Microsoft

人工智能系统 System for AI
课程介绍 Lecture Introduction

主要内容

- 不包括的内容
  - 特定的深度学习算法
  - 如何编写深度学习模型
  - 如何使用深度学习工具
- 主要内容
  - 深度学习系统
  - 设计、实现和优化针对深度学习的系统和工具
  - 例如：如何从计算机体系结构、编译器、编程语言和分布式系统等方面支持深度学习
  - *人工智能 AI 和 深度学习 deep learning 会在本课程交替使用

先修知识

- 编程语言：C/C++, Python
- 机器学习（基本原理和知识）
- 计算机组成原理
- 计算机体系结构（有一定了解）
- 计算机网络基础

主要目的

- 系统学习并掌握深度学习系统的原理及组成。
  - 通过实例全面了解深度学习系统
  - 了解深度学习整个生命周期需要的系统技术
- 了解深度学习系统领域前沿的研究方向。
  - 通过课程和课后论文阅读了解系统与人工智能交叉领域的前沿研究方向
  - 提高定义和选择研究问题的能力
- 通过实验来提高动手搭建系统和解决问题的能力。
  - 简单而生动的实验深入理解系统内部原理
  - 为以后设计实现人工智能相关的系统打好基础

课程大纲 – 基础知识

课程	讲义名称	备注
1	课程介绍	Overview and system/AI basics
2	人工智能系统概述 System perspective of System for AI	System for AI: a historic view; Fundamentals of neural networks; Fundamentals of System for AI
3	深度神经网络计算框架基础 Computation frameworks for DNN	Backprop and AD, Tensor, DAG, Execution graph Papers and systems: PyTorch, TensorFlow
4	矩阵运算与计算机体系结构 Computer architecture for Matrix computation	Matrix computation, CPU/SIMD, GPGPU, ASIC/TPU Papers and systems: Blas, TPU
5	分布式训练算法 Distributed training algorithms	Data parallelism, model parallelism, distributed SGD Papers and systems:
6	分布式训练系统 Distributed training systems	MPI, parameter servers, all-reduce, RDMA Papers and systems: Horovod
7	异构计算集群调度与资源管理系统 Scheduling and resource management system	Running DNN job on cluster: container, resource allocation, scheduling Papers and systems: KubeFlow, OpenPAI, Gandiva, HiveD
8	深度学习推导系统 Inference systems	Efficiency, latency, throughput, and deployment

课程大纲 （2） - 高级技术

课程	讲义名称	备注
9	计算图编译优化 Computation graph compilation and optimization	IR, sub-graph pattern match, Matrix multiplication and memory optimization Papers and systems: XLA, MLIR, TVM, NNFusion
10	模型压缩和稀疏化处理 Efficiency via compression and sparsity	Model compression, Sparsity Pruning
11	自动机器学习系统 AutoML systems	Hyper parameter tuning, NAS Papers and systems: Hyperband, SMAC, ENAS, AutoKeras, NNI
12	强化学习系统 Reinforcement learning systems	Theory of RL, systems for RL Papers and systems: AC3, RLlib, AlphaZero
13	模型安全与隐私保护 Security and Privacy	Federated learning, security, privacy Papers and systems: DeepFake
14	用AI技术优化计算机系统 AI for systems	AI for traditional systems problems, for system algorithms Papers and systems: Learned Indexes, Learned query path

课程大纲 （3） - 实验

课程	讲义名称	备注
Lab 1 (for week 1,2)	框架及工具入门示例 A simple throughout end-to-end AI example, from a system perspective	Understand the systems from debugger info and system logs
Lab 2 (for week 3)	定制一个新的张量运算 Customize operators	Design and implement a customized operator (both forward and backward): in python
Lab 3 (for week 4)	CUDA实现和优化 CUDA implementation	Add a CUDA implementation for the customized operator
Lab 4 (for week 5,6)	AllReduce实现和优化 AllReduce	Improve one of AllReduce operators' implementation on Horovod
Lab 5 (for week 7, 8)	配置Container来进行云上训练或推理准备 Configure containers for customized training and inference	Configure containers
Lab 6	学习使用调度管理系统 Scheduling and resource management system	Get familiar with OpenPAI or KubeFlow
Lab 7	分布式训练任务练习 Distributed training	Try different kinds of all reduce implementations
Lab 8	自动机器学习系统练习 AutoML	Search for a new neural networkNN structuree for Image/NLP tasks
Lab 9	强化学习系统练习 RL Systems	Configure and get familiar with one of the following RL Systems: RLlib, …

课程资源

- 课程网站： https://github.com/microsoft/ai-system
- 相关知识：https://github.com/microsoft/ai-edu

Microsoft

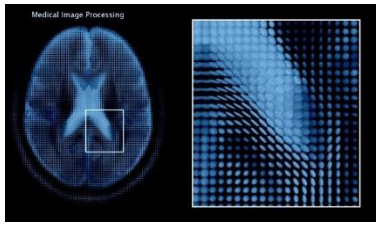
人工智能系统概述
System perspective of System for AI

人工智能系统：概览
System for AI: a historic view

Self-driving

Surveillance detection

Translation



Medical diagnostics

Game

Personal assistant

Deep Learning
深度学习正在改变世界

Art

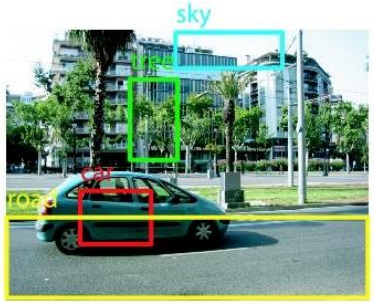
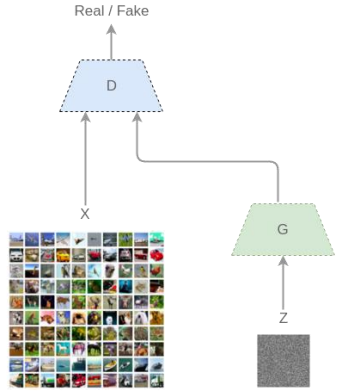


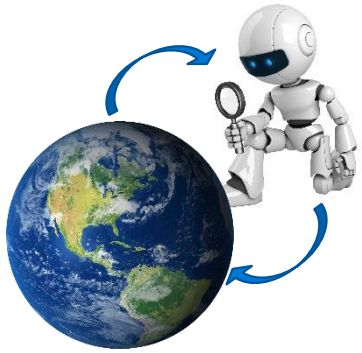
Image recognition

Speech recognition

Natural language

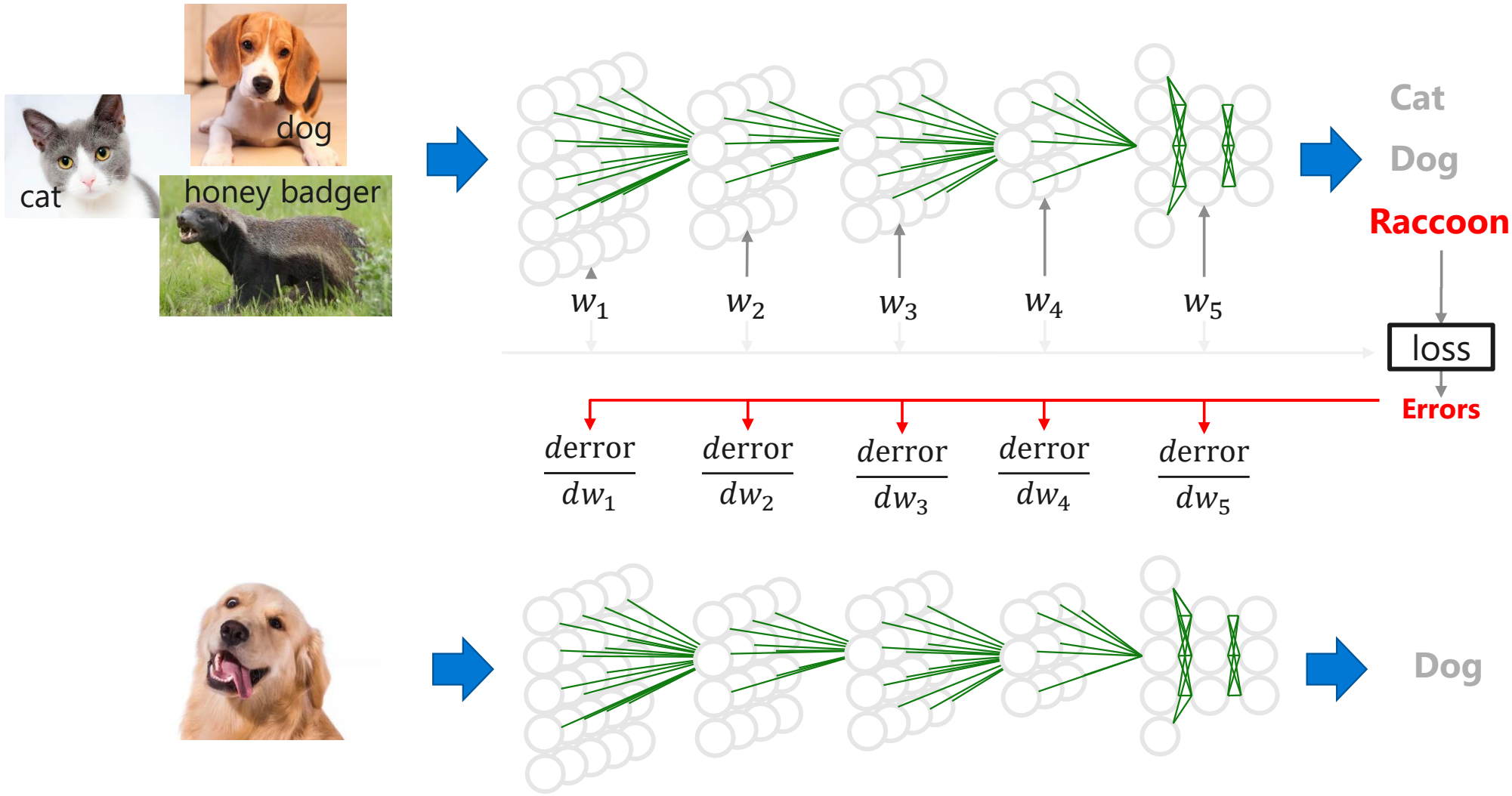


Generative model

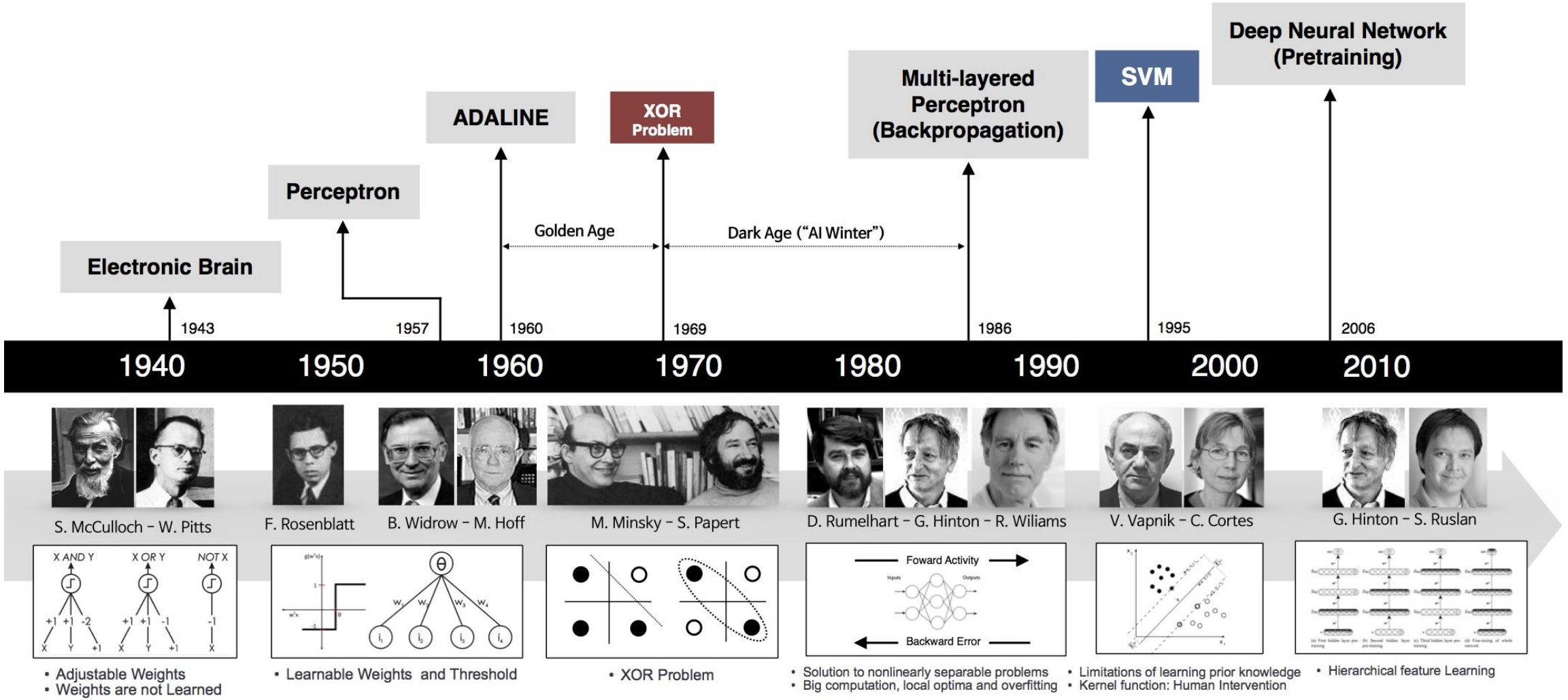


Reinforcement learning

深度学习方法 Deep Learning Approach

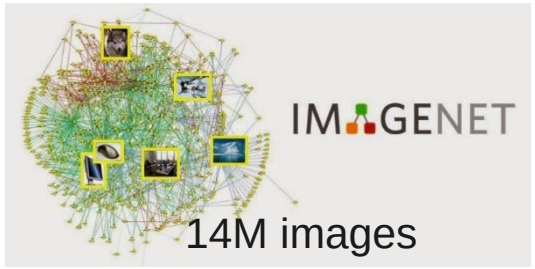


神经网络的基本理论在深度学习前已基本奠定

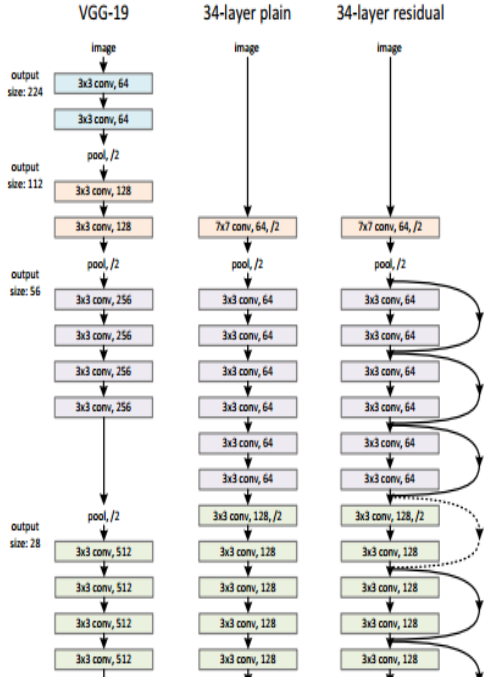


为什么深度学习在最近十年才变得成功?

海量的（标识）数据



深度学习算法的进步



语言、框架

TensorFlow

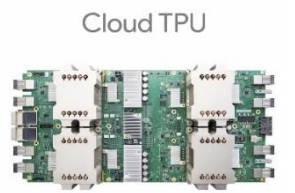
Microsoft CNTK

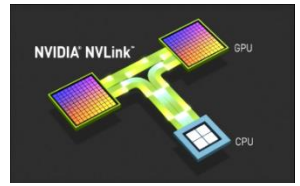
Caffe2

dmlc mxnet

PYTORCH

计算能力





深度学习+系统的进步: 编程语言、优化、计算机体系结构、并行计算以及分布式系统

大数据和分布式系统

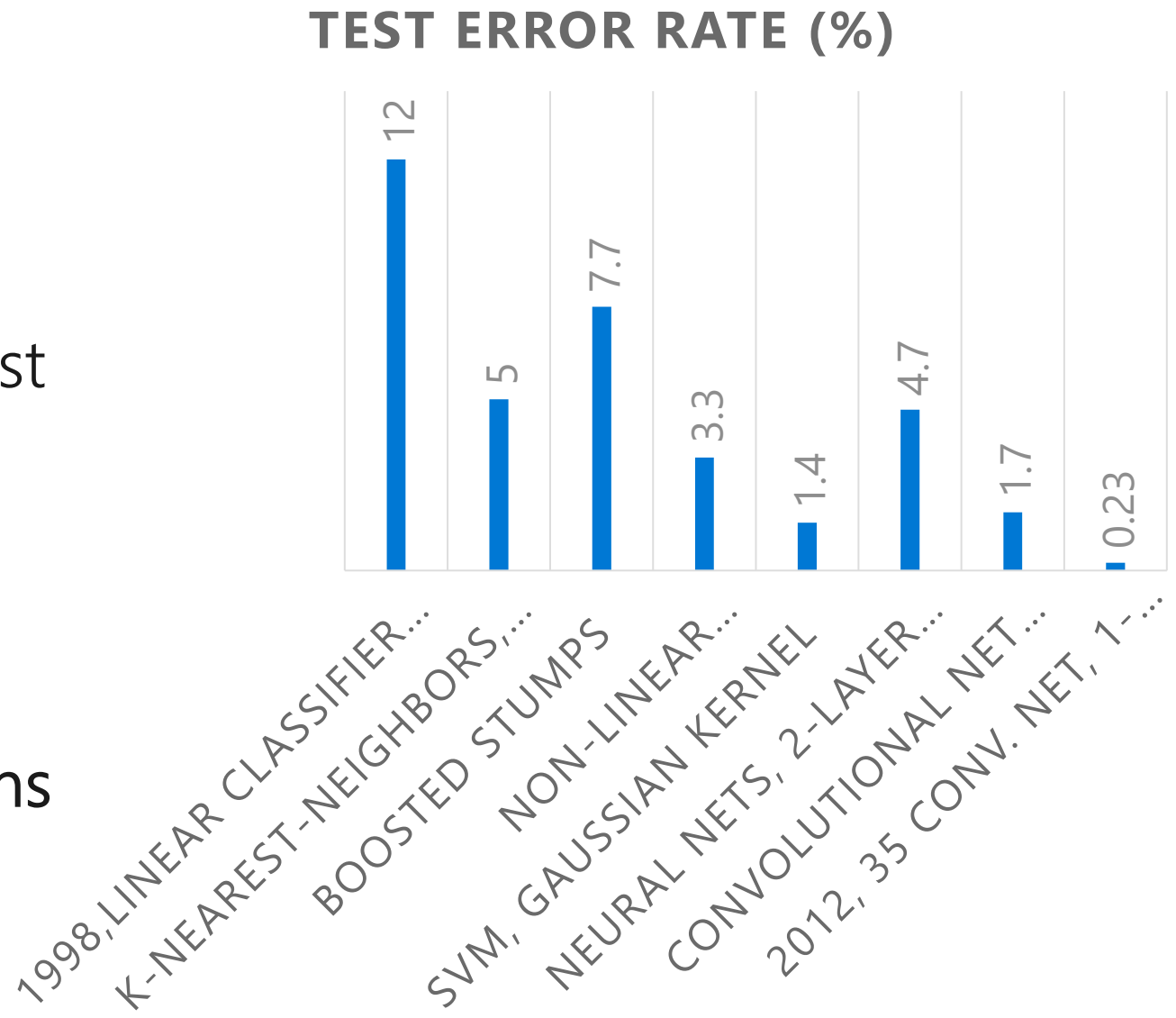
- 互联网服务和大数据平台给深度学习带来了大量的数据集
  - 搜索引擎 (Search Engine)
    - Image search: ImageNet, Coco
    - Text search: Wikipedia, Natural language datasets
  - 商业网站
    - Amazon, Taobao: recommendation data sets, ads data sets
  - 其他互联网服务：Internet services
    - Conversational services: XiaoIce, Siri, Cortana

E.g., image classification problem

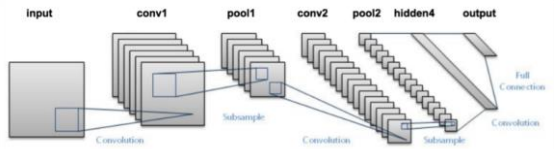
MNIST	ImageNet	Web Images
60K samples	16M samples	Billions of Images
10 categories	1000 categories	Opened categories

深度学习算法的进步 – e.g., MNIST

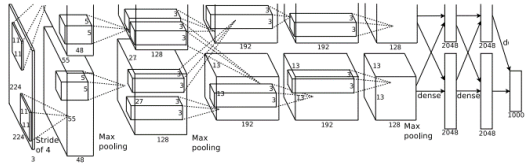
- THE MNIST DATABASE of handwritten digits
- A simple convolutional NN approach can equal the best SVM approach (1998)
- A deep convolutional NN approach can reduce the error rate to 0.23% (2012), which is **compare to humans** at 0.2%



深度学习算法的进步 – e.g., IMAGENET

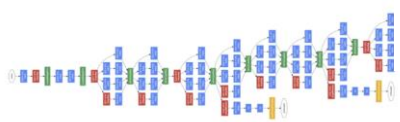


LeNet, convolution, max-pooling, softmax, 1998

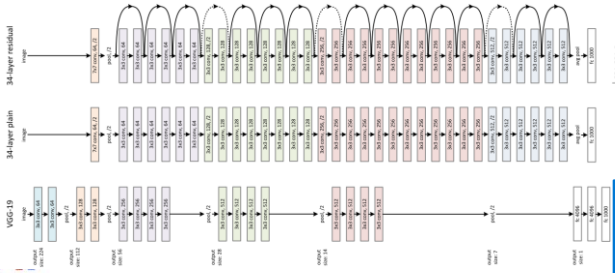


AlexNet, **16.4%**

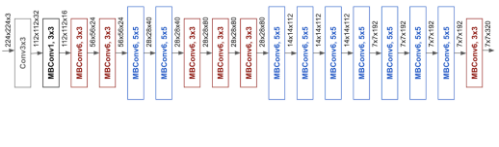
ReLU, Dropout, 2012



Inception,**6.7%**
Batch normalization, 2015



ResNet,**3.57%**
Residual way, 2015



EfficientNet, **3.1%**

NAS

2019

- 更好的激活函数和结构: ReLU, BN, …
- 更复杂的网络和更多参数:
- 更好的训练方法: regularization, initialization, learning methods

深度学习算法的进步 – 其他领域 (human parity or +)

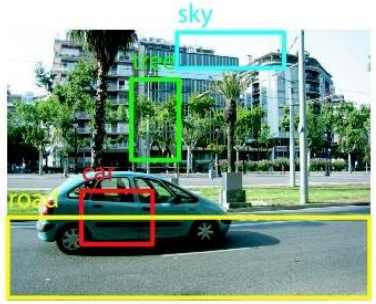


Image recognition

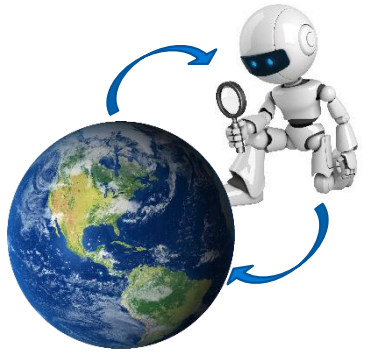
- 2015, ImageNet, MSRA, Resnet

Natural language

- 2019, CoQA, SQuAD, MSRA, Multistage Multitask Learning

Speech recognition

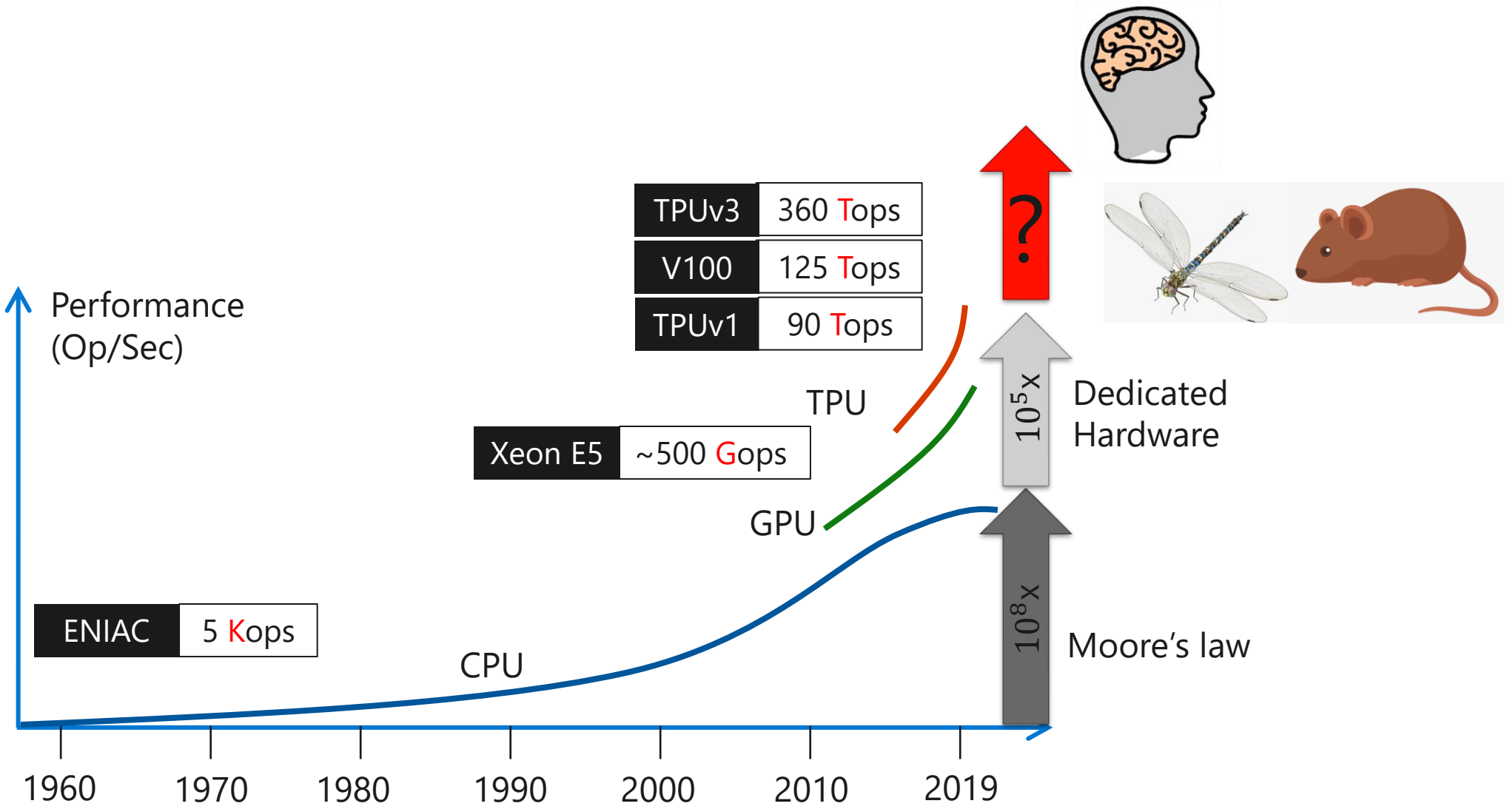
- 2016, NIST 2000, MSR, Combined model



Reinforcement learning

- 2016, Go Game, DeepMind
- 2019, Dota2, OpenAI

计算机体系结构和计算能力的进步



计算框架的进步

Gen 1 *pre-2010*

Gen 2 *2010-present*

Frameworks

Custom purpose machine learning **algorithms**

Theano
DisBelief
Caffe

Deep learning frameworks

MxNet
TensorFlow
CNTK
PyTorch

Language Frontend

Swift for TensorFlow

Compiler Backend

TVM
TensorFlow XLA

Hardware

Algebra & linear libs

- CPU
- GPU

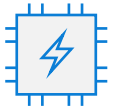
Dense matmul engine

- GPU
- FPGA

Special AI accelerators

- TPU
- GraphCore
- Other ASICs

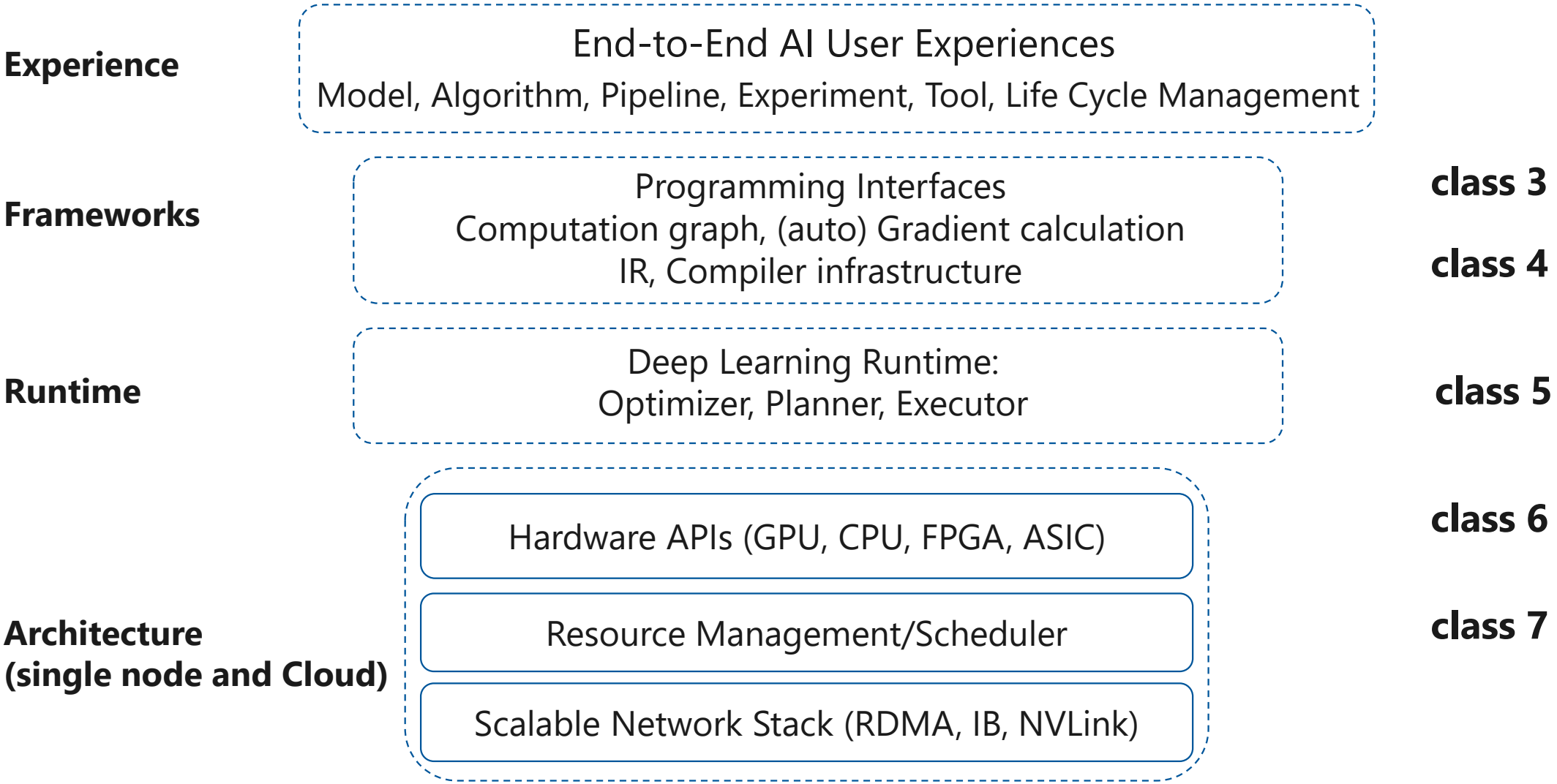
计算框架的进步 (2)

	Gen 1 *pre-2010*	Gen 2 *2010-present*	**Gen 3** *present-*
Frameworks	**Custom** purpose machine learning **algorithms**  Theano DisBelief Caffe	**Deep learning frameworks** provide easier ways to leverage various libraries  	**Machine Learning Language and Compiler**  A Full-Featured Programming Language for ML: Expressive and flexible Control flow, recursion, sparsity  Powerful Compiler Infrastructure: Code optimization, sparsity optimization, hardware targeting
Hardware	**Algebra & linear libs** • CPU • GPU	AI framework Dense matmul engine	 SIMD → MIMD Sparsity Support Control Flow and Dynamicity Associated Memory

深度学习系统的目的

- 提供更加高效的编程语言、框架和工具
  - 更具表达能力和简洁的神经网络计算原语和编程语言
  - 更直观的编辑、调试和实验工具
  - 整个深度学习生命周期中的系统问题：模型压缩、推理、安全、隐私保护等
  - 提供全面的学习系统：强化学习、自动机器学习等
- 提供更强大和可扩展的计算能力
  - 自动编译优化算法，包括不限于：
  - 自动推导计算图
  - 根据不同体系结构自动并行化
  - 自动分布式化，并扩展到多个计算节点
  - 持续优化
- 探索并解决新挑战下的系统设计、实现和演化的问题

深度学习系统的大致组成（Deep Learning System Stack）



深度学习系统生态

更广泛的AI系统生态

class 8 机器学习新模式 (RL)

class 9 自动机器学习 (AutoML)

class 10 安全与隐私

class 11 模型推导、压缩与优化

深度学习算法和框架

- 广泛用途的高效新型通用AI算法
- 多种深度学习框架的支持与进化
- 深度神经网络编译架构及优化

核心系统软硬件

- 深度学习任务运行和优化环境
- 通用资源管理和调度系统
- 新型硬件及相关高性能网络和计算栈

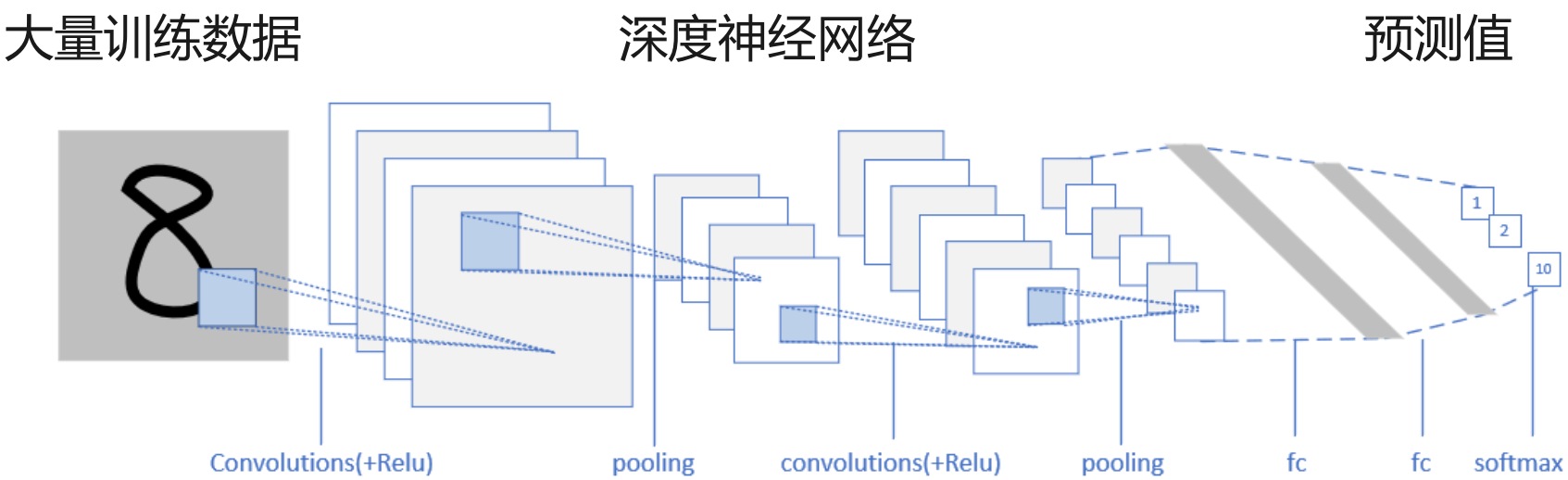
小结

- 深度学习的进步来源于算法、数据、系统等多方面的突破
  - 系统新问题总是与新的应用问题和挑战相伴而生
- 系统的解决问题意味着需要全面深入的了解整个问题空间
  - 任何维度的短板都会影响整个系统
- 下面将介绍深度神经网络的基础知识和深度学习系统概览
  - 基础性的知识和概览

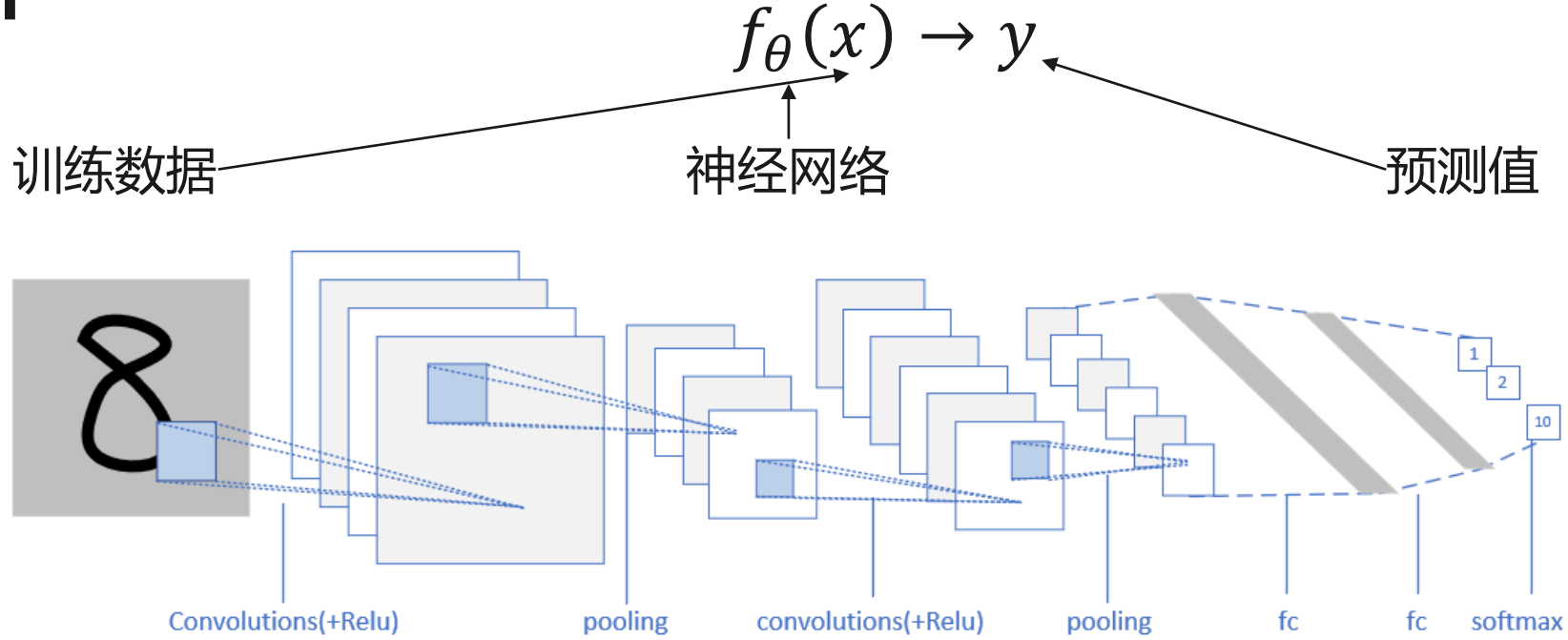
深度神经网络基础
Fundamentals of deep neural networks

Deep Neural Network

- Depends on lots of data and mostly supervised
- End-to-End learning
- Learns a deep hierarchy of abstract features with some general neural networks, such as full connected, convolutional, Recurrent, and transformer



Math



- 优化目标： $\theta^* = \underset{\theta}{\operatorname{argmin}} \sum [Loss(f_\theta(x), y)]$
- 在n个样本下近似： $\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} Loss(f_\theta(x_i), y_i)$
- 每次计算一个mini-batch：1个或者多个样本

Math (2)

- 模型参数用w 来表示： θ means a collection of w
  - W包括神经网络的weights, bias等等

- 前向计算： $\hat{y}_i = f(w, x_i)$,

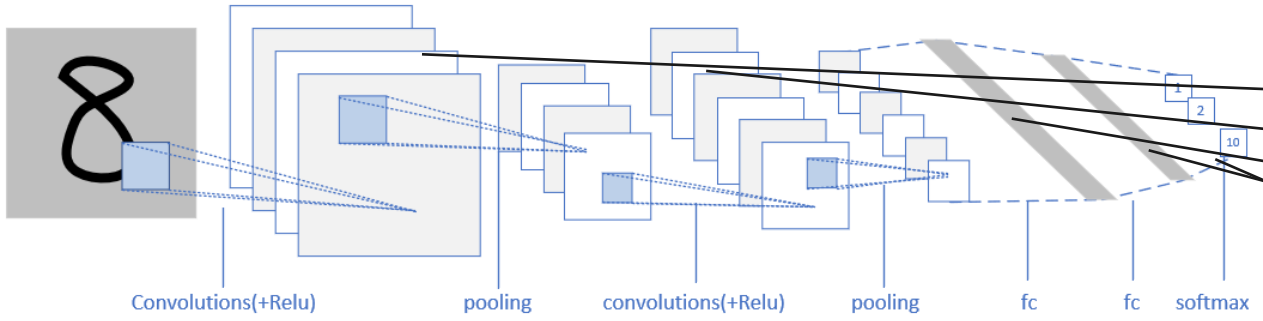
- 优化目标： $\mathrm{L}(w) = \sum_{i=1}^{n} Loss(\hat{y}_i, y_i) + \lambda \|w\|^2$

Loss function　　Regularization

- 反向传播： $\mathrm{w} \leftarrow w - \eta \nabla_w L(w)$

Learning rate　　Gradients

LeNet: PyTorch Example



（1）定义网络结构

```
5   class LeNet(nn.Module):
6       def __init__(self):
7           super(LeNet, self).__init__()
8           self.conv1 = nn.Conv2d(3, 6, 5)
9           self.conv2 = nn.Conv2d(6, 16, 5)
10          self.fc1   = nn.Linear(16*5*5, 120)
11          self.fc2   = nn.Linear(120, 84)
12          self.fc3   = nn.Linear(84, 10)
13
14      def forward(self, x):
15          out = F.relu(self.conv1(x))
16          out = F.max_pool2d(out, 2)
17          out = F.relu(self.conv2(out))
18          out = F.max_pool2d(out, 2)
19          out = out.view(out.size(0), -1)
20          out = F.relu(self.fc1(out))
21          out = F.relu(self.fc2(out))
22          out = self.fc3(out)
23          return out
```

（2）开始训练

88: mini-batch遍历样本

89: 设置计算资源

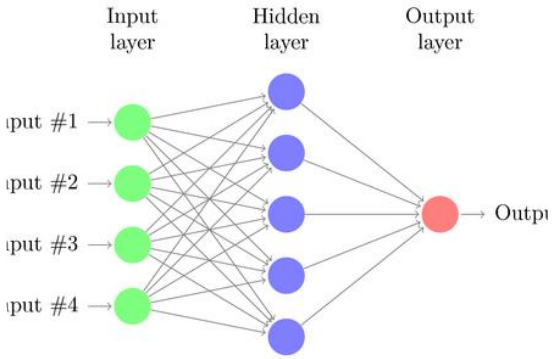
91: 前向计算

92: 计算Loss

```
88      for batch_idx, (inputs, targets) in enumerate(trainloader):
89          inputs, targets = inputs.to(device), targets.to(device)
90          optimizer.zero_grad()
91          outputs = net(inputs)
92          loss = criterion(outputs, targets)
93          loss.backward()
94          optimizer.step()
```

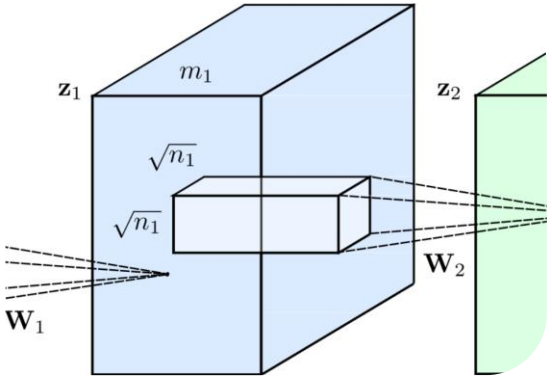
不同类型的神经网络

- 神经网络结构通常由基本的Layers所组成



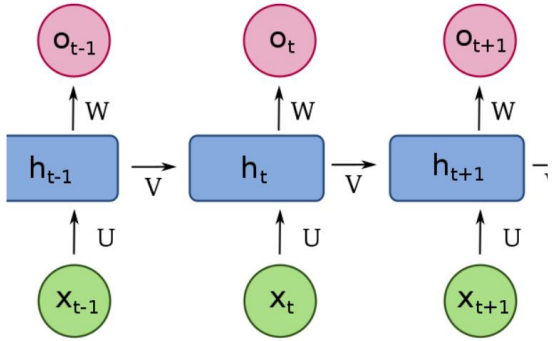
Fully connected

- 通常用作分类问题的最后几层



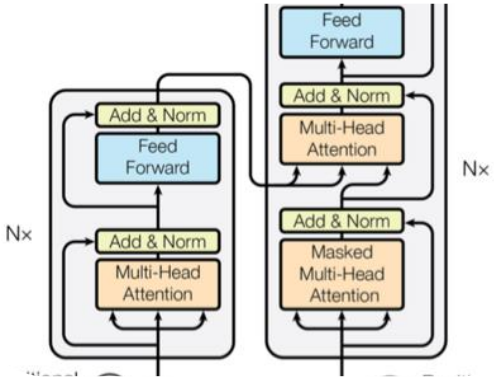
Convolutional neural network

- 通常用作图像、语音等Locality强的数据



Recurrent neural network

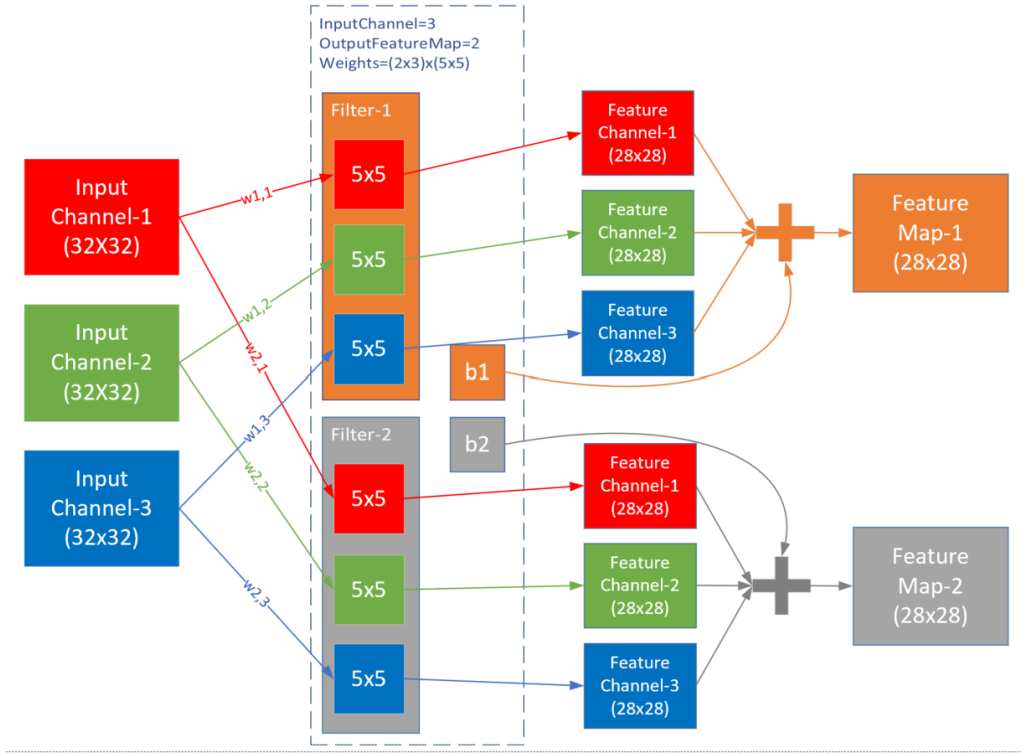
- 通常用作序列及结构化的数据，比如文本信息、知识图



Transformer neural network

- 通常用作序列数据，比如文本信息

神经网络的计算比如卷积可以抽象为张量运算 (Tensor)




```python
def conv_4d(x, weights, bias, out_h, out_w, stride=1):
batch_size = x.shape[0]
input_channel = x.shape[1]
output_channel = weights.shape[0]
filter_height = weights.shape[2]
filter_width = weights.shape[3]
rs = np.zeros((batch_size, num_output_channel, out_h, out_w))

for bs in range(batch_size):
    for oc in range(output_channel):
        rs[bs,oc] += bias[oc]
        for ic in range(input_channel):
            for i in range(out_h):
                for j in range(out_w):
                    ii = i * stride
                    jj = j * stride
                    for fh in range(filter_height):
                        for fw in range(filter_width):
                            rs[bs,oc,i,j] +=
                              x[bs,ic,fh+ii,fw+jj] * weights[oc,ic,fh,fw]
```

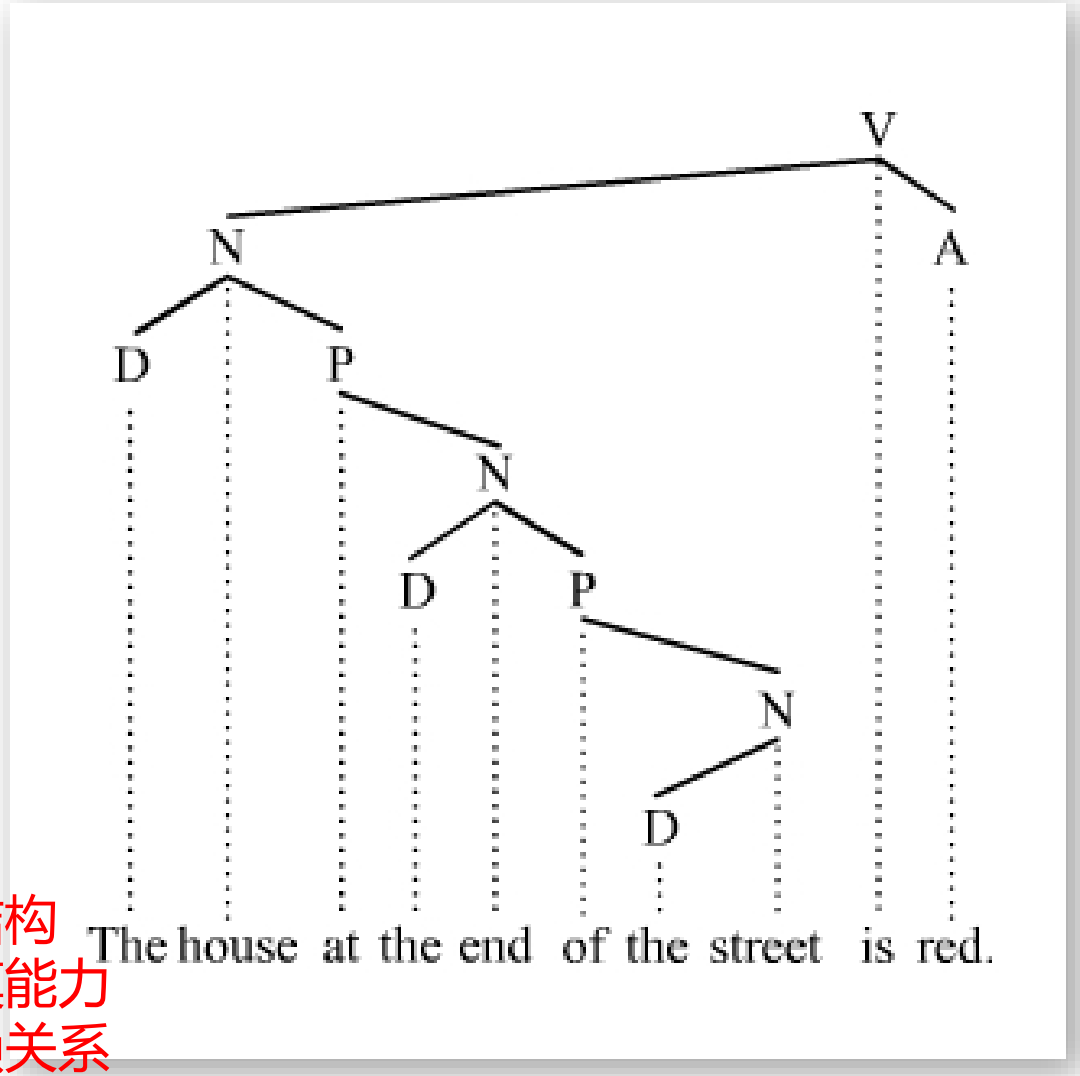
7层循环：数据，输入和输出 Channel，2维图像，2维卷积

通过数据变换，转化为更简单的矩阵运算

```python
self.col_x = img2col(x, self.FH, self.FW, self.stride, self.padding)
self.col_w = self.WB.W.reshape(self.OutC, -1).T
self.col_b = self.WB.B.reshape(-1, self.OutC)
out1 = np.dot(self.col_x, self.col_w) + self.col_b
out2 = out1.reshape(batch_size, self.OutH, self.OutW, -1)
return np.transpose(out2, axes=(0, 3, 1, 2))
```

更多新的稀疏的有结构化的神经网络

```
# A recursive TreeBank model in a dozen lines of JPL code
function sentiment(weights, tree::Tree{Word})
    # Walk the tree, accumulating embedding vecs
    function embedding(weights, tree)
        if isleaf(tree)
            token = tree.value
            # Word embedding model is used at the leaf node to map word
            # index into high-dimensional semantic word representation.
            return weights.embedding[token]
        else
            # Get semantic representations for left and right children.
            sl = embedding(weights, tree.l)
            sr = embedding(weights, tree.r)
            # A composition function is used to learn semantic
            # representation for phrase at the internal node.
            return tanh.(weights.Wl*sl + weights.Wr*sr)
    end
    # Map tree embedding to sentiment
    return softmax(weights.dense * embedding(weights, tree))
end
```



- 更多样化的结构
- 更强大的建模能力
- 更复杂的依赖关系
- 更细粒度的计算模式

小结

- 深度学习模型需要大数据、大运算量和大规模集群
  - 如何找到或者是否存在不需要如此规模需求的算法?
- 深度学习模型可以较好的抽象
  - Tensor和计算图模型有什么局限性?
- 未来的深度学习模型具有更多动态性
  - 会对系统设计有什么影响?

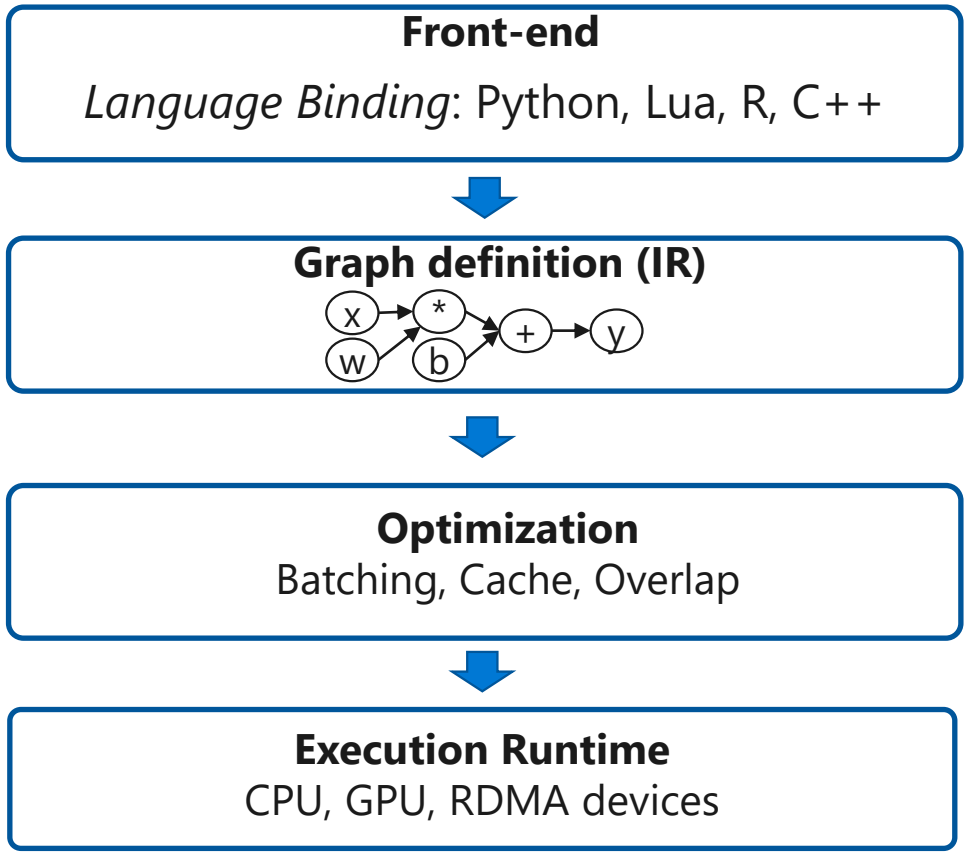
深度学习系统基础
Fundamentals of System for AI

深度学习系统简图（Deep Learning System Stack）

TensorFlow

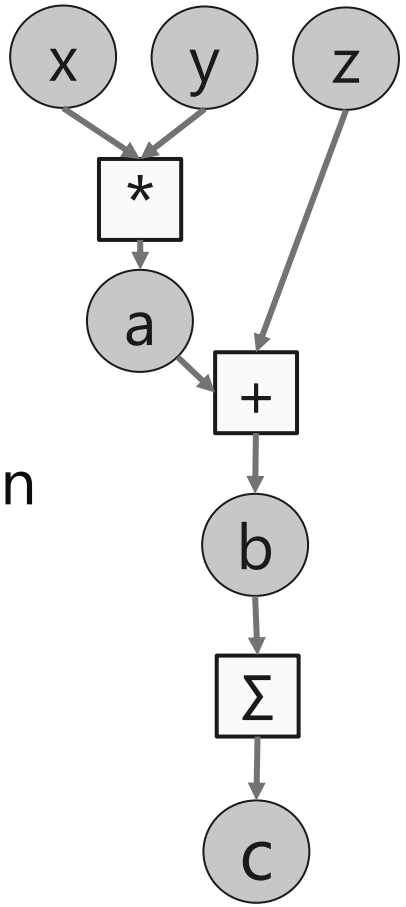
Microsoft CNTK

Caffe2



深度学习运算的表示

Data-Flow Graph (DFG)
as
Intermediate Representation

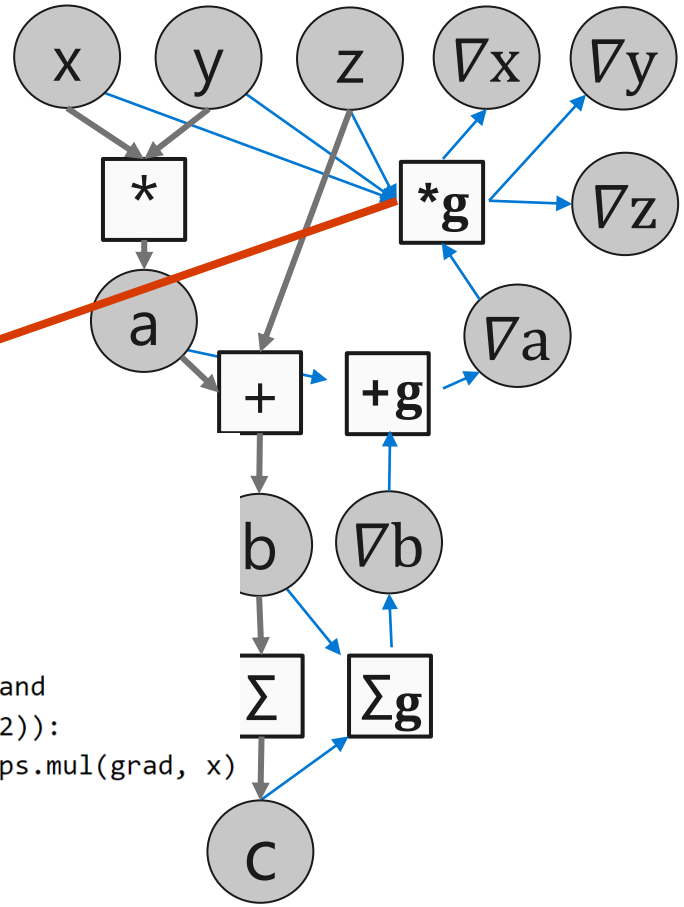


TensorFlow

```
x = tf.placeholder(tf.float32)
y = tf.placeholder(tf.float32)
z = tf.placeholder(tf.float32)

a = x * y
b = a + z
c = tf.reduce_sum(b)
```

反向求导 (Automatic Differentiation)

Add gradient back propagation to Data-Flow Graph (DFG)




```
@ops.RegisterGradient("Mul")
def _MulGrad(op, grad):
  """The gradient of scalar multiplication."""
  y = op.inputs[1]
  x = op.inputs[0]
  if (isinstance(grad, ops.Tensor) and
      _ShapesFullySpecifiedAndEqual(x, y, grad) and
      grad.dtype in (dtypes.int32, dtypes.float32)):
    return gen_math_ops.mul(grad, y), gen_math_ops.mul(grad, x)
```

TensorFlow

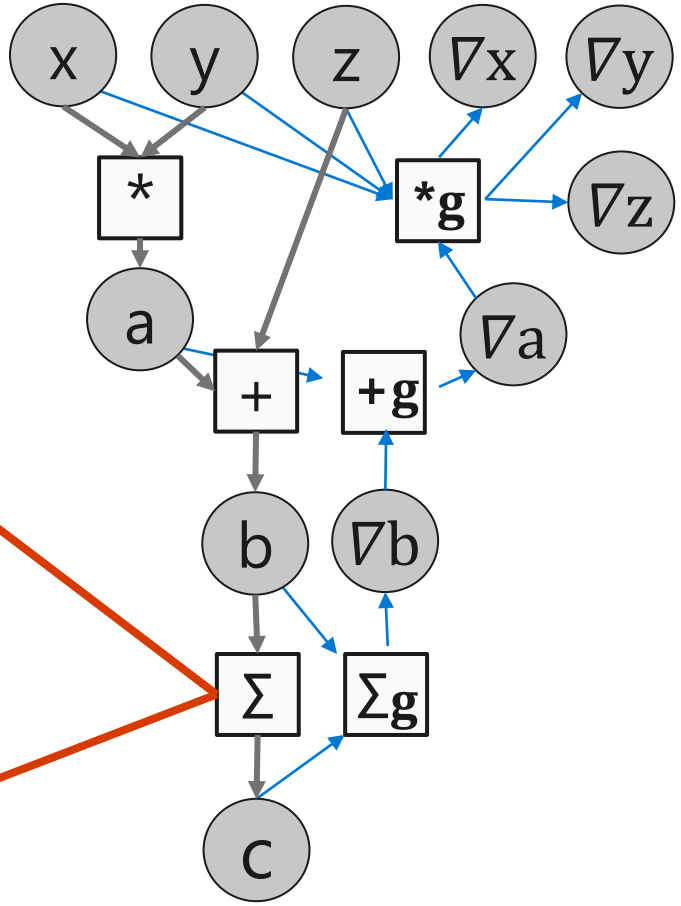
```
x = tf.placeholder(tf.float32)
y = tf.placeholder(tf.float32)
z = tf.placeholder(tf.float32)

a = x * y
b = a + z
c = tf.reduce_sum(b)

grad_x,grad_y,grad_z = tf.gradients(c, [x,y,z])

with tf.Session() as sess:
  sess.run([grad_z], feed_dict=values)
```

产生运行期代码

```
1  template <typename Device, typename OUT_T, typename IN_T,
2            typename ReductionAxes, typename Scalar>
3  struct ReduceEigenImpl<Device, OUT_T, IN_T, ReductionAxes,
4                         functor::MeanReducer<Scalar>> {
5    void operator()(const Device& d, OUT_T out, IN_T in,
6                    const ReductionAxes& reduction_axes,
7                    const functor::MeanReducer<Scalar>& reducer) {
8      static_assert(std::is_same<Scalar, typename OUT_T::Scalar>::value, "");
9      Eigen::internal::SumReducer<Scalar> sum_reducer;
10     out.device(d) = in.reduce(reduction_axes, sum_reducer) /
11                     static_cast<Scalar>(in.size() / out.size());
12   }
13 };
14
15 // T: the data type
16 // REDUCER: the reducer functor
17 // NUM_AXES: the number of axes to reduce
18 // IN_DIMS: the number of dimensions of the input tensor
19 #define DEFINE(T, REDUCER, IN_DIMS, NUM_AXES)                          \
20   template void ReduceFunctor<GPUDevice, REDUCER>::Reduce(             \
21       OpKernelContext* ctx, TTypes<T, IN_DIMS - NUM_AXES>::Tensor out, \
22       TTypes<T, IN_DIMS>::ConstTensor in,                              \
23       const Eigen::array<Index, NUM_AXES>& reduction_axes,             \
24       const REDUCER& reducer);
```

CPU code

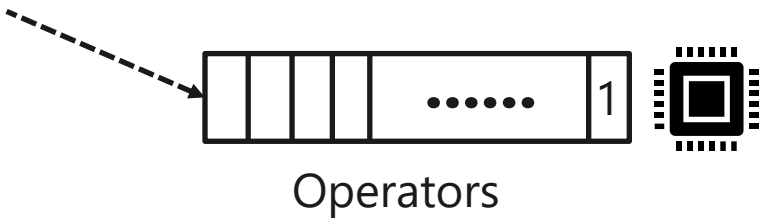
GPU code

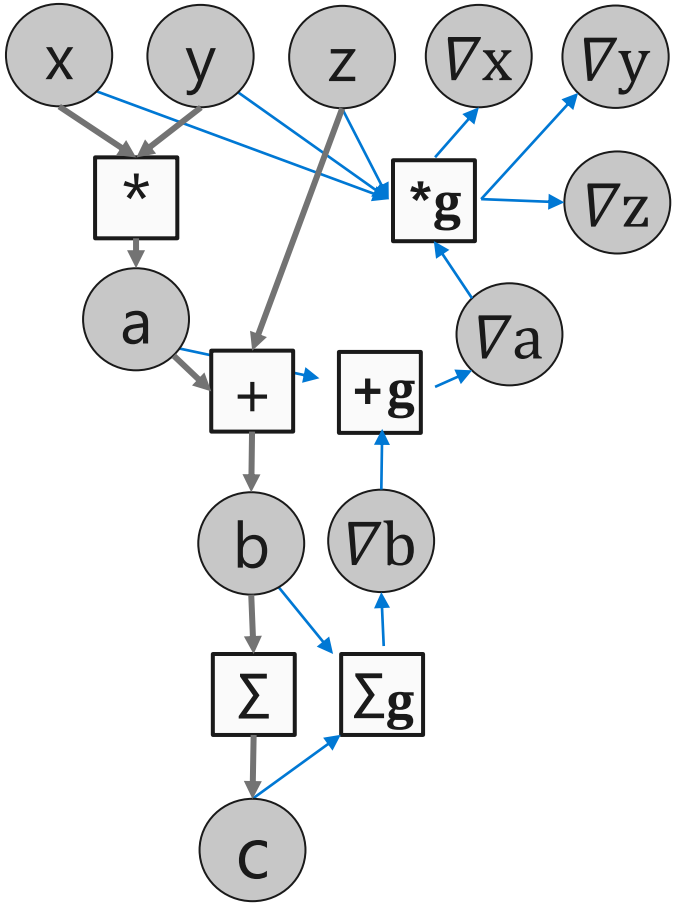


调度并运行代码

根据依赖关系，依次调度运行代码

1. Multiply(x, y) -> a
2. Add(z, a) -> b
3. ReduceSum(b) -> c
4. ReduceSum_grad(c, b) -> b_delta
5. Sum_grad(b_delta, a) -> a_delta
6. Add_grad(a_delta, z) -> z_delta
7. Multiply(a_delta, x, y) -> x_delta, y_delta





深度学习系统详图（Deep Learning System Stack）

Experience	**IDE**	Programming with: VS Code, Jupiter Notebook		
	Language	Integrated with mainstream PL: PyTorch and TensorFlow inside Python		
Frameworks	**Compiler**	Intermediate representation	Compilation	Optimization
		Basic data structure: Tensor	Lexical analysis: Token	User controlled: mini-batch
		Basic computation: DAG	Parsing: AST	Data parallelism and model parallelism
		Advance features: control flow	Semantic analysis: Symbolic AD	Loop nets analysis: pipeline parallelism, control flow
		General IRs: MLIR	Code optimization	Data flow analysis: CSP, Arithmetic, Fusion
			Code generation	Hardware dependent optimizations: matrix computation, layout
				Resource allocation and scheduler: memory, recomputation,
Architecture	**Runtimes**	Single node: CuDNN	Multimode: Parameter servers, All reducer	
		Computation cluster resource management and job scheduler		
	Hardware	Hardware accelerators: CPU/GPU/ASIC/FPGA	Network accelerators: RDMA/IB/NVLink	

编译框架与IR (Intermediate representation)

- 编译框架为整个编译过程需要的数据结构和算法设计提供支持
  - 传统框架：LLVM
  - 为深度学习设计的框架：TensorFlow Graph, TVM
  - 更加统一化的框架:MLIR
- Intermediate representation (IR)
  - The ***data structure or code*** used internally by a compiler or virtual machine to represent source code
  - ***Accurate*** and capable to representing the source code without loss of information
  - ***Independent*** of any particular source or target lanugage

编译框架与IR（2）

Gen 2 *2010-present*

Frameworks

Deep learning frameworks

MxNet
TensorFlow
CNTK
PyTorch

Language Frontend

Swift for TensorFlow

Compiler Backend

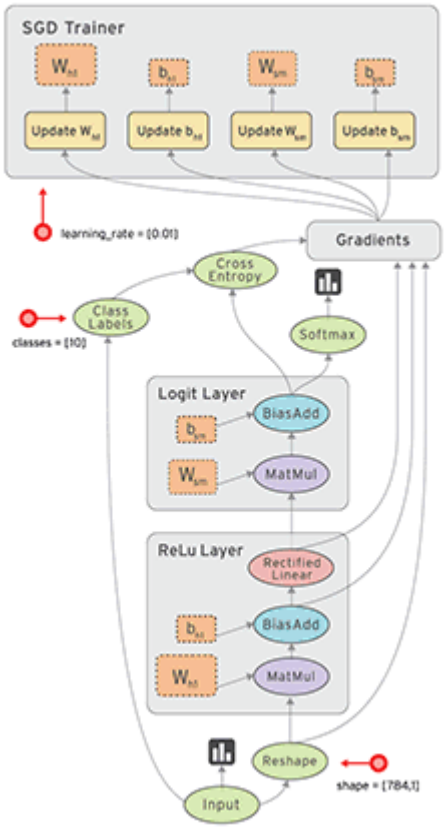
TVM
TensorFlow XLA

Hardware

AI Framework Dense matmul engine
- GPU
- FPGA

Special AI accelerators
- TPU
- GraphCore
- Other ASICs

Tensor -> Operators -> Graph




```
import "tensorflow/core/framework/graph.proto";
import "tensorflow/core/framework/op_def.proto";
import "tensorflow/core/framework/tensor_shape.proto
```

编译框架与IR （3）

Gen 3 *present-*

Frameworks

Machine Learning Language and Compiler

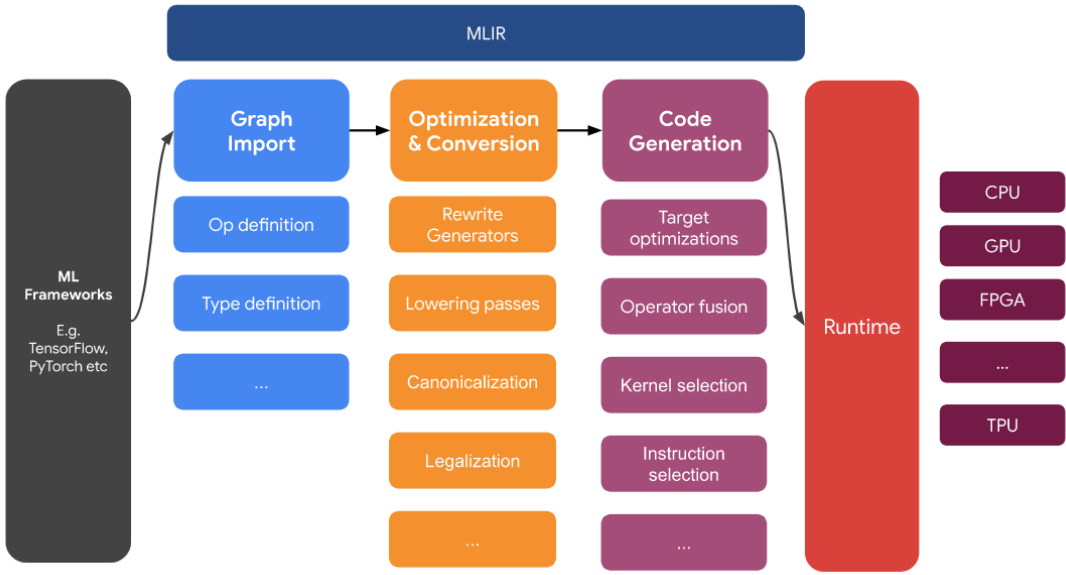
A ***Full-Featured*** Programming Language for ML: Expressive and flexible
Control flow, recursion, sparsity

Powerful ***Compiler Infrastructure***:
Code optimization, sparsity optimization, hardware targeting

Hardware

SIMD → MIMD
Sparsity Support
Control Flow and Dynamicity
Associated Memory

An example: MLIR



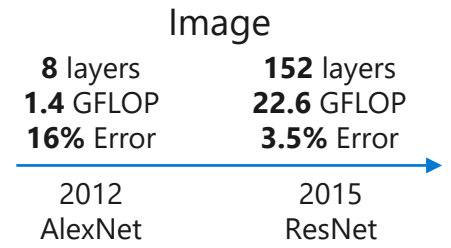

```
// Syntactically similar to LLVM:
func @testFunction(%arg0: i32) {
  %x = call @thingToCall(%arg0) : (i32) -> i32
  br ^bb1
^bb1:
  %y = addi %x, %x : i32
  return %y : i32}
```

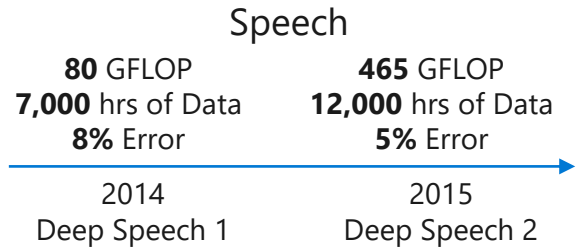
自动反向求导系统

- 反向传播需要计算前向函数的导数
- 数学上自动求导并不难，但是
  - 有些Operators并不是完全可导的，比如ReLU，需要人提供一个近似函数
  - 有些Operators人可以给出一个更优的近似实现，比如
  - 有些Operators自动求导并不是最优，比如复杂的网络结构，convolutional, Recurrent
- 现代大部分框架同时提供：
  - 1. 自动求导系统，针对常见函数的组合
  - 2. 大量内置的Operators和与之对应的反向函数，
  - 3. 提供接口让用户自己提供反向函数
- 研究方向
  - 一个更加自动化的自动求导系统，比如用传统编译方法或者强化学习的方法来寻找性能更优的反向函数

深度学习系统性能优化

- 深度学习高度依赖数据规模和模型规模
  - A rule of thumb: larger data and model bring better accuracy
  - 系统需要有能力进行大模型和大数据量训练





- 提高训练速度可以加快深度学习模型的开发速度
  - 研究人员需要不停的迭代来寻找更好的模型或者模型参数
  - Training ResNet-50 on ImageNet dataset with single GPU takes more than one week.
  - Training GNMT on WMT'14 En→Fr takes a week on 96 GPUs

- 大规模部署深度学习模型需要更快和更高效的推演速度
  - Inference performance → Serving latency
  - Many large-scale models have to be deployed with hand-coded optimizations.

系统中编译优化的挑战

- Different architectures: CNN, RNN, Transformer, …
- High computation resource requirements: model size, …
- Different goals: latency, throughput, accuracy, …

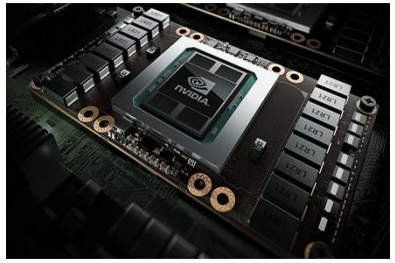
Be ***transparent*** to ***various*** user requirements

Huge optimization space at different assumptions

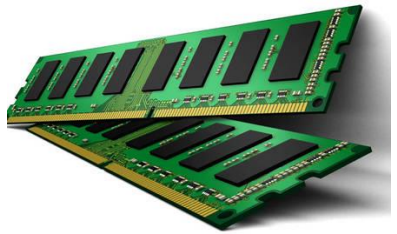
Transparently apply over ***heterogeneous*** hardware environment

Scale-out

Local Efficiency



Memory Effectiveness



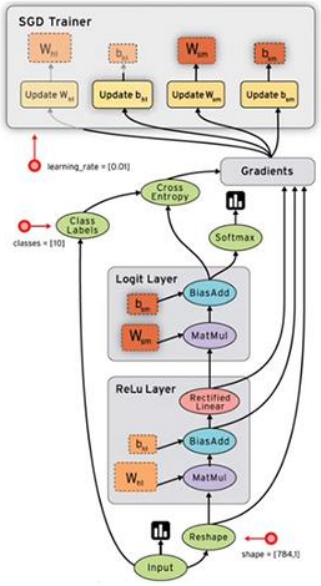
编译优化的基本原则

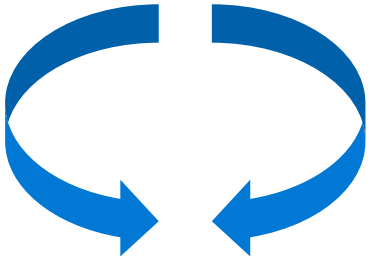
- 对用户透明
  - 减少用户的参与，并不破坏用户原本的语义
- 从各个角度增加并行性
  - 提高并行性是高效利用计算资源的基础
- 减少冗余和优化计算操作
  - 通过CSP等操作减少冗余计算，以及通过Fusion等技术优化计算算子
- 提高资源分配和调度的效率
  - 提高内存，运算和通信的资源的分配效率
  - 通过计算、IO和通信的Overlap，更好的调度计算子任务
- 通用性
  - 针对不同的计算资源进行自动优化

深度学习编译优化的复杂性

- 系统、算法和硬件必须相互结合：
  - 算法层面：模型的结构，是否可压缩、可稀疏化，batch的大小、学习算法
  - 系统层面：各个层次的并行化，去重，Overlap，调度与资源分配，硬件的特定优化
  - 硬件层面：不同精度不同性能的算子，Cache, Pre-fetch

A model to train





Parallelism	Data parallelism	Model parallelism	Pipeline parallelism
Topology	Parameter-server	All-reduce	
Scheduling	I/O	Computation	Communication
Hyper-params	Optimizer	Mini-batch	Learning rate
Optimizations	Caching	I/O overlapping	compress	Mixed precision
Hardware	SSD	CPU/GPU/FGPA	InfiniBand/NVLink

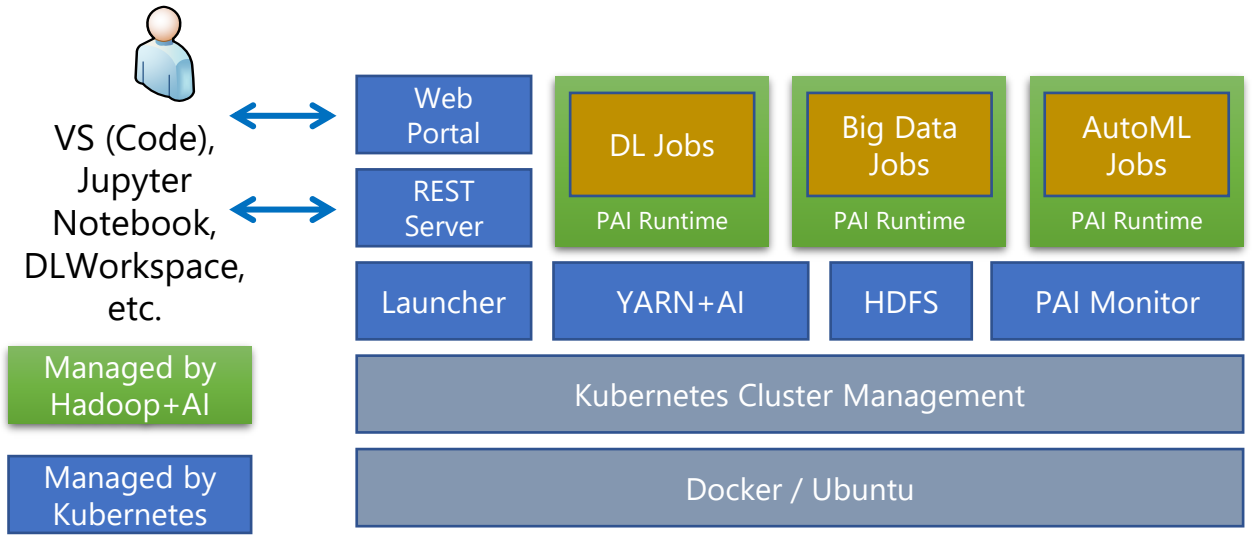
运行态和硬件

- 硬件需要提供更强大的运算、通信、IO能力
  - 运算：SIMD based hardware: GPU, ASIC, FPGA
  - 通信：高带宽低延时：RDMA, IB
  - 存储：低延迟
- 同时，硬件更需要提供好的抽象
  - 计算原语：矩阵运算➔Tensor运算➔Neural network运算
  - 通信原语：Aggregate and distributed gradient and weight ➔ Parameter server, All reducer
  - 存储原语：访问数据集：Cache, Pre-fetch, Pre-process
- 编译器自动针对不同硬件进行优化
  - 自动分配资源：memory, GPU threads, …
  - 自动推演矩阵算法参数：TVM with ML

分布式的运行多个训练任务

- 研究人员需要运行大量的训练任务
  - 需要在不同的数据集上进行训练
  - 寻找最佳的模型结构或者模型参数
  - 大量的研究人员需要同时进行实验
- 要求一个能够大规模运行训练任务和管理训练任务的平台
  - 不同于HPC: 更多样化的计算任务
  - 不同于大数据系统：不仅需要数据，而且需要大量的计算资源
  - 不同于传统Cloud IaaS: 需要更高效一体化优化的计算服务
- 研究设计新的分布式系统成为新的挑战
  - Microsoft：Azure ML, OpenPAI
  - Google: Cloud TPU
  - Community: KubeFlow

一个基本的训练平台示例：OpenPAI



https://github.com/Microsoft/pai

- **提供接口和工具供用户提交训练任务**
- **提供文件管理系统管理训练所需的大数据**
- **提供运行环境供Frameworks访问**
  - 计算资源：GPU/FPGA/ASIC
  - 网络资源： IB/RDMA
  - 存储资源：HDFS/NFS
- **高效的调度算法**
  - 分配异构计算资源
  - 错误恢复和容错管理
- **日志、性能监控系统**
- **用户、安全管理**

小结

- 框架、语言、编译、优化、调度等系统问题需要为深度学习重新思考与设计
  - 哪些最先得到重视，哪些更具长远影响？
- 性能优化必须贯穿整个系统栈
  - 为什么性能优化对深度学习系统如此重要？
- 后面的课程将依次介绍各个重要组成部分
  - 相辅相成

课后作业

· 推荐补充阅读材料

Lab 1 (for week 1, 2)

- Purpose
  - A simple throughout end-to-end AI example, from a system perspective
  - Understand the systems from debugger info and system logs
- Get ready
  - https://github.com/microsoft/ai-edu/ai-system/labs/1

Backups

Piling up hardware is not sustainable: energy-efficiency wall