编译论坛

深度学习框架发展 III

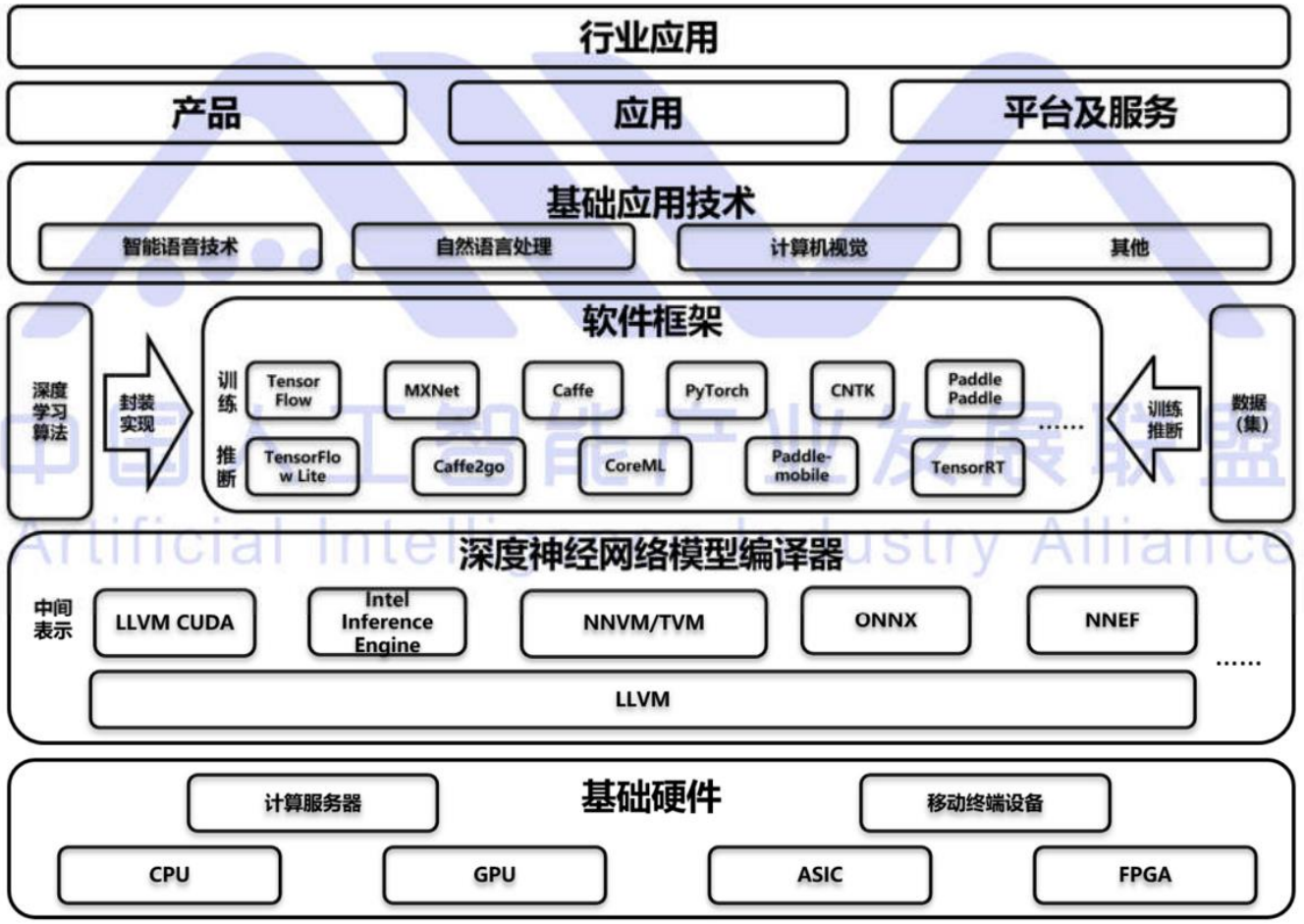
嘉宾：王磊

目录

背景——软件框架

软件框架（Software Framework）

通常指为了实现某个业界标准或者完成特定基本任务的软件组件规范，也指为了实现某个软件规范时，提供规范所要求的基础功能的软件产品。

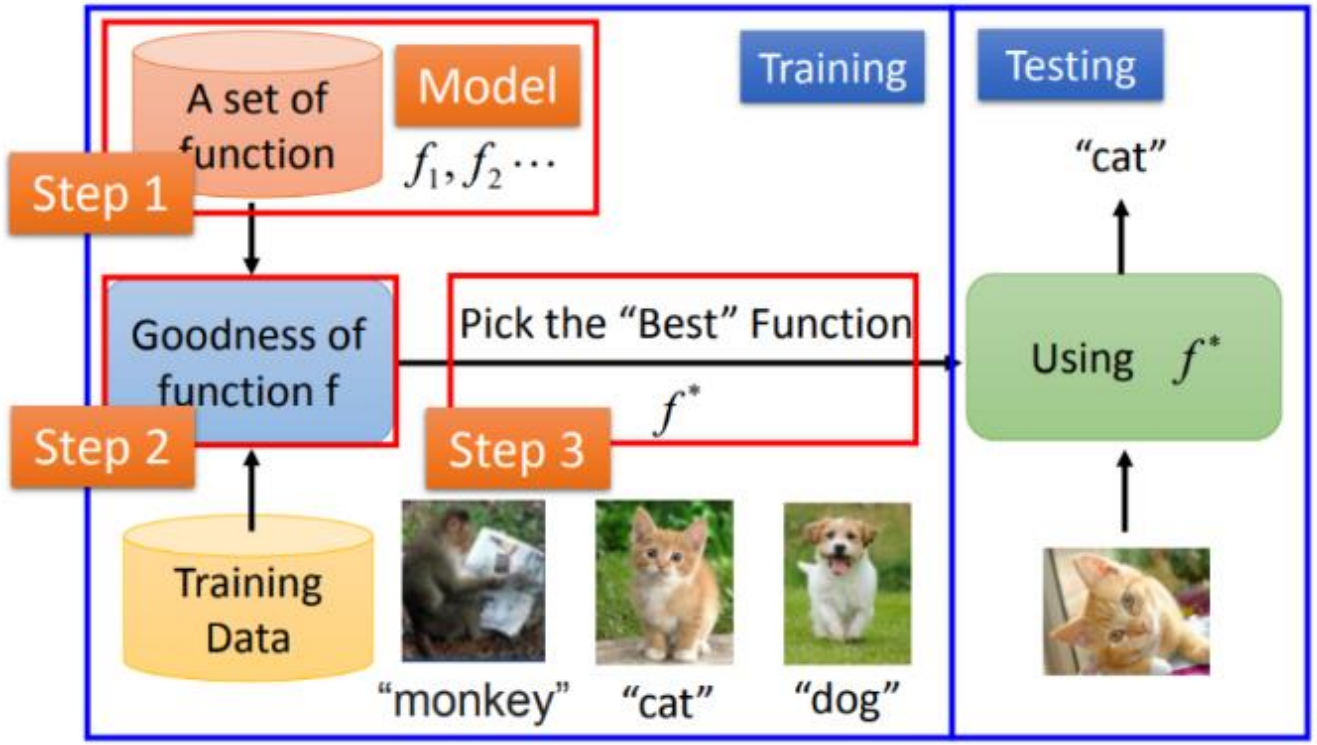


先进编译实验室
Advanced Compiler

背景——机器学习框架

机器学习框架（ML Framework）

机器学习是根据提供的资料，通过使用多种算法寻找function，来自动化各种业务操作并使机器模仿人的行为，但这些算法仅能被ML领域专家所理解，因此，机器学习框架简化了这些复杂的算法，以便可以轻松地在企业应用程序中实现ML。



先进编译实验室
Advanced Compiler

背景——机器学习框架

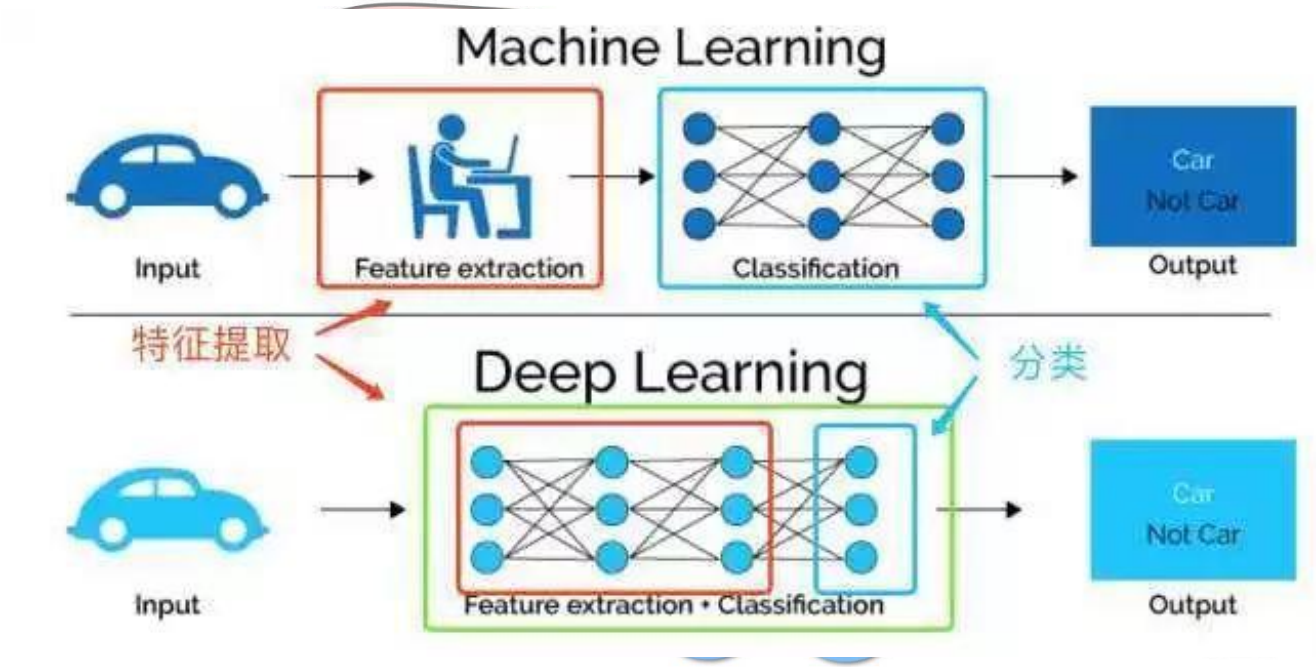
机器学习框架（ML Framework)

代表性框架Scikit-learn、Apache Mahout、SystemDS、Spark Mllib等

机器学习框架 VS 深度学习框架

机器学习框架包含了深度学习框架

- 对于数据处理的方式不同
- 模型封装的抽象化程度不同
- 针对的群体和项目不同



背景——深度学习训练框架与推理框架

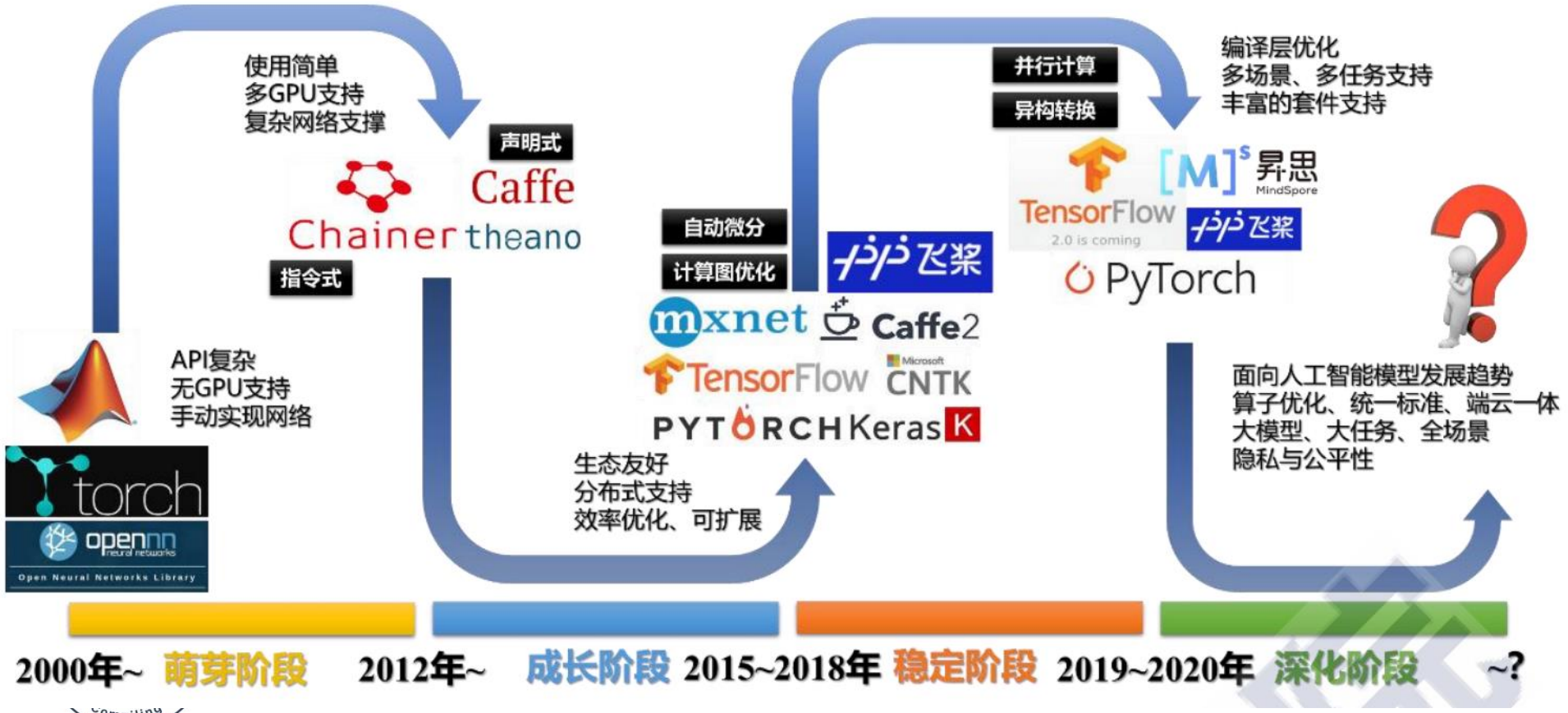
训练、推理、部署

云端训练框架：主要完成面向海量数据的模型训练任务，对算力要求最高，实际应用中需要采用分布式计算等技术，同时对于工业级模型及稳定性也有特殊要求。

云端推理框架：完成训练模型的优化、云端部署及推断计算等工作，对于效率及并发性等具有特殊要求

端侧推理框架：主要完成训练模型在终端的部署及计算，由于终端功耗、功能、芯片等众多限制，终端推断框架的性能、能耗及自身优化需满足多种限制要求

发展历程


使用简单
多GPU支持
复杂网络支撑
声明式
Caffe
Chainer
theano
指令式
API复杂
无GPU支持
手动实现网络
torch
opennn
Open Neural Networks Library
自动微分
计算图优化
mxnet
Caffe2
TensorFlow
CNTK
PYTORCH
Keras
生态友好
分布式支持
效率优化、可扩展
并行计算
异构转换
编译层优化
多场景、多任务支持
丰富的套件支持
TensorFlow
2.0 is coming
昇思
MindSpore
PyTorch
面向人工智能模型发展趋势
算子优化、统一标准、端云一体
大模型、大任务、全场景
隐私与公平性
2000年~ 萌芽阶段
2012年~ 成长阶段
2015~2018年 稳定阶段
2019~2020年 深化阶段
~?

```lua
require 'nn';
net = nn.Sequential()
net:add(nn.SpatialConvolution(1, 6, 5, 5)) - 1个图像输
net:add(nn.SpatialMaxPooling(2,2,2,2))     -- 1个2x2
net:add(nn.SpatialConvolution(6, 16, 5, 5))
net:add(nn.SpatialMaxPooling(2,2,2,2))
net:add(nn.View(16*5*5))                   -- 将16x5x
net:add(nn.Linear(16*5*5, 120))            -- 全连接层
net:add(nn.Linear(120, 84))
net:add(nn.Linear(84, 10))                 -- 10是网络

print(net:__tostring());
mlp=nn.Parallel(2,1);     -- iterate over dimension 2
mlp:add(nn.Linear(10,3)); -- apply to first slice
mlp:add(nn.Linear(10,2))  -- apply to first second sl
x = torch.randn(10,2)
print(x)
print(mlp:forward(x))
criterion = nn.ClassNLLCriterion() -- 多分类的负对数似
loss = criterion:forward(output, 3) --
gradients = criterion:backward(output, 3)
gradInput = net:backward(input, gradients)
parameters, gradParameters = net:getParameters()
```

```lua
    d(sub2ind(size(d), D(k), 1)) = 1;

-- 定义计算损失和dloss/dx的闭包。
feval = function(x)
    -- 重置梯度
    gradParameters:zero()
    -- 1. 计算每个数据点的输出（对数概率）
    local output = net:forward(input)
    -- 2. 根据真实标签，计算这些输出的损失
    local loss = criterion:forward(output, 3)
    -- 3. 计算损失对模型输出的导数
    local dloss_doutput = criterion:backward(output, 3)
    -- 4. 使用梯度更新权重
    net:backward(input, dloss_doutput)

    -- 返回 损失 和 相对于权重的损失梯度
    -- loss, (gradient of loss with respect to the weights)
    return loss, gradParameters
end
-- 定义 SGD 参数.
sgd_params = {
    learningRate = 1e-2,
    learningRateDecay = 1e-4,
    weightDecay = 0,
    momentum = 0
}
-- 训练多个批次数据
epochs = 1e2
losses = {}
for i = 1,epochs do
    -- SGD优化的一个步骤（梯度下降法）
    _,local_loss = optim.sgd(feval, parameters, sgd_params)
    -- 累计误差
    losses[#losses + 1] = local_loss[1]
end
print(losses[1])
print(losses[#losses])
print(torch.exp(net:forward(input)))
```

```
layer {
  name: "mnist"
  type: "Data"
  transform_param {
    scale: 0.00390625
  }
  data_param {
    source: "mnist_train_lmdb"
    backend: LMDB
    batch_size: 64
  }
  top: "data"
  top: "label"
}
layer {
  name: "pool1"
  type: "Pooling"
  pooling_param {
    kernel_size: 2
    stride: 2
    pool: MAX
  }
  bottom: "conv1"
  top: "pool1"
}
.....
```

```
# The train/test net protocol buffer definition
net: "examples/mnist/lenet_train_test.prototxt"
# test_iter specifies how many forward passes the test should carry out.
# In the case of MNIST, we have test batch size 100 and 100 test iterations,
# covering the full 10,000 testing images.
test_iter: 100
# Carry out testing every 500 training iterations.
test_interval: 500
# The base learning rate, momentum and the weight decay of the network.
base_lr: 0.01
momentum: 0.9
weight_decay: 0.0005
# The learning rate policy
lr_policy: "inv"
gamma: 0.0001
power: 0.75
# Display every 100 iterations
display: 100
# The maximum number of iterations
max_iter: 10000
# snapshot intermediate results
snapshot: 5000
snapshot_prefix: "examples/mnist/lenet"
# solver mode: CPU or GPU
solver_mode: GPU
```

发展历程——铁器时代

2015~2018年：效率优化、分布式支持、蓬勃发展

Caffe2

PYTORCH

TensorFlow

Keras

Microsoft CNTK

mxnet

飞桨 PaddlePaddle

发展历程——罗马时代

2018~2019：大模型训练、并行计算、可用性提升

当前发展——工业时代

2020+：多场景多任务支持、编译层优化、丰富的套件支持

- 基于编译器的算子优化
- 编程接口优化
- 自动分布式训练
- 全场景、异构设备支持
- 动态图静态图融合
- 训练推理一体化
- 小样本训练支持
- AI+Science
- 性能领先的模型库

飞桨 PaddlePaddle

昇思 MindSpore

ONEFLOW

天元 MegEngine

Jittor计图

参考资料

AI框架发展白皮书（2022年）

https://syncedreview.com/2020/12/14/a-brief-history-of-deep-learning-frameworks/