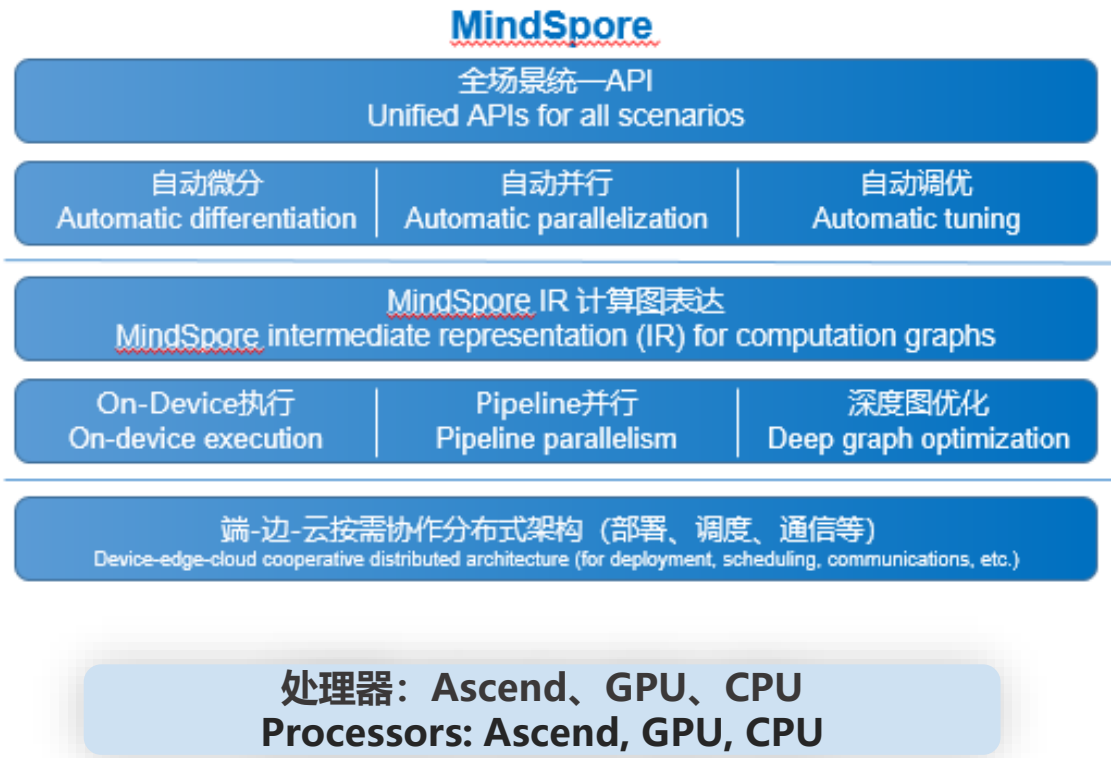
深入浅出介绍全场景AI框架MindSpore

目录

MindSpore总体架构

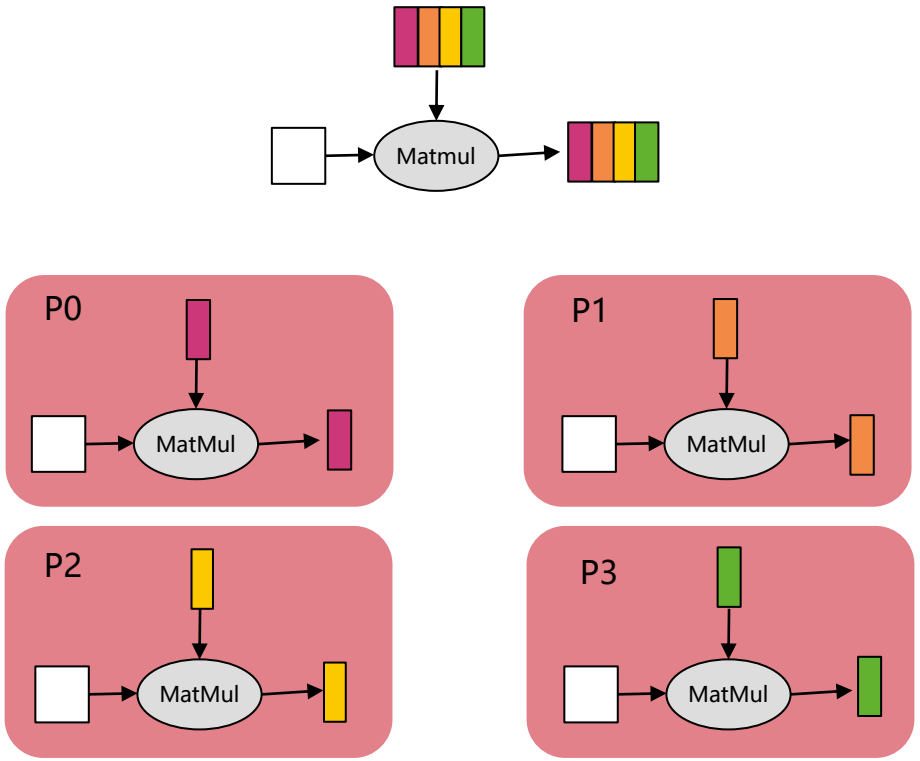


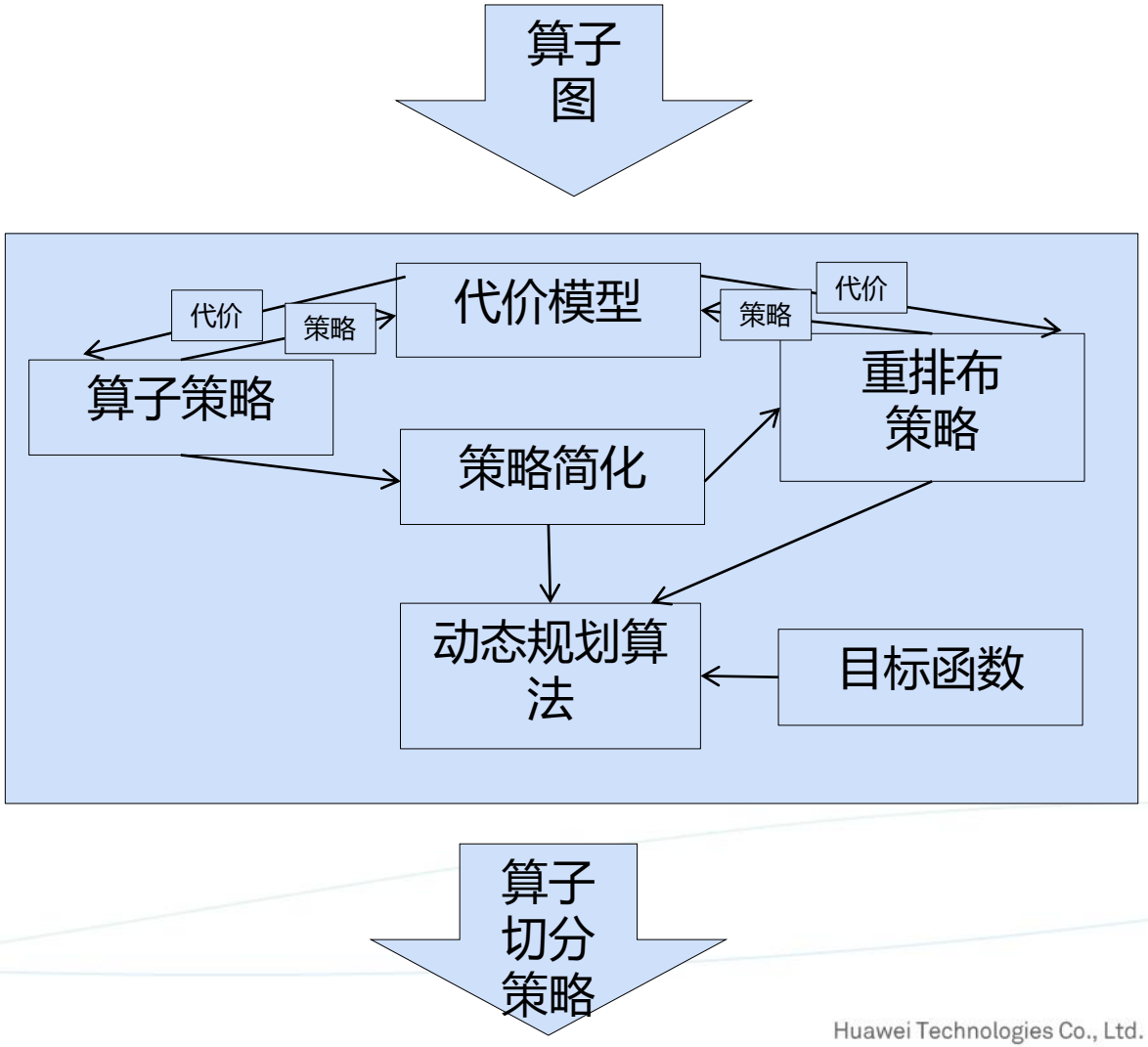
处理器：Ascend、GPU、CPU
Processors: Ascend, GPU, CPU

- **易用性：统一编译**
  - 网络和算子统一表达和编译
  - 自动并行：复杂网络自动并行
  - 自动微分：E2E的自动微分(网络和算子)
  - 精准可视调优
- **性能：全栈编译加速，软硬件协同**
  - MindSporeIR实现图和算子统一融合优化
  - 与Ascend软硬件协同，深度图优化
- **全场景：端侧推理和云上训练协同**
  - 轻量化和时延（IoT设备）
  - 根据设备信息自适应模型生成
  - 训练时量化，更好的量化精度，更小的计算开销

自动并行

让一个大规模的网络可以在一个大规模的数据集上高效训练





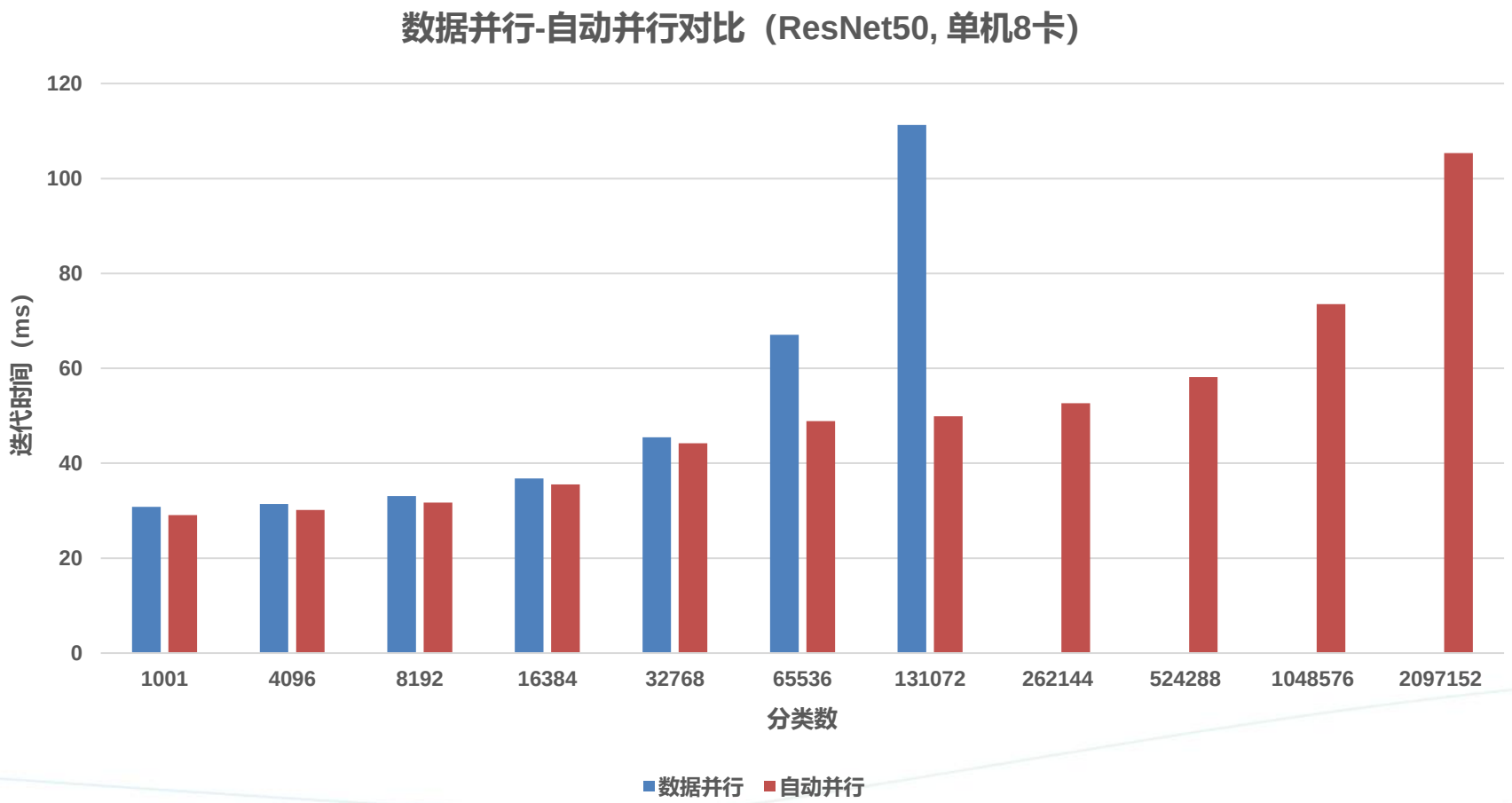
自动并行

```
class DenseNet(nn.Cell):
    def __init__(self):
        super(DenseMutMulNet, self).__init__()
        self.embedding_weight = Parameter(Tensor(12288, 128))
        self.embedding = P.MatMul()
        self.fc1 = nn.Dense(128, 768, activation='relu')
        self.fc2 = nn.Dense(128, 768, activation='relu')
        self.fc3 = nn.Dense(128, 768, activation='relu')
        self.transpose = P.Transpose()
        self.matmul1 = P.MatMul()
        self.matmul2 = P.MatMul()

    def construct(self, x):
        x = self.embedding(x, self.embedding_weight)
        q = self.fc1(x)
        k = self.fc2(x)
        v = self.fc3(x)
        k = self.transpose(k, (1, 0))
        c = self.matmul1(q, k)
        s = self.matmul2(c, v)
        return s

def train_step():
    context.set_auto_parallel_context(parallel_mode=ParallelMode.AUTO_PARALLEL)
    input = Tensor(np.ones([32, 128]).astype(np.float32))
    label = Tensor(np.zeros([32, 768]).astype(np.float32))
    net = DenseNet()
    model = Model(net, opt, loss)
    model.train(input, label)
```

单机模型代码

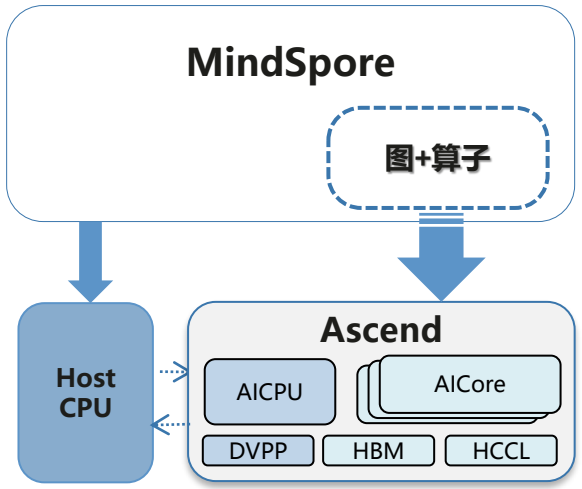
Auto Parallel

自动并行


数据并行-自动并行对比（ResNet50, 单机8卡）
迭代时间（ms）
120
100
80
60
40
20
0
1001
4096
8192
16384
32768
65536
131072
262144
524288
1048576
2097152
分类数
数据并行
自动并行

适配昇腾芯片

MindSpore协同昇腾芯片全栈优化，整图卸载执行，充分发挥昇腾大



- **On-Device执行**：整图下沉到Device执行，减少host-device交互开销
- **深度图优化:** 包括整图的格式转换消除、类型转换消除、算子融合
- **昇腾芯片亲和的高性能数据处理pipeline**：数据增强，全局shuffle

单卡训练数据，预计集群性能更高

网络	验收规格	与标准TF+GPU对比（精度和性能） 单卡对比（910 vs V100）
ResNet50	16卡小集群，猫狗、花卉、手势4种小数据集	性能：+83%
ReID	256卡集群，XX万ID，XX亿图片数据集	性能：+75%
Faster-RCNN	单机8卡，118K数据集	性能：+65%
Bert-Nezha	80卡集群，400G数据集，3亿参数	性能：+72.5%

过程数据，未进行针对性的性能优化

Graph Mode 与 PyNative Mode

调试灵活　　**运行高效**

一套神经网络训练代码

单算子/子图执行

一条语句

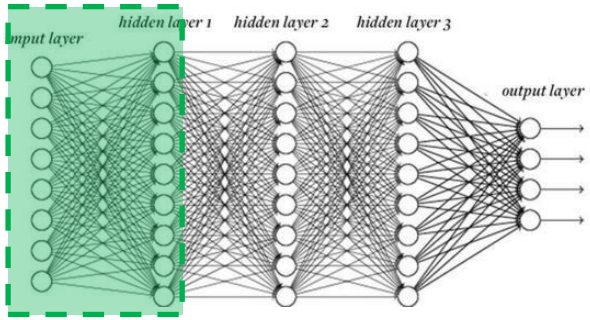
context.set_context(mode=context.xxx_MODE)

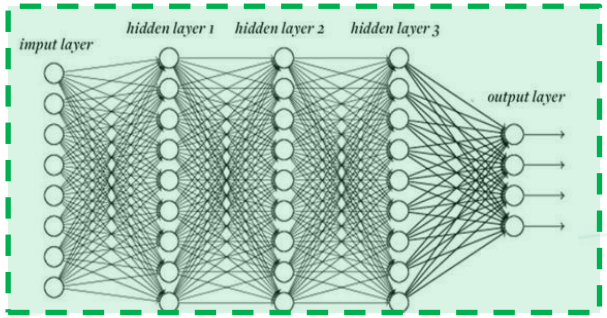
整图编译执行

动态图：灵活的开发调试

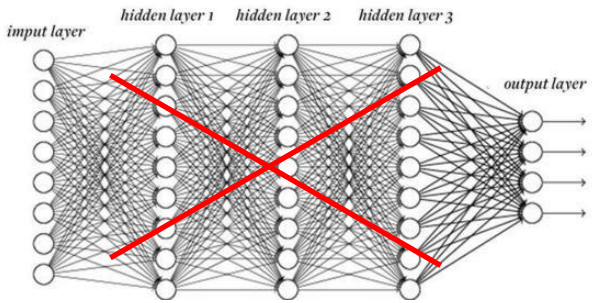
静态图：高效的图编译优化，性能高

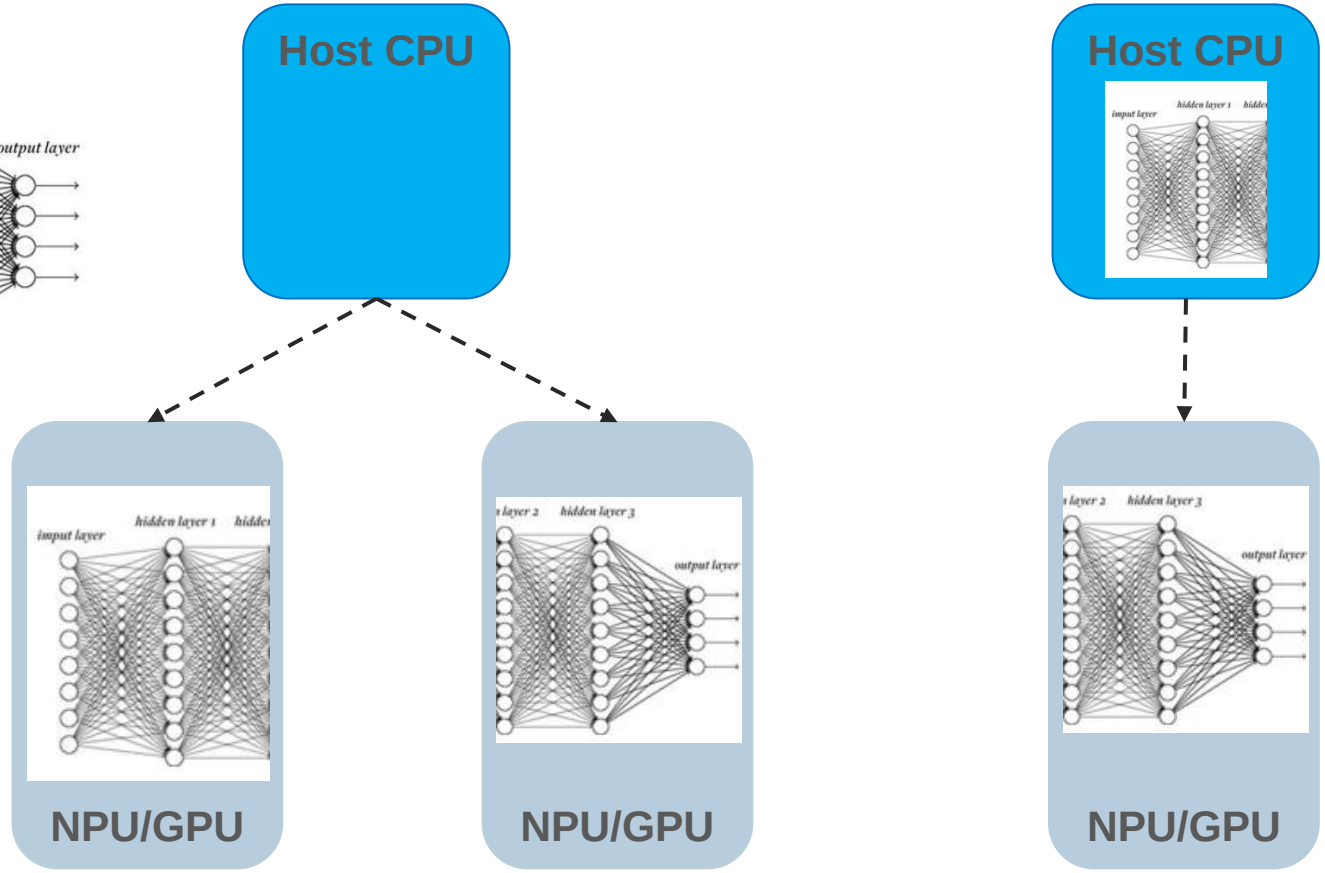
1行代码完成PDB调试与运行切换





Host+Device混合训练



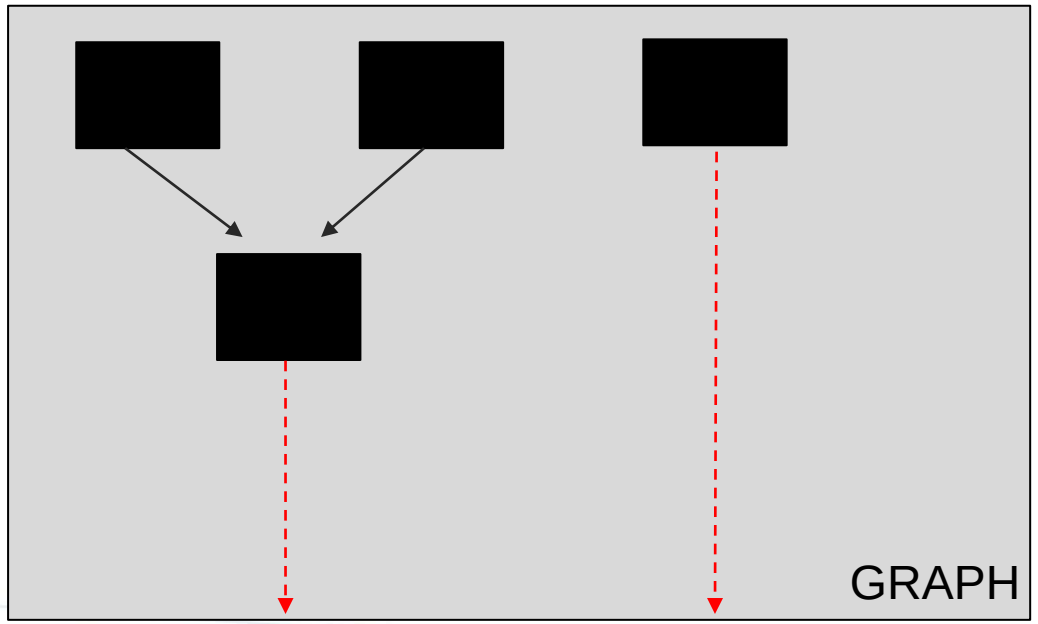


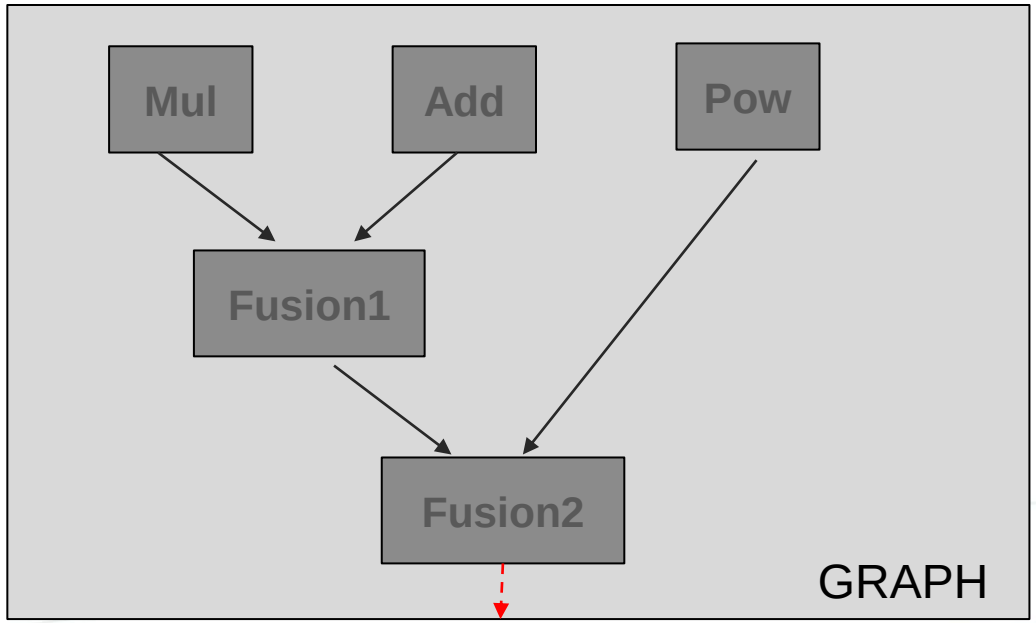
- 同时发挥了主机端内存大和加速器端计算快的优势
- 灵活应对各类场景

图算融合

`context.set_context(enable_graph_kernel=True)` → **开启图算融合**

- 将算子与图层的表达进行统一
- 对原有计算逻辑进行拆分、重组、融合等操作，以减少算子执行间隙的开销并且提升设备计算资源利用率
- 在后端由AKG(Auto Kernel Generator)自动生成高性能的融合算子





感知量化训练

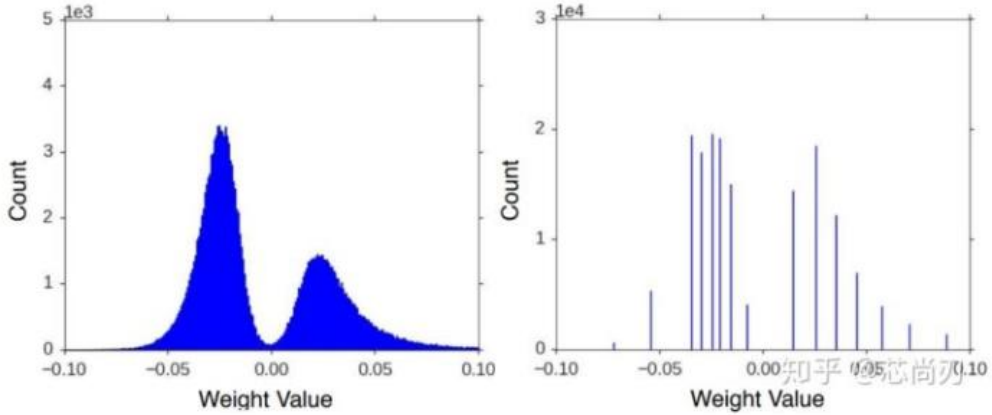
模型量化即以较低的推理精度损失将连续取值的浮点型模型权重或流经模型的张量数据定点近似为有限多个离散值的过程，达到减少模型尺寸大小、减少模型内存消耗及加快模型推理速度等目标。

- 训练后量化
- 感知量化训练

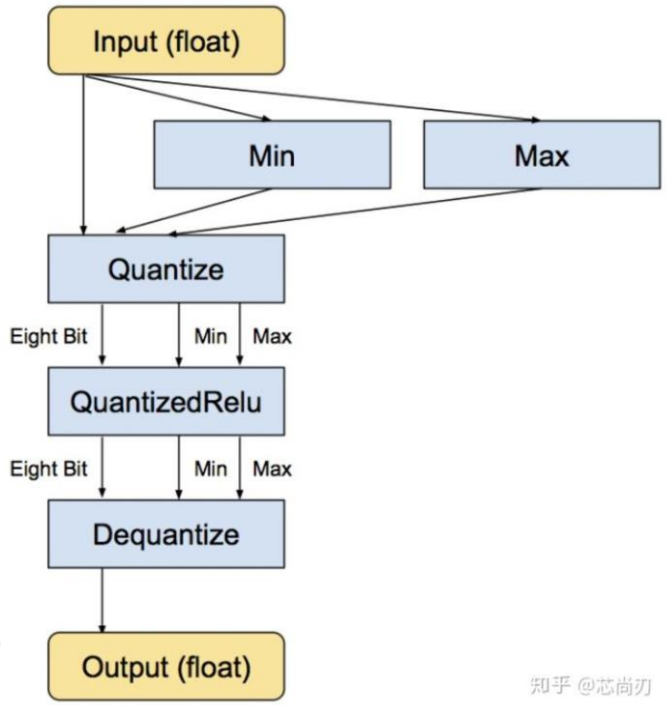
$$Q = \frac{R}{S} + Z \qquad S = \frac{R_{max} - R_{min}}{Q_{max} - Q_{min}}$$

$$R = (Q - Z) * S \qquad Z = Q_{max} - R_{max} \operatorname{div} S$$

```
from mindspore.train.quant import quant as qat

net = qat.convert_quant_network(net, quant_delay=0, bn_fold=False,
freeze_bn=10000, weight_bits=8, act_bits=8)
```



定点量化近似表示[1]



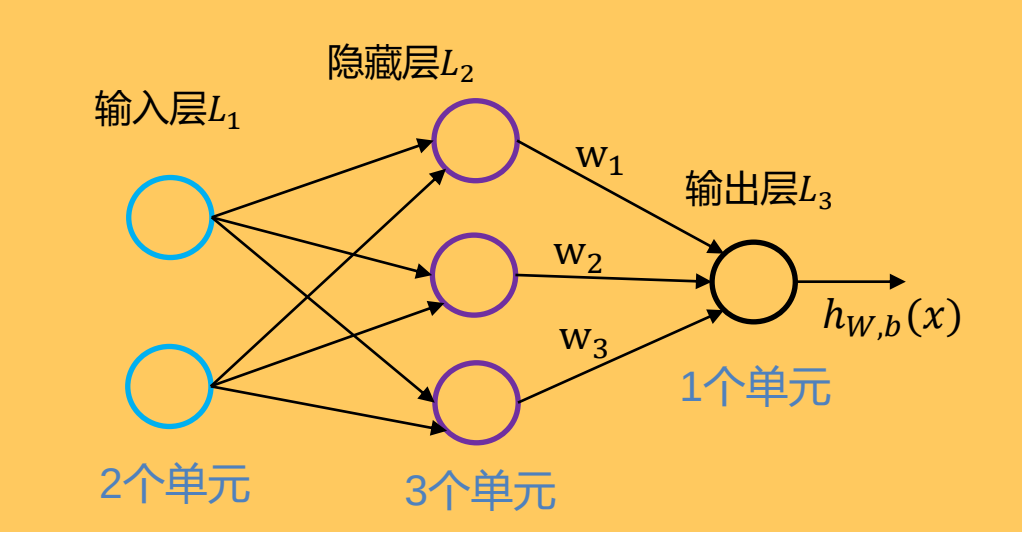
ReLU感知量化训练过程[1]

[1] https://zhuanlan.zhihu.com/p/79744430

LeNet介绍—全连接层

神经网络

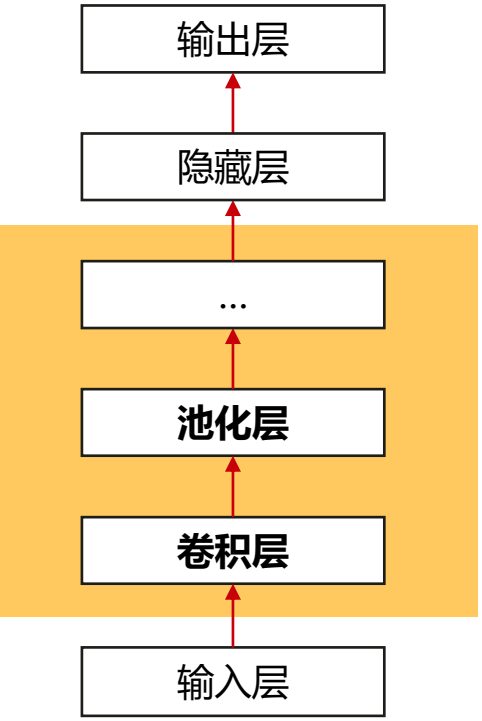
- 人工神经网络，Artificial neural network (ANN)
- 一种用于信息处理的数学模型
- 通常有两个及以上隐藏层



人工神经网络

深度学习vs 神经网络

- 深度学习是使用深度神经网络的机器学习方法
- **CNN**
  - ✓ 最著名的深度神经网络
  - ✓ 卷积层
  - ✓ 池化层



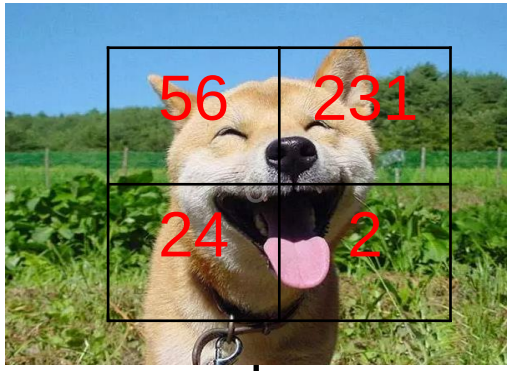
卷积神经网络

LeNet介绍—全连接层

$$z = XW^T + b$$

0.2	-0.5	0.1	2.0
1.5	1.3	2.1	0.0
0	0.25	0.2	-0.3

W



56	231	24	2
5	31	31	12

展成行向量

0.2	1.5	0
-0.5	1.3	0.25
0.1	2.1	0.2
2.0	0.0	-0.3

+

1.1
3.2
-1.2

-96.8	437.9	61.95
150.2	55.3	98.2

Cat score　Dog score　Ship score

LeNet介绍—全连接层

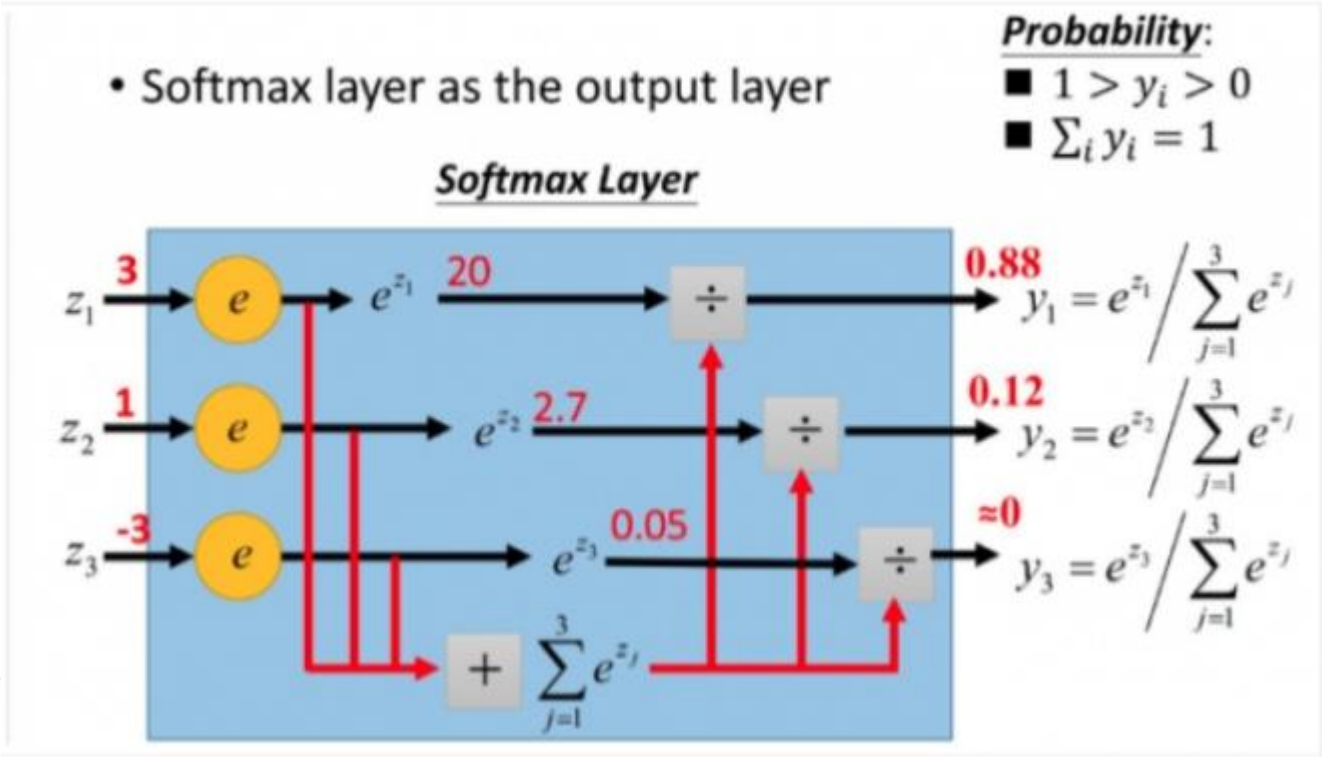
-96.8	437.9	61.95
150.2	55.3	98.2

$$S_i = \frac{e^i}{\sum_j e^j}$$

0.01	0.85	0.14
0.66	0.15	0.14

Z

P



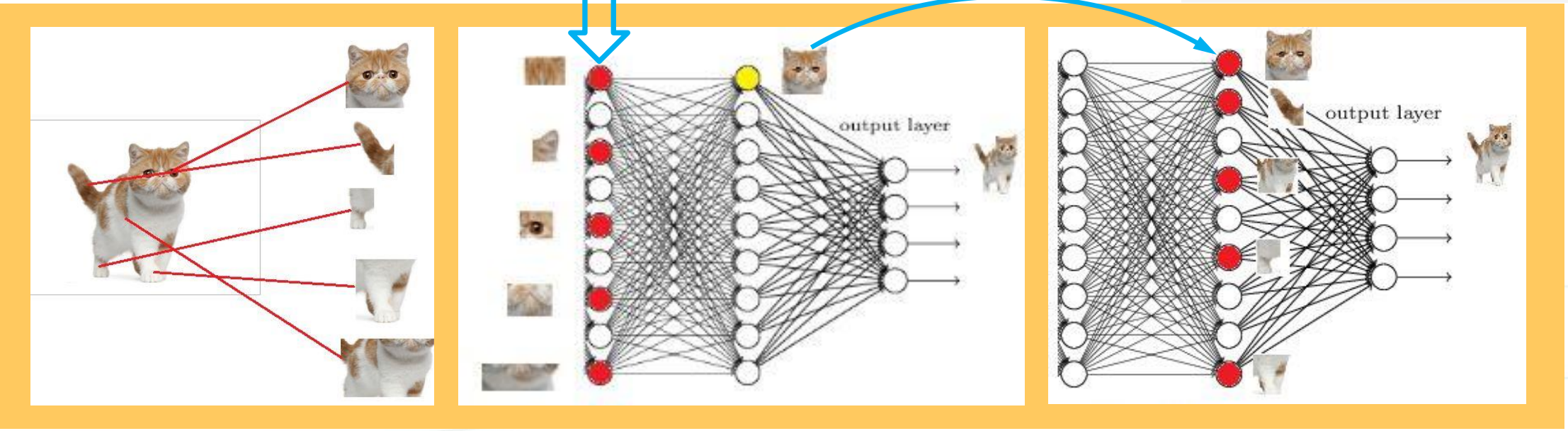
LeNet介绍—全连接层

1. 将局部特征通过权值矩阵组装成完整图像，类似分类器
2. 本质是特征空间的线性变换
3. 用到所有的局部特征，所以叫全连接
4. 常出现在最后几层
5. 一般采用ReLU激活函数

```
class Net(nn.Cell):
    def __init__(self,
                 input_channels,
                 output_channels,
                 weight='normal',
                 bias='zeros',
                 activation=''):
        super(Net, self).__init__()
        self.dense = nn.Dense(input_channels,
                              output_channels,
                              weight,
                              bias,
                              activation)
    def construct(self, input):
        return self.dense(input)

weight = Tensor(np.random.randint(0, 255, [8, 64]), ms.float32)
bias = Tensor(np.random.randint(0, 255, [8]), ms.float32)
net = Net(64, 8, weight=weight, bias=bias)
input = Tensor(np.random.randint(0, 255, [128, 64]), ms.float32)
```

MindSpore 全连接层

- 红色神经元表示特征被找到（或激活）
- 同一层其他神经元，标识特征不明显或没找到



https://zhuanlan.zhihu.com/p/33841176

LeNet介绍—卷积层

时域卷积等于频域相乘

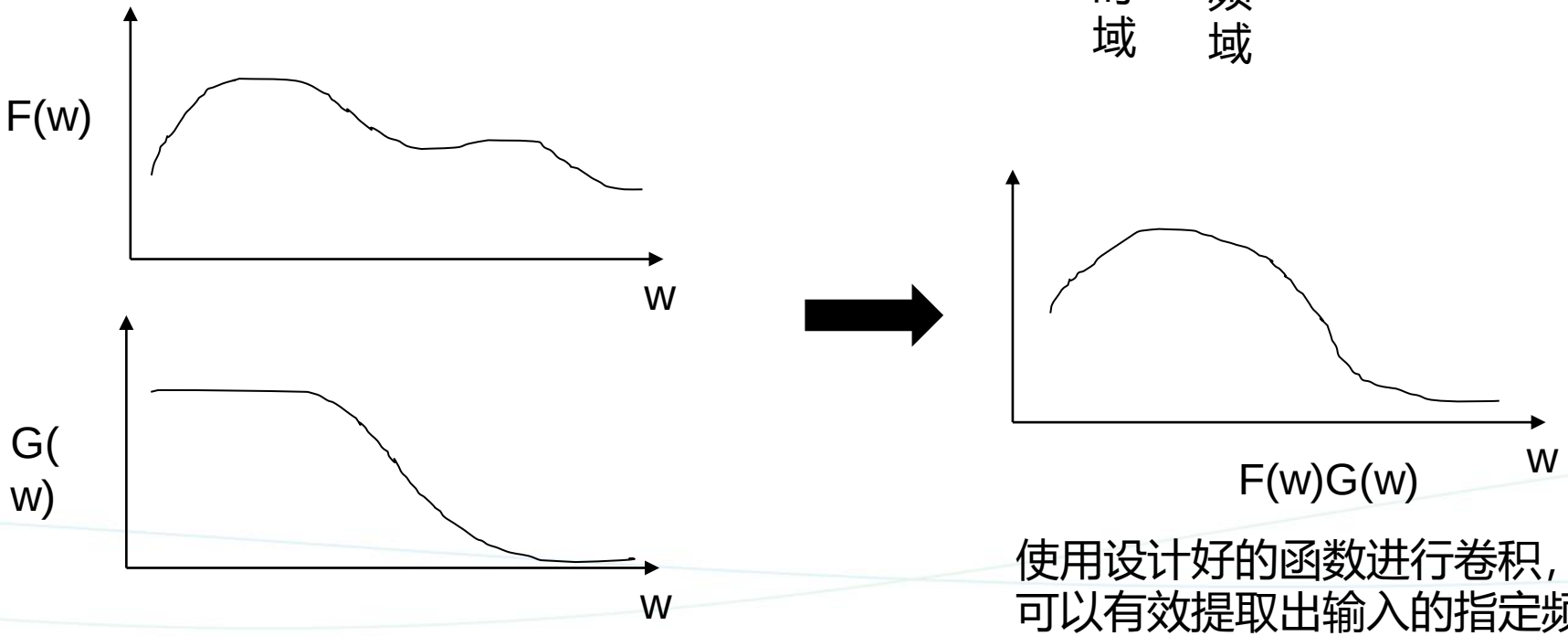
连续 $(f*g)(n)=\int_{-\infty}^{\infty} f(\tau)g(n-\tau)d\tau$

离散 $(f*g)(n)=\sum_{\tau=-\infty}^{\infty} f(\tau)g(n-\tau)$

f -> F
g -> G

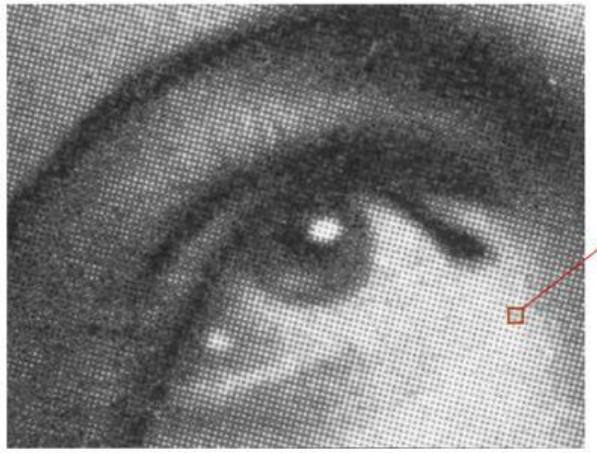
时域 频域

F(w)G(w)=FT[f(t)*g(t))]

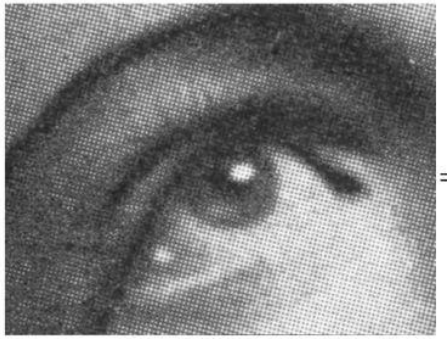


使用设计好的函数进行卷积，
可以有效提取出输入的指定频段信号

LeNet介绍—卷积层



这些噪点，属于高频信号



$$\Longrightarrow \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & \cdots & a_{0,n} \\ a_{1,0} & a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,0} & a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{m,0} & a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix} \qquad g = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

a, b的下标相加都为$1, 1$

$$f = \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} \\ a_{1,0} & a_{1,1} & a_{1,2} \\ a_{2,0} & a_{2,1} & a_{2,2} \end{bmatrix} \qquad g = \begin{bmatrix} b_{-1,-1} & b_{-1,0} & b_{-1,1} \\ b_{0,-1} & b_{0,0} & b_{0,1} \\ b_{1,-1} & b_{1,0} & b_{1,1} \end{bmatrix}$$

$$c_{1,1} = a_{0,0}b_{1,1}$$

LeNet介绍—卷积层

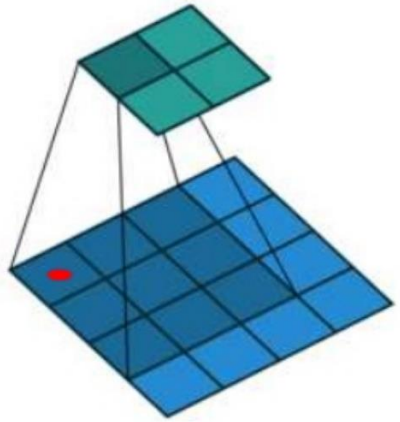
-1	0	+1
-2	0	+2
-1	0	+1

Gx

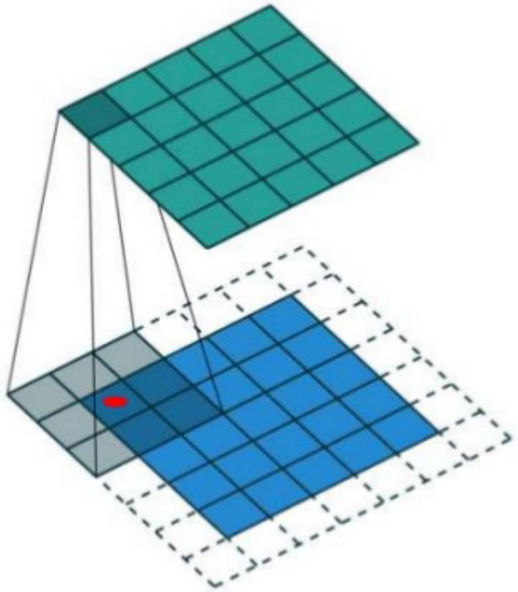
+1	+2	+1
0	0	0
-1	-2	-1

Gy

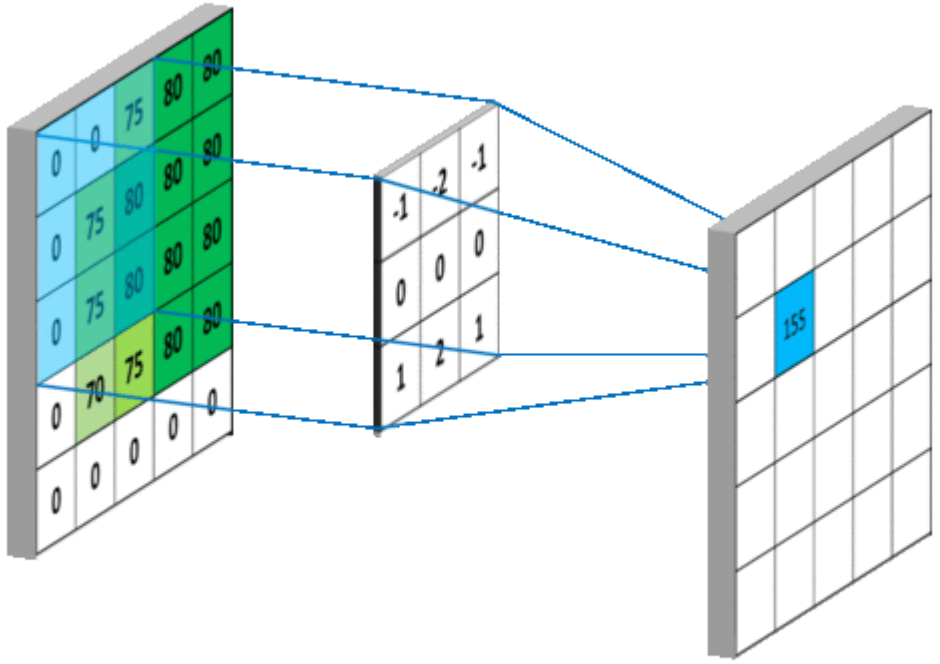
LeNet介绍—卷积层



Without Padding



With Padding



Output = (input – kernel + 2*padding)/stride + 1

LeNet介绍—卷积层

1. 卷积

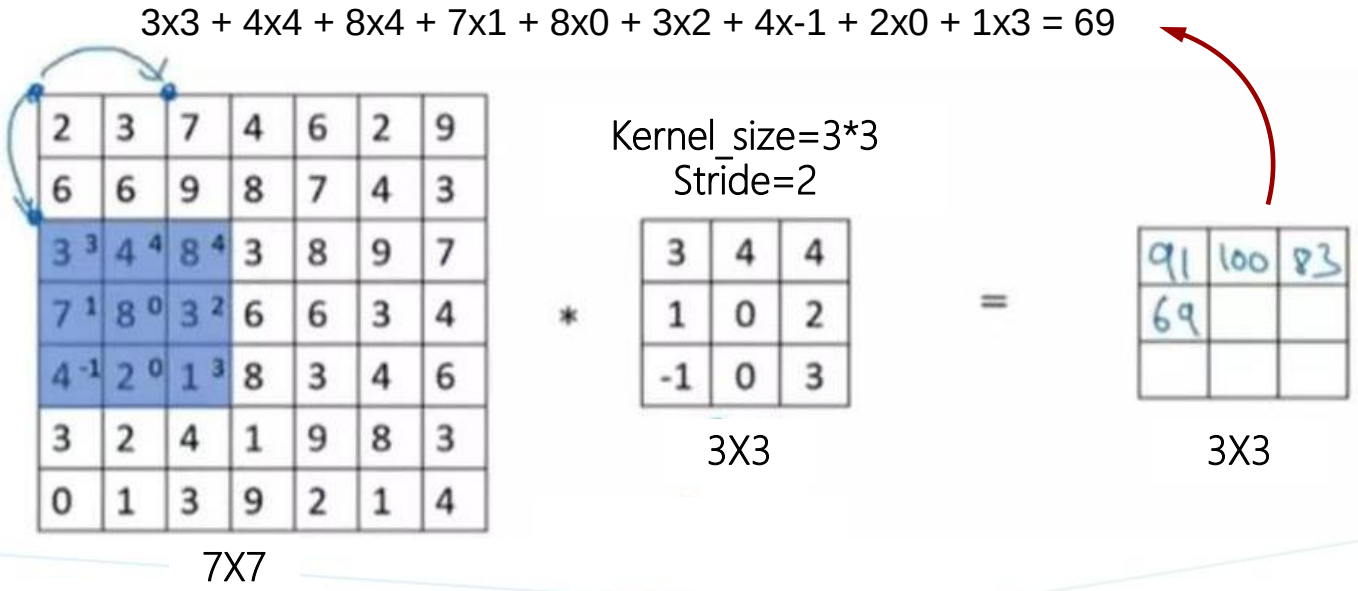
① 类似加权运算操作

- 图像处理中，卷积核逐行逐列扫描像素矩阵，得到新的像素矩阵
- 卷积核：滤波器

② 感受野:滤波器在输入图像像素矩阵上扫过的区域

2. 卷积神经网络

① 包含卷积计算且具有深度结构的神经网络



Mind Spore卷积层

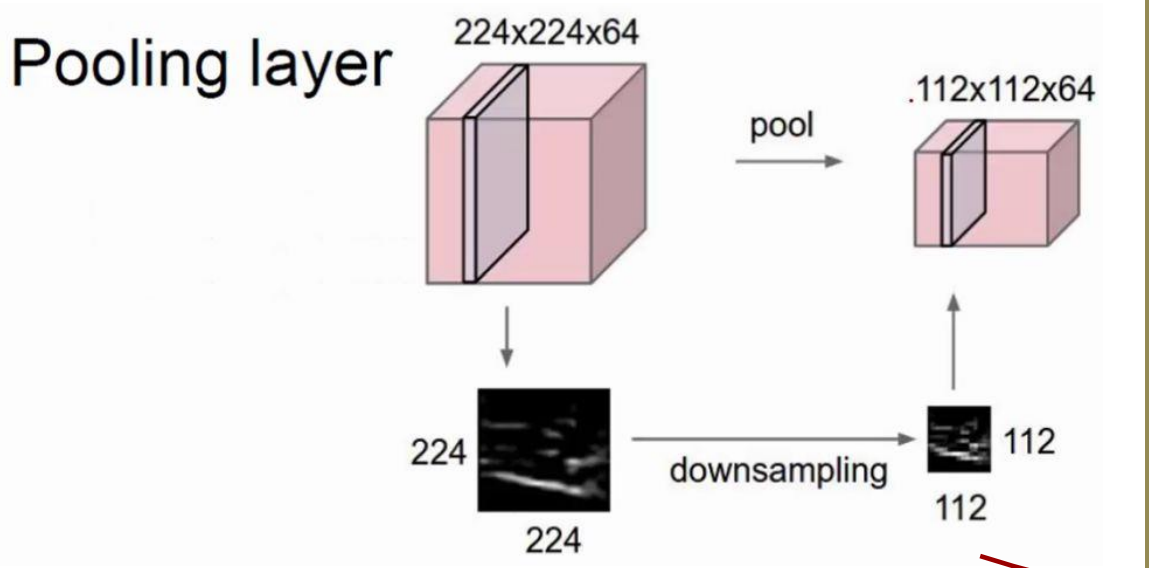
```
class Net(nn.Cell):
    def __init__(self, cin, cout, kernel_size, stride=1, pad_mode="valid",
                 padding=0, dilation=1, group=1, data_format='NCHW',
                 has_bias=True, weight_init='normal', bias_init='zeros'):
        super(Net, self).__init__()
        self.conv = nn.Conv2d(cin, cout, kernel_size, stride, pad_mode,
                              padding, dilation, group, data_format, has_bias,
                              weight_init, bias_init)

    def construct(self, input):
        return self.conv(input)

net = Net(3, 64, 4, has_bias=False, weight_init='normal')
input = Tensor(np.ones([1, 3, 16, 50]), ms.float32)
```

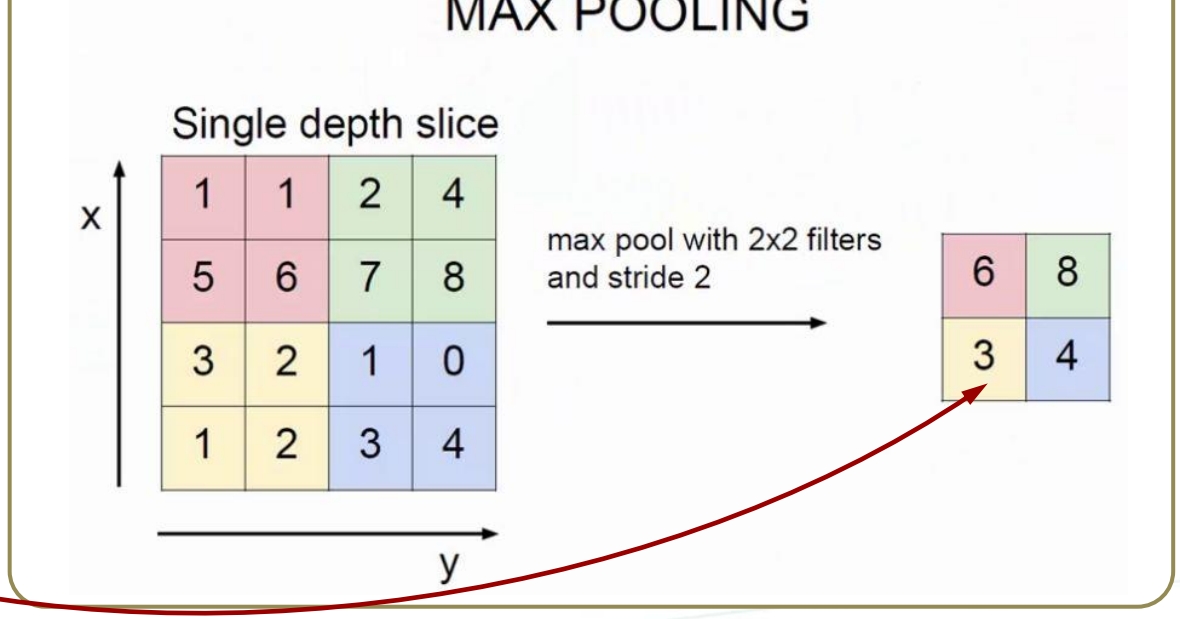
LeNet介绍—池化层

1. 指对特征图（或矩阵）进行特征压缩，亦称为下采样
2. 特征矩阵常划分为子矩阵，选择max或mean像素值替代导致特征矩阵尺寸变小

MindSpore池化层

```
pool = AvgPool2d(kernel_size=2, stride=2)
```





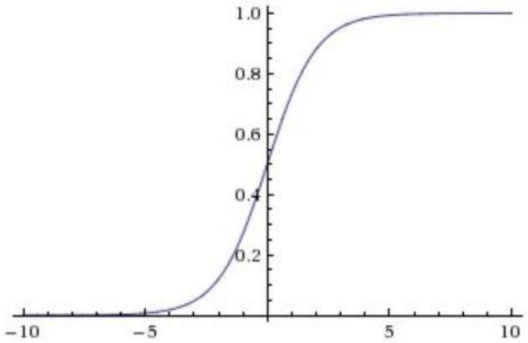
左图: 使用池化将原图224*224 变成112*112

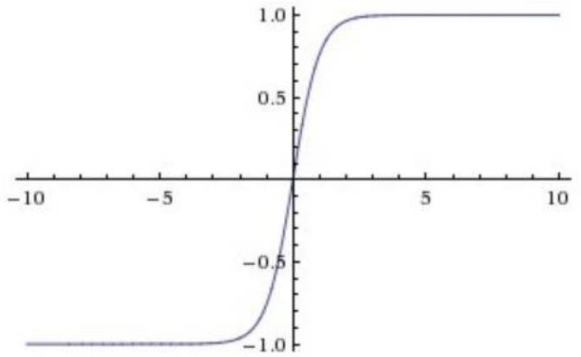
右图: 使用池化将特征矩阵4*4变成2*2

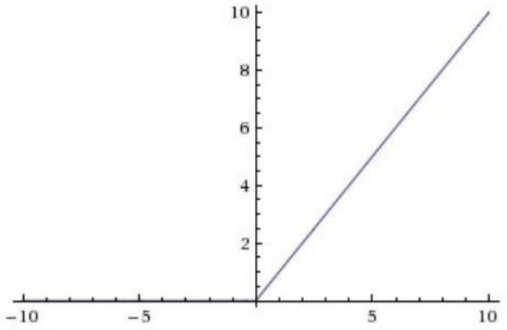
LeNet介绍—激活函数

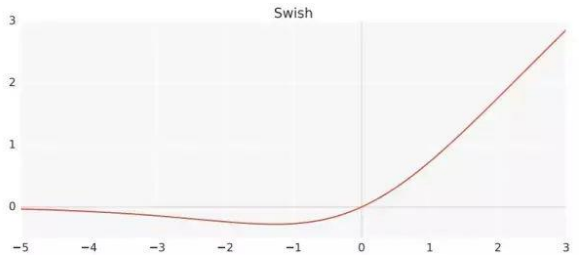
如果不用激励函数（其实相当于激励函数是f(x) = x，在这种情况下你每一层节点的输入都是上层输出的线性函数，很容易验证，无论你神经网络有多少层，输出都是输入的线性组合，与没有隐藏层效果相当，这种情况就是最原始的感知机（Perceptron）了，那么网络的逼近能力就相当有限。正因为上面的原因，我们决定引入非线性函数作为激励函数，这样深层神经网络表达能力就更加强大

- Sigmoid $f(z) = \frac{1}{1+e^{-z}}$
- Tanh $tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
- Relu $Relu = max(0, x)$
- Leaky-relu $f(x) = max(\alpha x, x)$
- Swish $f(x) = x \cdot \sigma(x)$

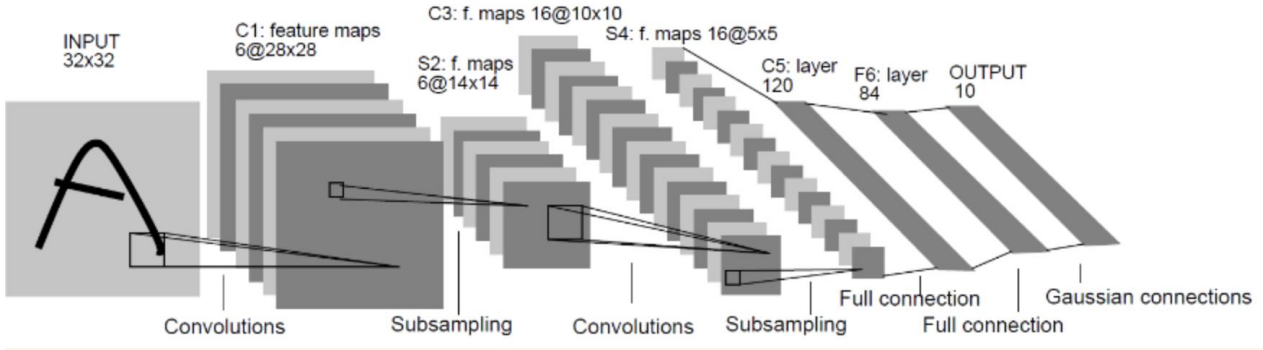




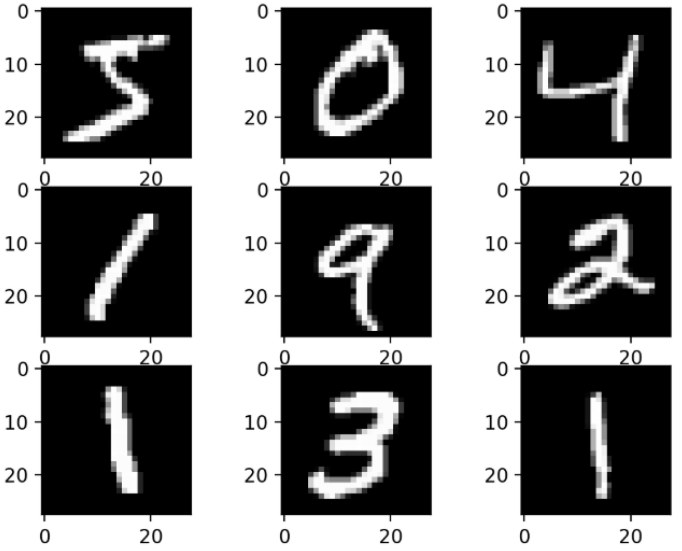




LeNet介绍

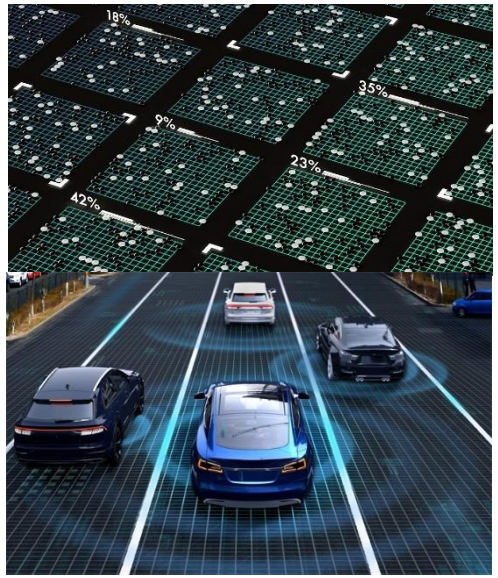


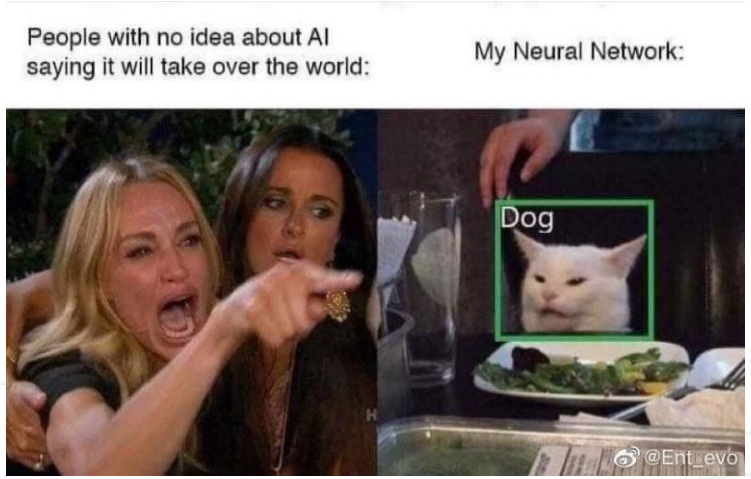
Yann Le Cun

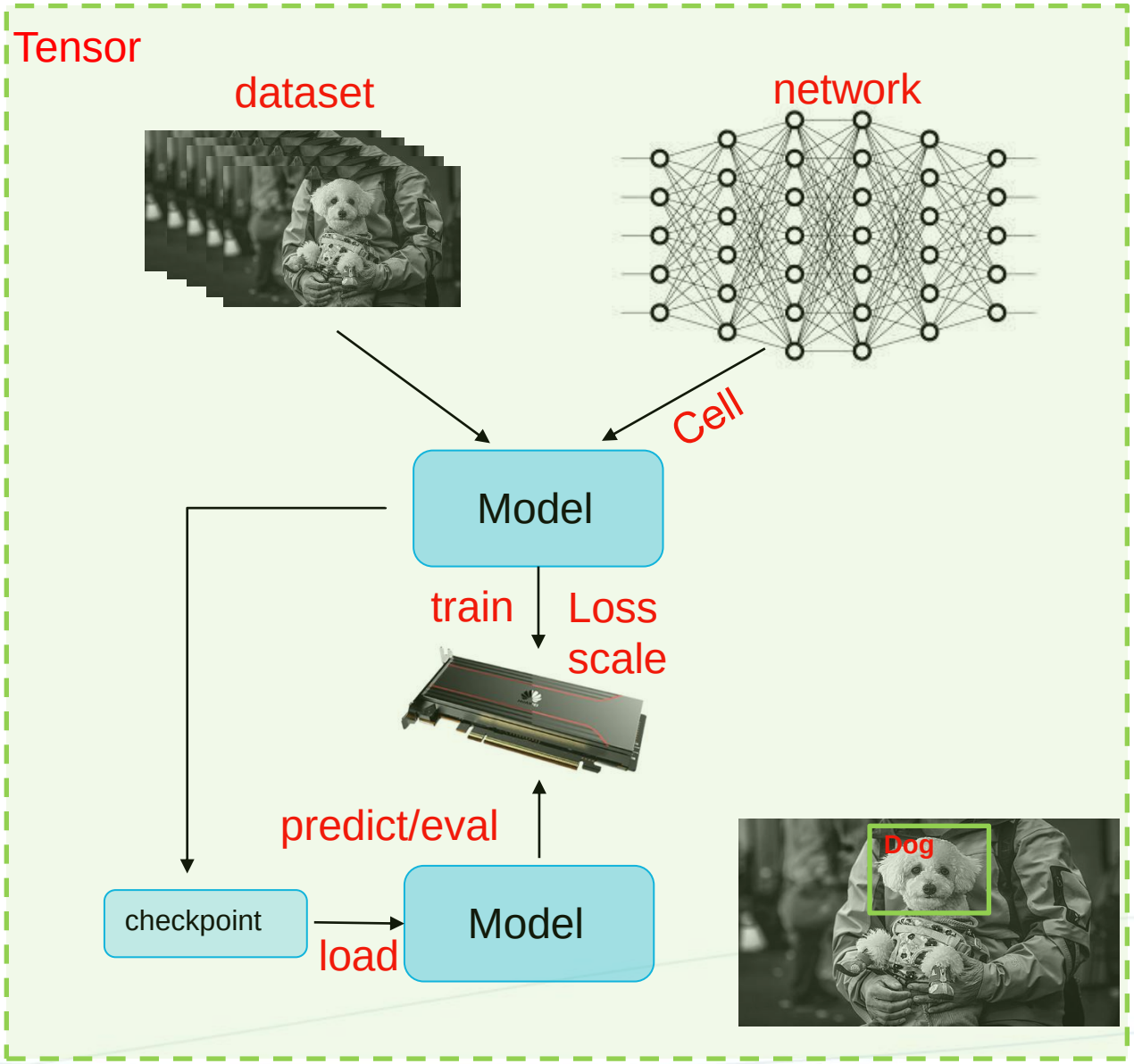


C1层是一个卷积层
S2层是一个下采样层
C3层也是一个卷积层
S4层是一个下采样层
C5层是一个全连接/卷积层
F6层试一个全连接层
Output是一个全连接层
Softmax+CrossEntropyLoss

构建网络模型基本步骤

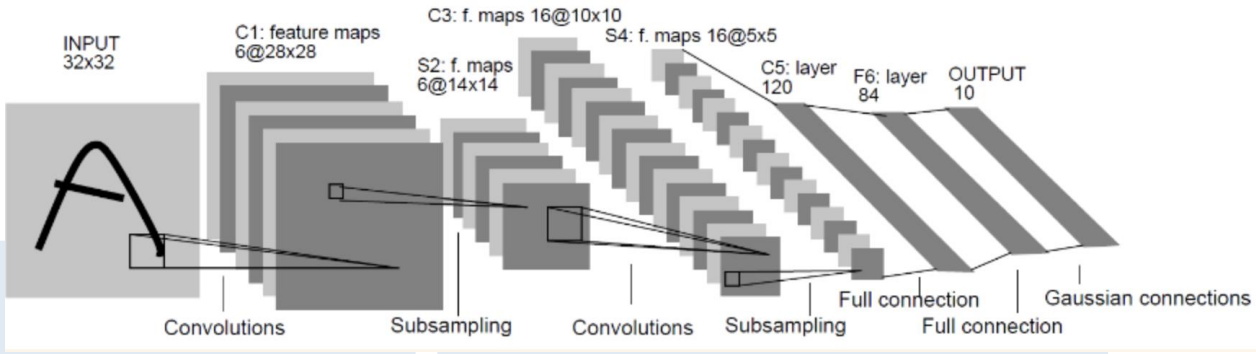

18%
35%
9%
23%
42%


People with no idea about AI saying it will take over the world:
My Neural Network:
Dog
@Ent_evo


Tensor
dataset
network
Cell
Model
train
Loss scale
predict/eval
checkpoint
load
Model
Dog

MindSpore构建分类网络模型 —— 组建一个简单的分类网络

```
import mindspore.nn as nn
from mindspore.common.initializer import TruncatedNormal
```




```
class LeNet5(nn.Cell):

    def __init__(self):
        super(LeNet5, self).__init__()
        param_init = TruncatedNormal(0.02)
        self.conv1 = nn.Conv2d(1, 6, 5, weight_init=param_init, pad_mode="valid")
        self.conv2 = nn.Conv2d(6, 16, 5, weight_init=param_init, pad_mode="valid")
        self.fc1 = nn.Dense(16 * 5 * 5, 120, weight_init=param_init, bias_init=param_init)
        self.fc2 = nn.Dense(120, 84, weight_init=param_init, bias_init=param_init)
        self.fc3 = nn.Dense(84, 10, weight_init=param_init, bias_init=param_init)
        self.relu = nn.ReLU()
        self.max_pool2d = nn.MaxPool2d(kernel_size=2, stride=2)
        self.flatten = nn.Flatten()

    …
```

```
def construct(self, x):
        x = self.conv1(x)
        x = self.relu(x)
        x = self.max_pool2d(x)
        x = self.conv2(x)
        x = self.relu(x)
        x = self.max_pool2d(x)
        x = self.flatten(x)
        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)
        x = self.relu(x)
        x = self.fc3(x)
        return x
```

模型训练与评估—— Mnist数据集训练LeNet5

1. 数据加载与数据预处理

model.train(epoch_size, dataset=ds)

```
import mindspore.dataset.engine as de
import mindspore.dataset.transforms.vision.c_transforms as C
import mindspore.dataset.transforms.c_transforms as C2

ds = de.MnistDataset(dataset_path, num_parallel_workers=8, shuffle=True)

# define map operations
random_horizontal_flip_op = C.RandomHorizontalFlip()

resize_op = C.Resize((32, 32))
rescale_op = C.Rescale(1.0 / 255.0, 0.0)
change_swap_op = C.HWC2CHW()

trans = []
if do_train:
    trans += [random_horizontal_flip_op]
trans += [resize_op, rescale_op, change_swap_op]

type_cast_op = C2.TypeCast(mstype.int32)
ds = ds.map(input_columns="label", num_parallel_workers=8, operations=type_cast_op)
ds = ds.map(input_columns="image", num_parallel_workers=8, operations=trans)

ds = ds.batch(batch_size, drop_remainder=True)
ds = ds.repeat(epoch_size)

return ds
```

对数据执行shuffle操作

定义数据集增强操作

调用map方法在图片上执行增强操作

数据批处理与repeat

MindSpore构建分类网络模型 —— 通过Model构建训练、推理网络

model = Model (network=net , loss_fn=loss , optimizer=opt ， metrics=metrics)

```
net = LeNet5(class_num = class_num)
loss = nn.SoftmaxCrossEntropyWithLogits(sparse=True)
lr = 0.002
opt = nn.Momentum(net.trainable_params(), lr=0.002, momentum=0.9, weight_decay=1e-4, loss_scale=loss_scale_num)
metrics ={'acc'}
model.train(epoch, dataset, callbacks=[LossMonitor(), Checkpoint_cb], dataset_sink_mode=True)
Acc = model.eval(dataset, dataset_sink_mode=False)
```

损失函数：

用于描述模型预测值与真实值之间的误差

MindSpore支持的损失函数有：

L1Loss、 MSELoss、SoftmaxCrossEntropyWithLogits等

优化器：

用于最小化损失函数

MindSpore支持的优化器有：

Momentum、Adam、LARS、ProximalAdagrad等

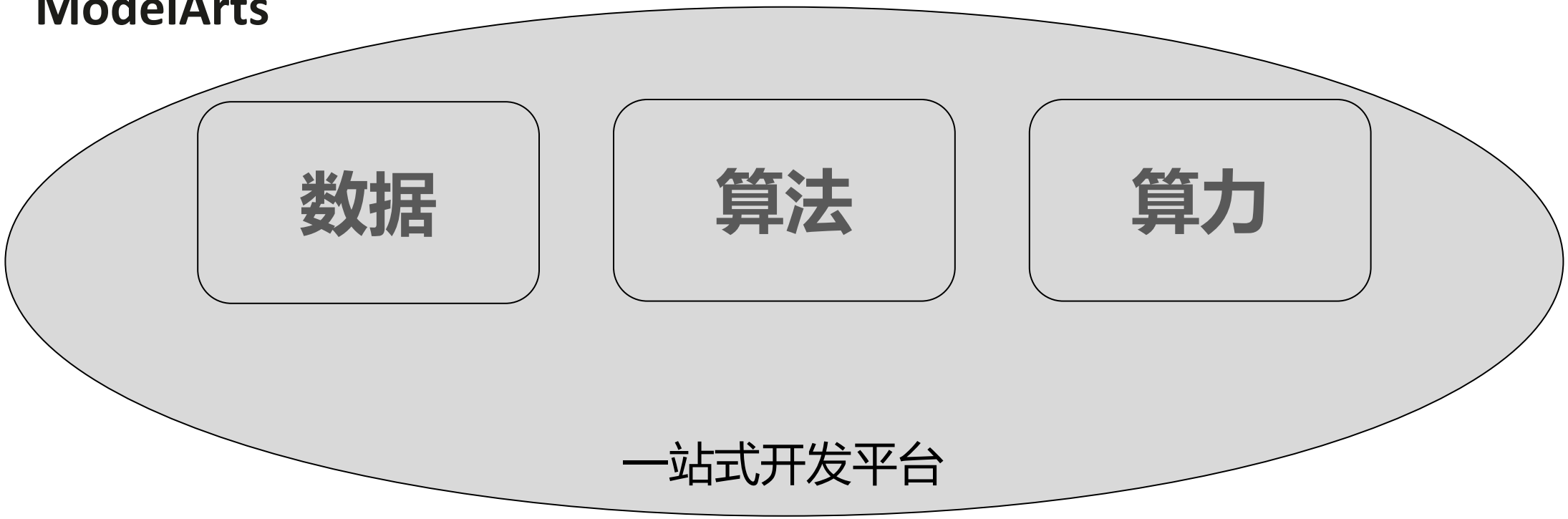
评价函数：

用于评估训练结果的准确度

MindSpore支持的评价函数有：

Accuracy、TopK等，评价函数可以Python语句和Cell两种方式表达

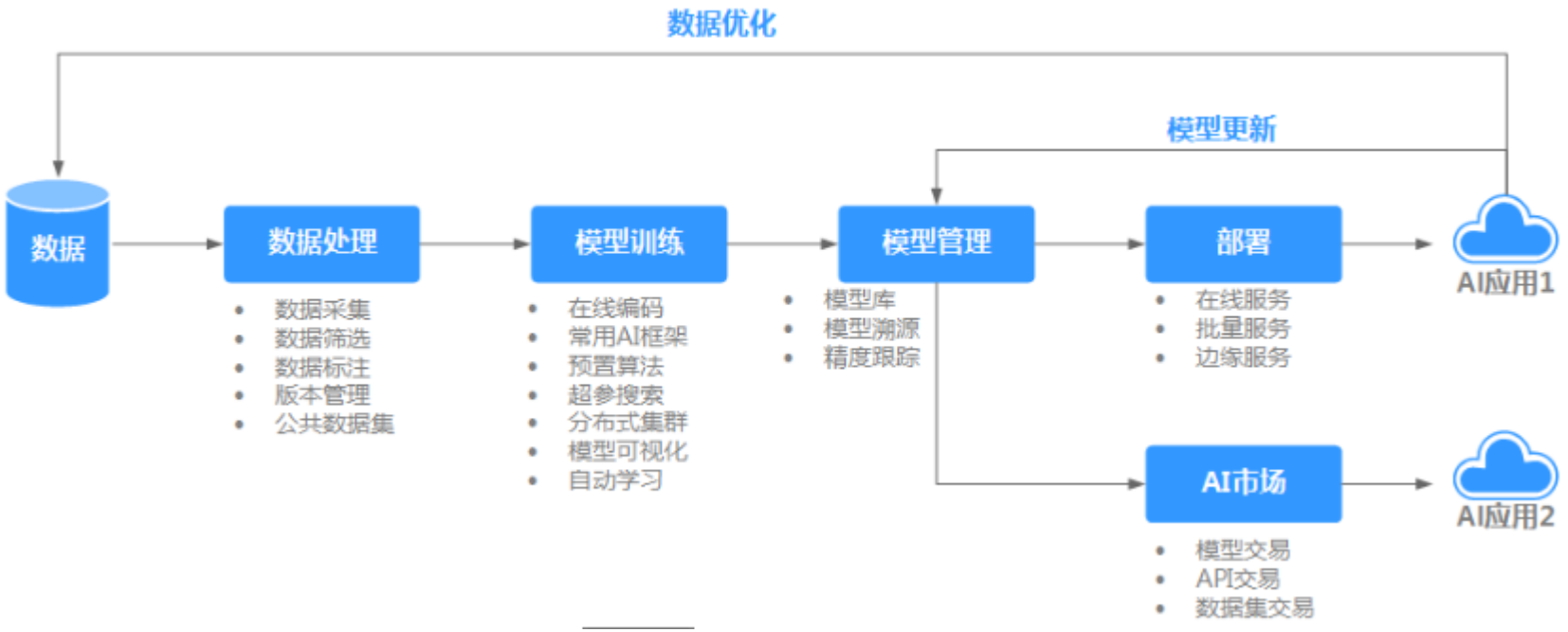
ModelArts



“一站式”是指AI开发的各个环节，包括数据处理、算法开发、模型训练、模型部署都可以在ModelArts上完成。从技术上看，ModelArts底层支持各种异构计算资源，开发者可以根据需要灵活选择使用，而不需要关心底层的技术。同时，ModelArts支持Tensorflow、MXNet等主流开源的AI开发框架，也支持开发者使用自研的算法框架，匹配您的使用习惯。

ModelArts

ModelArts是面向AI开发者的一站式开发平台，提供海量数据预处理及半自动化标注、大规模分布式训练、自动化模型生成及端-边-云模型按需部署能力，帮助用户快速创建和部署模型，管理全周期AI工作流。



https://www.huaweicloud.com
https://gitee.com/mindspore/course/blob/master/experiment_1/1-LeNet5_MNIST.ipynb
https://www.bilibili.com/video/BV1va4y1Y7is?from=search&seid=27055485625143577 52

THANK YOU

MindSpore开源社区：
https://www.mindspore.cn

MindSpore代码：
https://gitee.com/mindspore/mindspore