

# 零基础入门旷视天元MegEngine

**MEGVII** 旷视

# 模型构建和训练入门

讲师：刘清一

天元开发者交流群

群号：1029741705



扫一扫二维码，加入群聊。

# | 零基础入门旷视天元MegEngine

MEGVII 旷视

## 课程大纲：

- 数据的加载与预处理
- 卷积神经网络的构建
- 梯度下降法与模型训练
- 多 GPU 数据并行
- 模型的保存与加载

天元官网：<https://megengine.org.cn/>

GitHub：<https://github.com/MegEngine>

天元开发者交流群

群号：1029741705



扫一扫二维码，加入群聊。

1

**数据的加载与预处理**

2

**卷积神经网络的构建**

3

**梯度下降法与模型训练**

4

**多 GPU 数据并行**

5

**模型的保存与加载**

天元开发者交流群

群号: 1029741705



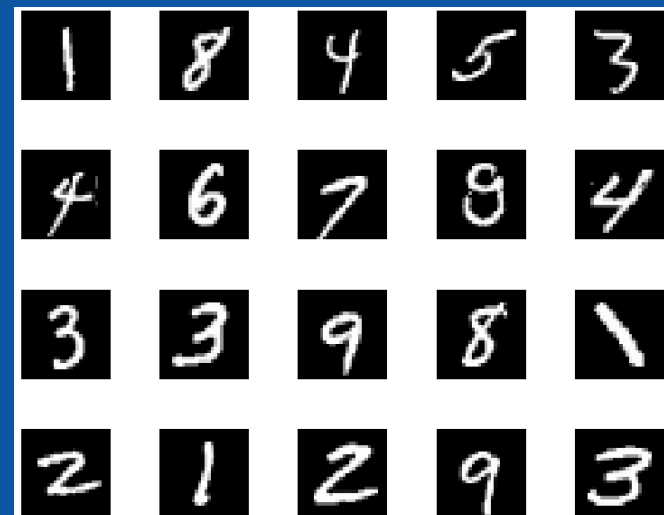
扫一扫二维码，加入群聊。

- Dataset
- Sampler
- DataLoader
- Transform
  - Pad
  - RandomResizedCrop
  - Normalize
  - ToMode
  - Compose



# | MNIST 数据集

- 0-9 手写数字
- 60000 张训练图片
- 10000 张测试图片
- `shape = 28 x 28 x 1`
- `mean = 33.32`
- `std = 78.57`

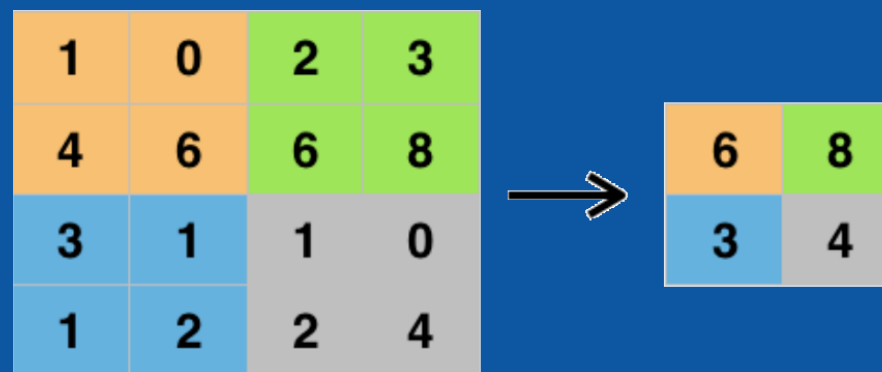


- 1 数据的加载与预处理
- 2 卷积神经网络的构建
- 3 梯度下降法与模型训练
- 4 多 GPU 数据并行
- 5 模型的保存与加载



# 卷积神经网络的基本算子

- Convolution
- MaxPooling
- ReLU
  - $relu(x) = \max(0, x)$
- Linear
  - $y = x \cdot w^T + b$





- Module = Functions + Parameters

```
class MnistNet(M.Module):  
    def __init__(self):  
        self.conv0 = M.Conv2d(1, 20, kernel_size=5)  
        self.pool0 = M.MaxPool2d(2)  
        self.conv1 = M.Conv2d(20, 20, kernel_size=5)  
        self.pool1 = M.MaxPool2d(2)  
        self.fc0 = M.Linear(320, 500)  
        self.fc1 = M.Linear(500, 10)  
  
    def forward(self, x):  
        x = self.conv0(x)  
        x = F.relu(x)  
        x = self.pool0(x)  
        x = self.conv1(x)  
        x = F.relu(x)  
        x = self.pool1(x)  
        x = F.flatten(x, 1)  
        x = self.fc0(x)  
        x = F.relu(x)  
        x = self.fc1(x)  
        return x
```

Conv0

ReLU

Pooling

Conv1

ReLU

Pooling

FC0

FC1

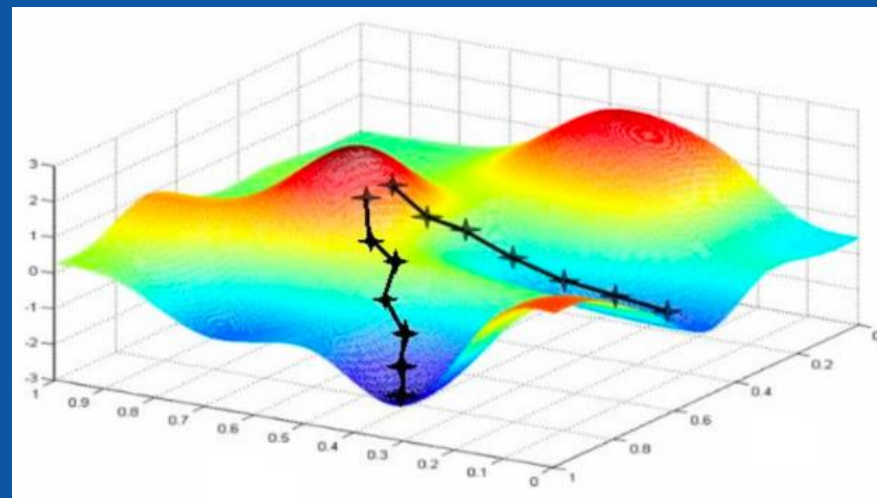


- 1 数据的加载与预处理
- 2 卷积神经网络的构建
- 3 梯度下降法与模型训练
- 4 多 GPU 数据并行
- 5 模型的保存与加载



# I 梯度下降法

- $pred = kx + b$
- $loss = (pred - y)^2$
- $\frac{\partial loss}{\partial k} = 2(pred - y)x$
- $\frac{\partial loss}{\partial b} = 2(pred - y)$
- $k \leftarrow k - lr \cdot \frac{\partial loss}{\partial k}$
- $b \leftarrow b - lr \cdot \frac{\partial loss}{\partial b}$



# I 梯度下降法

```
k = 0  
b = 0  
lr = 0.01
```

```
for epoch in range(10):  
    sum_grad_k = 0  
    sum_grad_b = 0  
    loss = 0
```

zero\_grad

```
for x, y in data:  
    pred = k * x + b  
    loss += (pred - y) * (pred - y)
```

forward

```
    sum_grad_k += 2 * (pred - y) * x  
    sum_grad_b += 2 * (pred - y)
```

backward

```
grad_k = sum_grad_k / N  
grad_b = sum_grad_b / N
```

```
k = k - lr * grad_k  
b = b - lr * grad_b
```

step



```
from megengine.optimizer import SGD
from megengine.jit import trace
```

```
opt = SGD(model.parameters(), lr = 0.01)
```

```
@trace
def train(data, label):
    pred = model(data)
    loss = F.cross_entropy_with_softmax(pred, label)
    opt.backward(loss)
    return loss
```

forward

backward

```
for epoch in range(10):
    train_loss = 0
    for data, label in train_data:
        opt.zero_grad()
        loss = train(data, label)
        opt.step()
    train_loss += loss.numpy()
```

zero\_grad

step

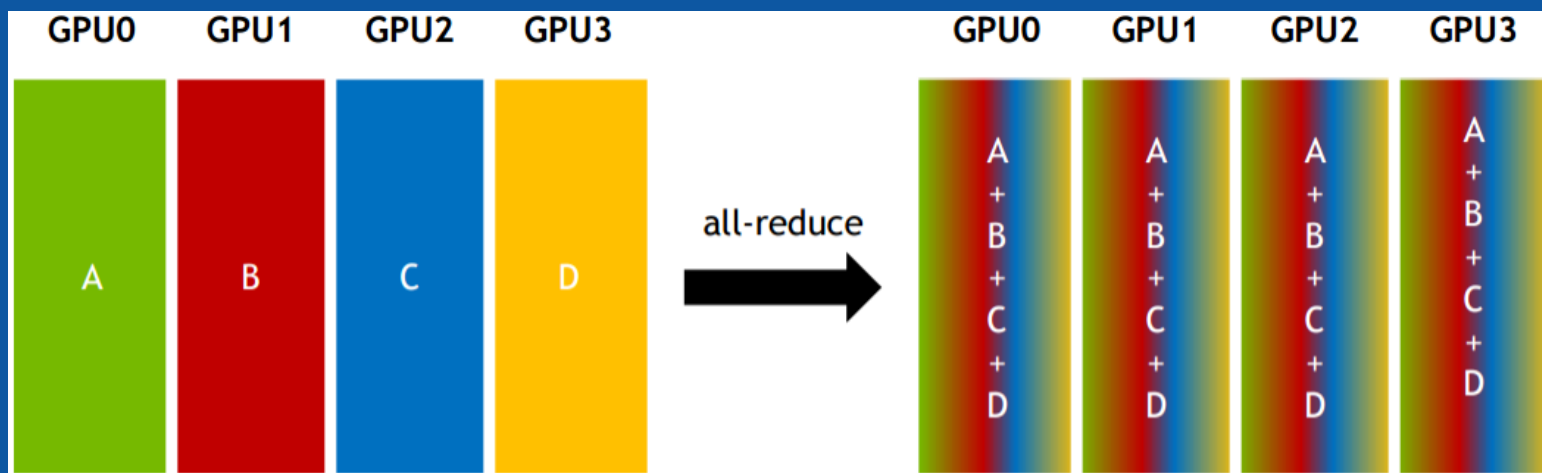


- 1 数据的加载与预处理
- 2 卷积神经网络的构建
- 3 梯度下降法与模型训练
- 4 多 GPU 数据并行**
- 5 模型的保存与加载



# | 多 GPU 数据并行

- $param = param - lr \cdot all\_reduce\_sum(grad)$
- NCCL (Nvidia Collective Communication Library)



```
import multiprocessing as mp
import megengine.distributed as dist

def run(rank):
    dist.init_process_group(master_ip, port, world_size, rank, device)

    // training

if __name__ == "__main__":
    for rank in range(8):
        p = mp.Process(target=run, args=(rank,))
        p.start()
        procs.append(p)

    for p in procs:
        p.join()
```

init process group

one process per GPU





- 1 数据的加载与预处理
- 2 卷积神经网络的构建
- 3 梯度下降法与模型训练
- 4 多 GPU 数据并行
- 5 模型的保存与加载



- 模型的保存

- `megengine.save(model.state_dict(), filename)`

- 模型的加载

- `model.load_state_dict(megengine.load(filename))`



# | 作业:

- 使用 MegEngine 搭建 LeNet5 网络
- 在 MNIST 数据集上进行训练
- 保存训练好的模型
- 计算在测试集上的准确率



## 提交:

- 将 MegStudio 运行成功的截图加上以下信息发送到 [mgesupport@megvii.com](mailto:mgesupport@megvii.com)

邮件标题: 天元入门第二次课程作业

姓名:

学校 (公司):

电话:

邮寄地址:



- 1 数据的加载与预处理
- 2 卷积神经网络的构建
- 3 梯度下降法与模型训练
- 4 多 GPU 数据并行
- 5 模型的保存与加载

