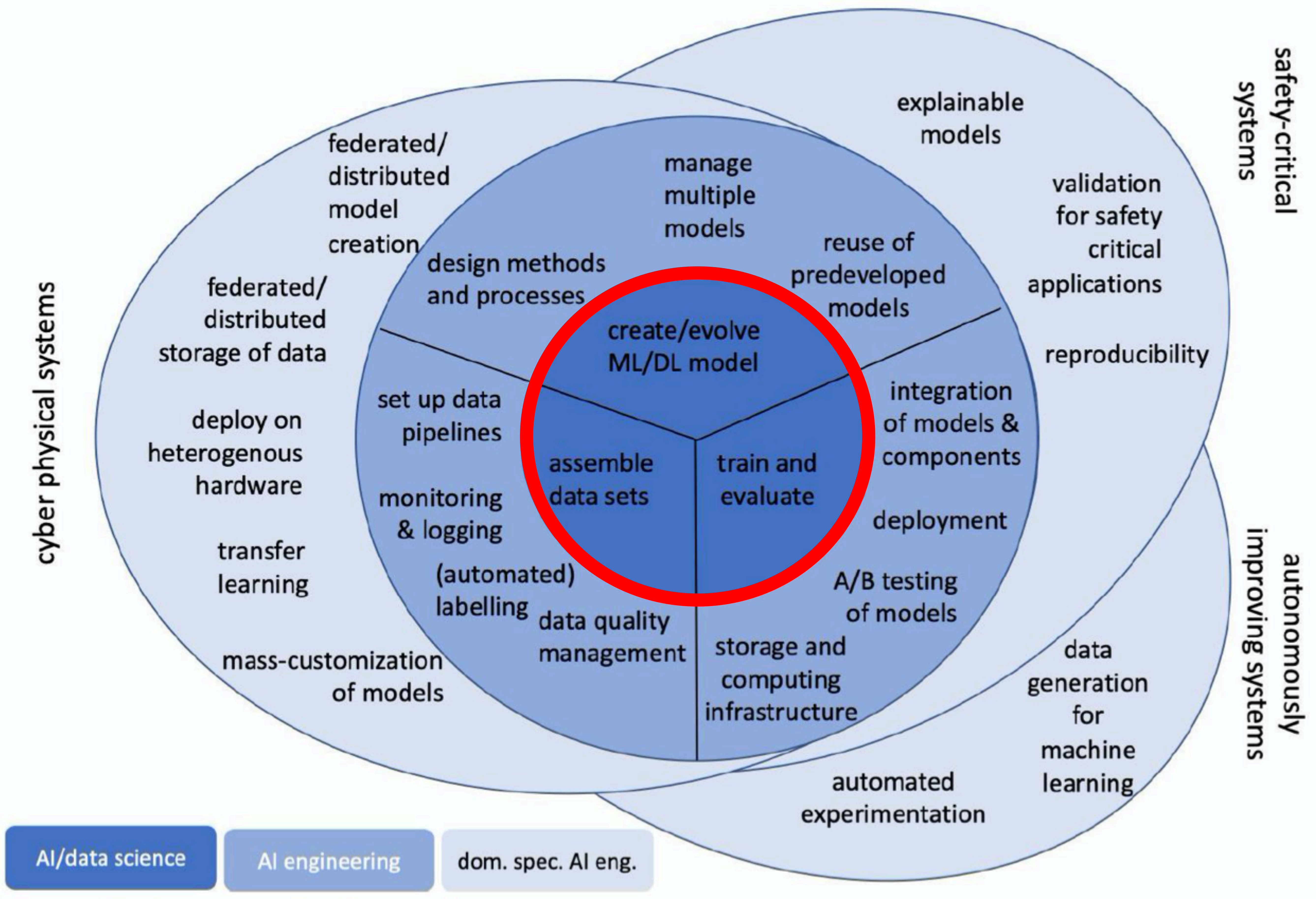


Figure 3: Conceptualization of AI engineering

Bosch et al, "Engineering AI Systems: A Research Agenda", in Artificial Intelligence Paradigms for Smart Cyber-Physical Systems

What if we want or need to revisit the center?

Have our frameworks converged too much?

Limitations induced by soft/hardware

Example: **capsule networks** as a recent neural network variant

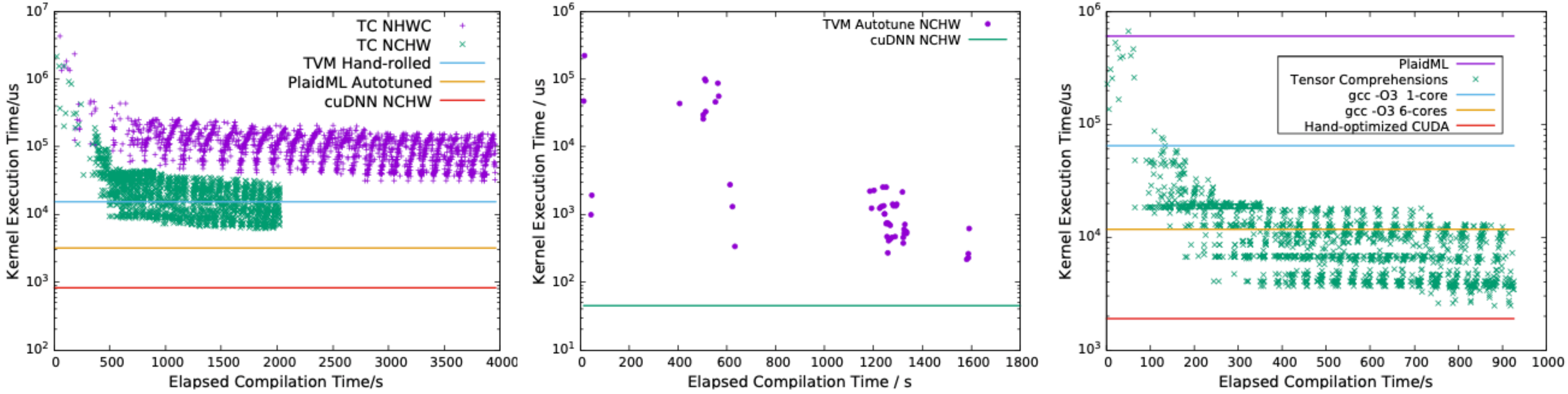
It is not trivial to **optimize operations over multiple dimensions** and there is a hardware **preference for specific memory layouts** such as batch-channel-width-height (BCWH).

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

Machine Learning Systems are Stuck in a Rut, Workshop on Hot Topics in Operating Systems, Barham and Isard 2019

Limitations induced by soft/hardware



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

Figure 3. Performance comparison of autotuned kernels

This is **counterintuitive & non-optimal**, since operations such as convolutions are polymorphic over number of dimensions
-> e.g. what if we want to use more dimensions?

Machine Learning Systems are Stuck in a Rut, Workshop on Hot Topics in Operating Systems, Barham and Isard 2019

Limitations induced by soft/hardware

"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"

- **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.**

Machine Learning Systems are Stuck in a Rut, Workshop on Hot Topics in Operating Systems, Barham and Isard 2019

Limitations induced by soft/hardware

"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****."*

Machine Learning Systems are Stuck in a Rut, Workshop on Hot Topics in Operating Systems, Barham and Isard 2019

Limitations induced by soft/hardware

“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/

So what about continual learning, open worlds, …?

AI & ML Software Frameworks

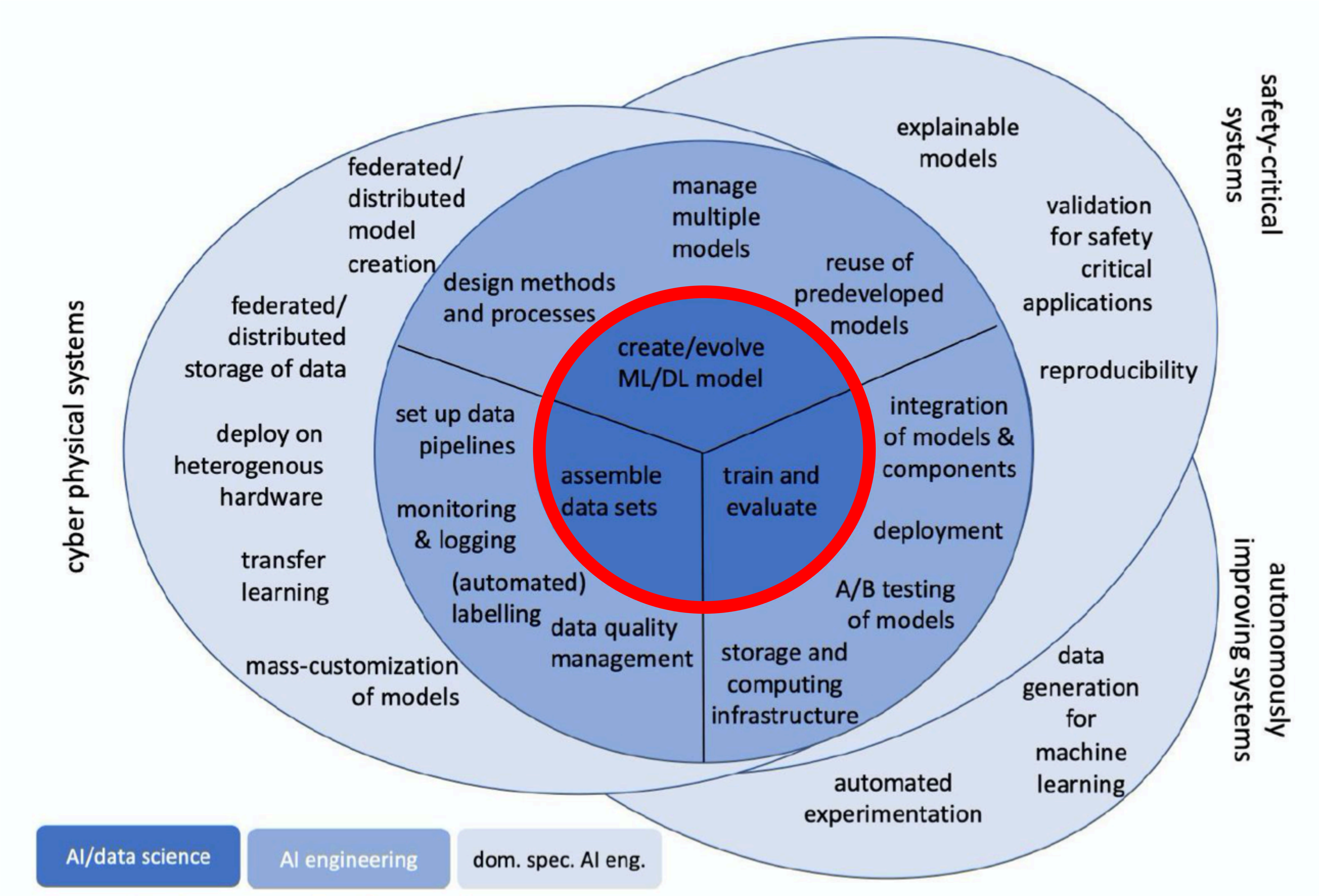


Figure 3: Conceptualization of AI engineering

Bosch et al, “Engineering AI Systems: A Research Agenda”, in Artificial Intelligence Paradigms for Smart Cyber-Physical Systems

We are slowly moving towards continual learning, but our software is still heavily focused on a typical “train-val-test” idea

Perhaps we require a revisit?

Guest lecture part: Avalanche
By Dr. Antonio Carta

powered by