



智能系统安全实践：CNN与对抗样本

复旦白泽智能
系统软件与安全实验室



大纲



- 理解卷积神经网络 (CNN)
 - 卷积和池化操作
 - 经典LeNet5模型
- 理解对抗样本的建模
 - 问题形式化
 - FGSM的解法

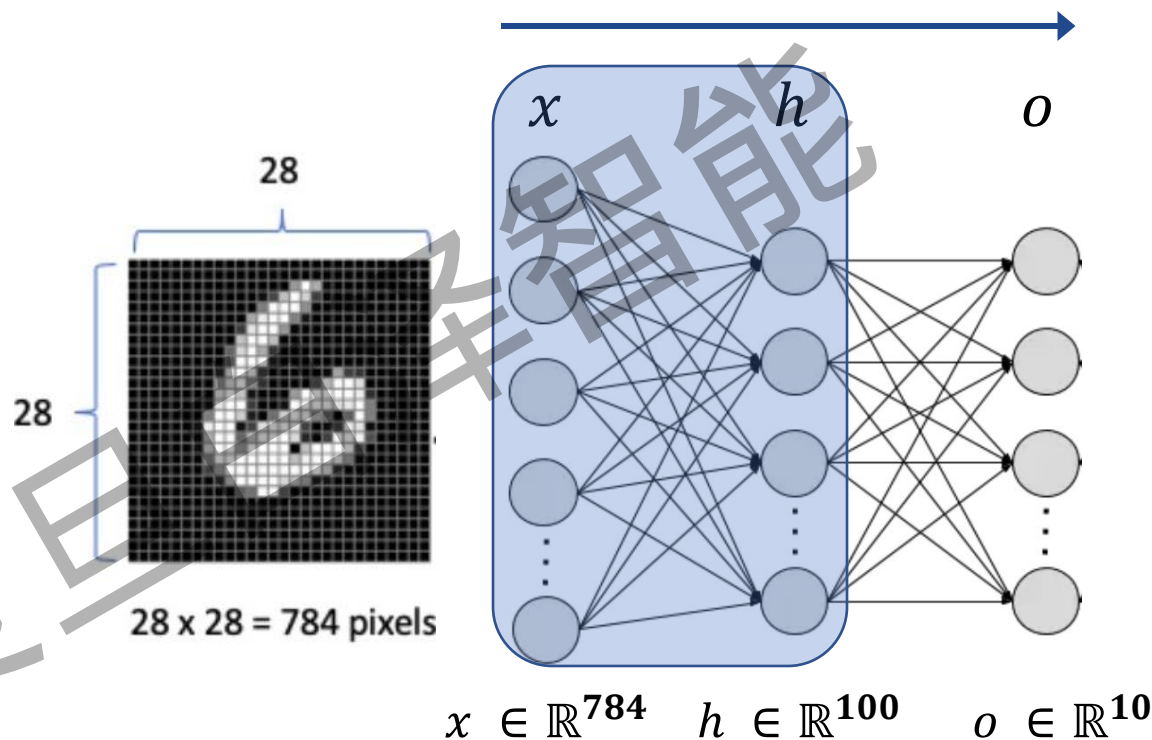
回顾：多层感知机（MLP）

■ 模型定义

■ $h = \sigma(W_h x + b_h)$

■ $o = W_o h + b_o$

■ 其中 $\sigma(h) = \frac{1}{1+e^{-h}}$



这里 $W_h \in \mathbb{R}^{784 \times 100}$

卷积操作

■ 卷积 (Convolution)

■ 局部特征提取:

- 不同于线性层的全局特征提取
- 能捕获更细粒度的图像**局部特征** (边缘、纹理等)

■ 各个局部位置进行相同操作:

■ 滑动窗口

- 参数共享 -> 减少参数量

■ 每个局部如何操作?

- 逐元素相乘、相加
- 本质上仍是 $y = Wx + b$

0	1	2
2	2	0
0	1	2

卷积核

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

图片

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

特征图

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

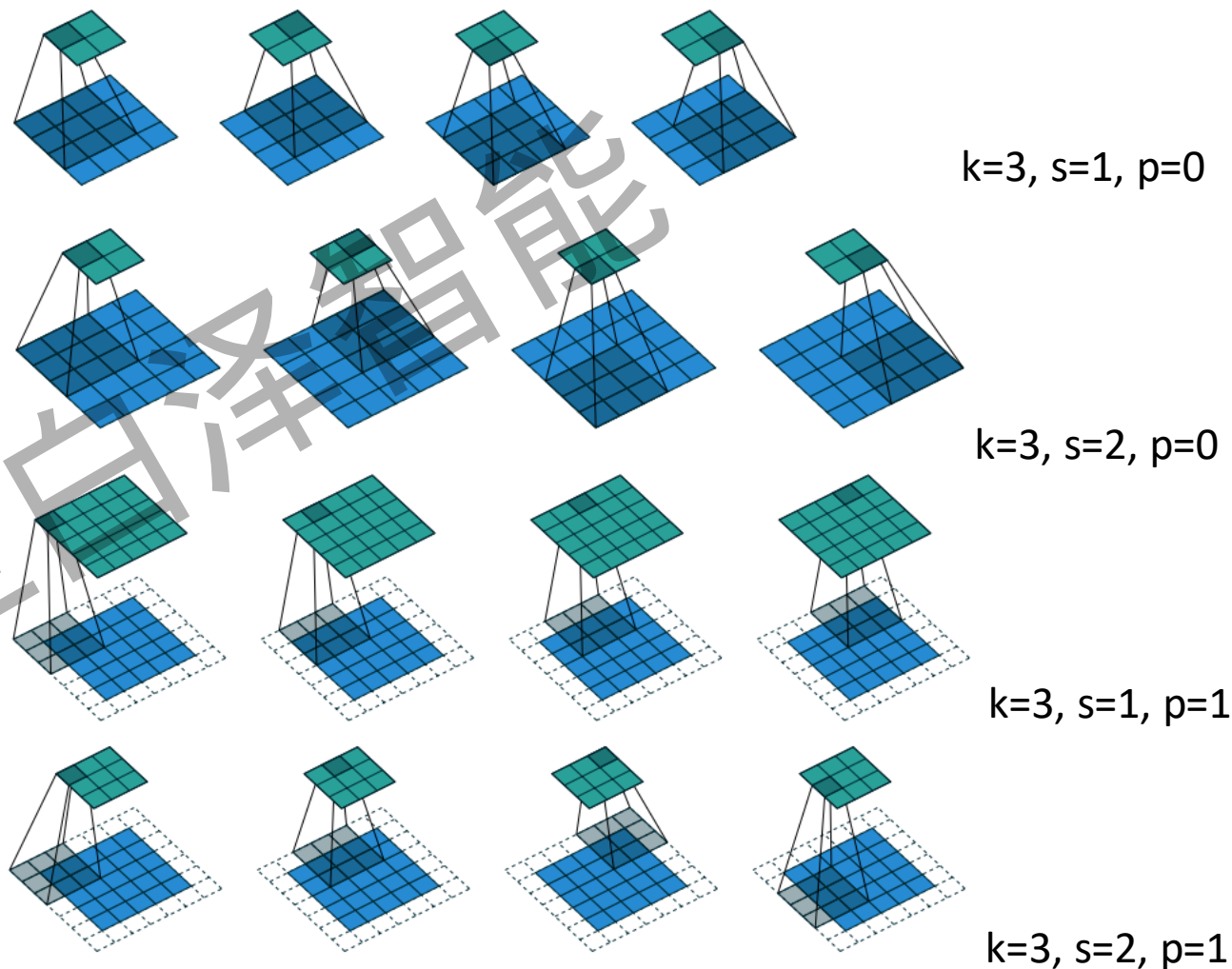
卷积的关键参数

- kernel_size: 卷积核大小
- stride: 滑动步长
- padding: 两侧填充数目

更多可视化资源:

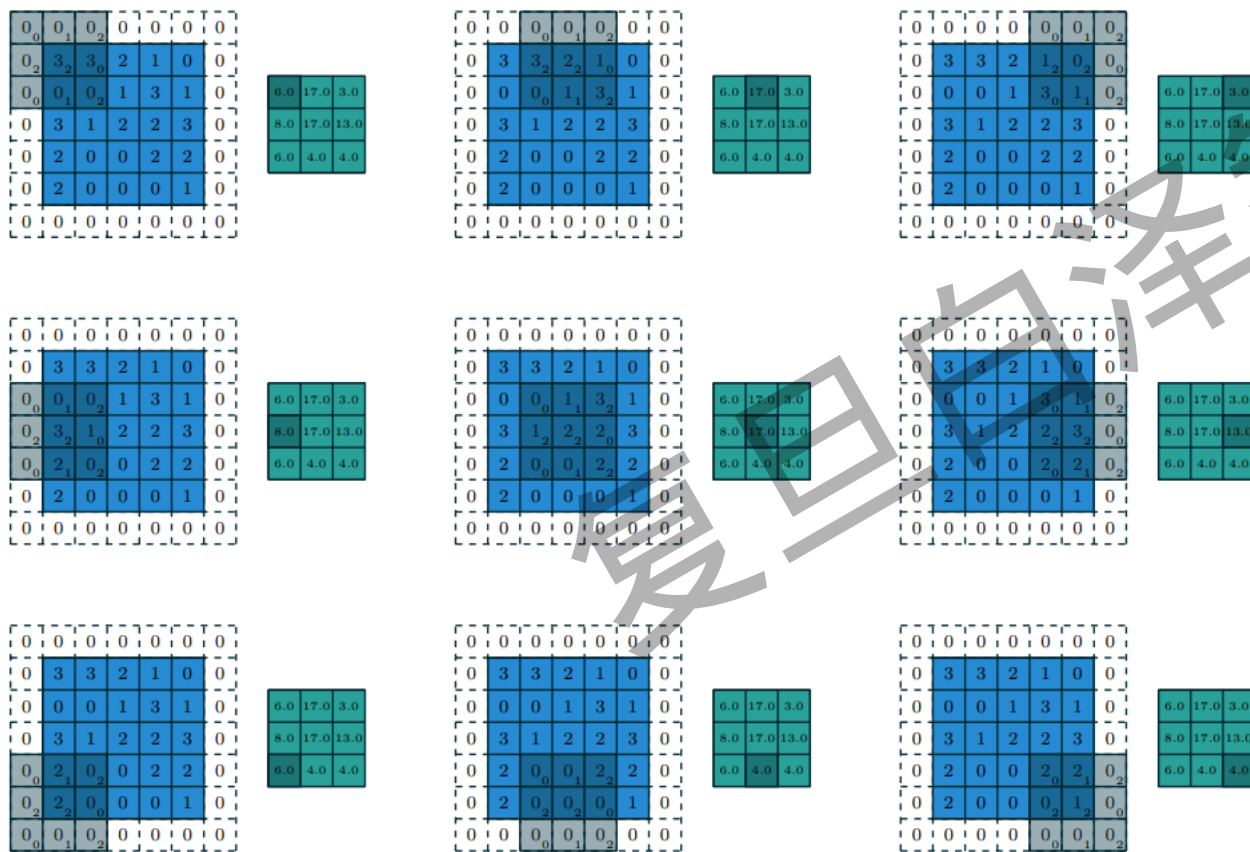
https://github.com/vdumoulin/conv_arithmetic/blob/master/README.md (有动图!)

<https://arxiv.org/pdf/1603.07285.pdf>



卷积的关键参数

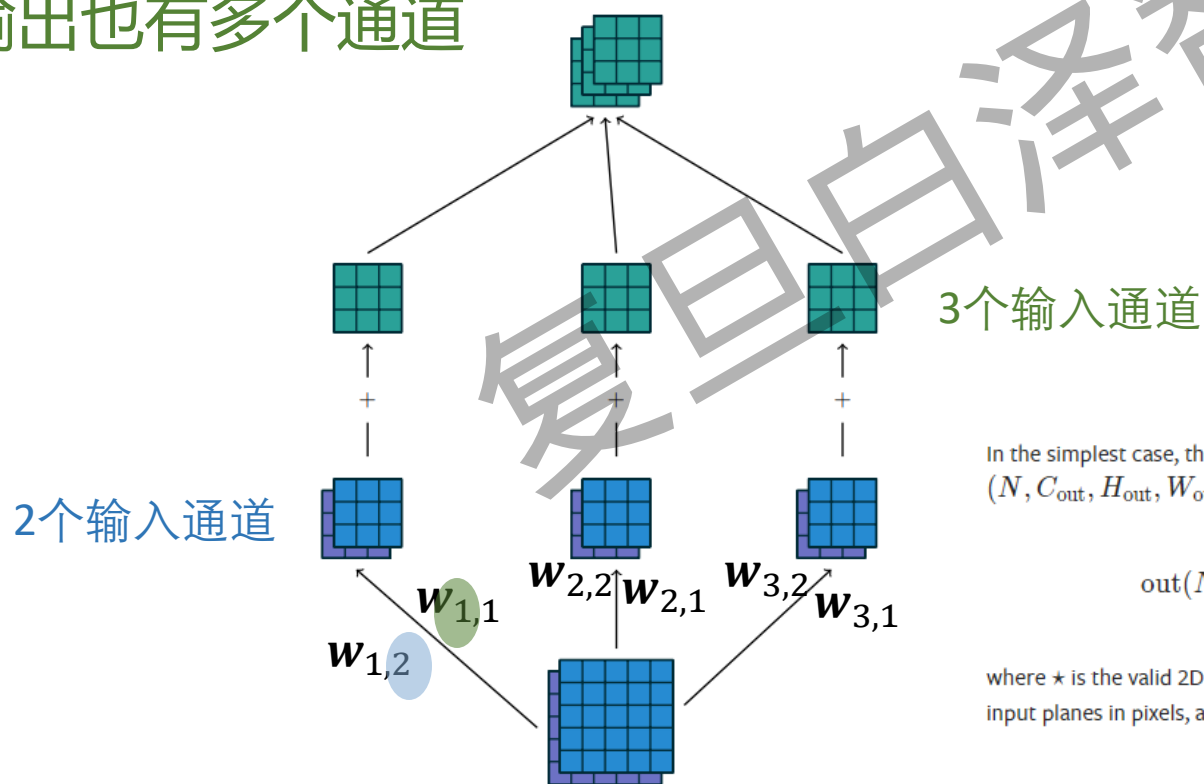
■ 下面这个卷积操作，各个参数应该是多少呢？



kernel_size = 3
stride = 2
padding = 1

CNN中的卷积

- 之前我们学习的卷积，输入都是单通道，输出也是单通道
- 在实际CNN中，每个卷积层
 - 输入有多个通道（例如，许多图片便是RGB三通道的）
 - 输出也有多个通道



In the simplest case, the output value of the layer with input size (N, C_{in}, H, W) and output $(N, C_{out}, H_{out}, W_{out})$ can be precisely described as:

$$\text{out}(N_i, C_{out_j}) = \text{bias}(C_{out_j}) + \sum_{k=0}^{C_{in}-1} \text{weight}(C_{out_j}, k) \star \text{input}(N_i, k)$$

where \star is the valid 2D cross-correlation operator, N is a batch size, C denotes a number of channels, H is a height of input planes in pixels, and W is width in pixels.

池化层

■ 池化 (Pooling)

■ 卷积得到的“特征图”可能包含许多冗余信息

■ 池化本质是降采样

■ 减少特征图大小

■ 保留最重要的特征

■ 常见的池化操作:

■ max pooling

■ average pooling

■ 关键参数

■ kernel_size

■ stride, padding

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

kernel_size = 3

ReLU激活函数

■ 如果没有激活函数？

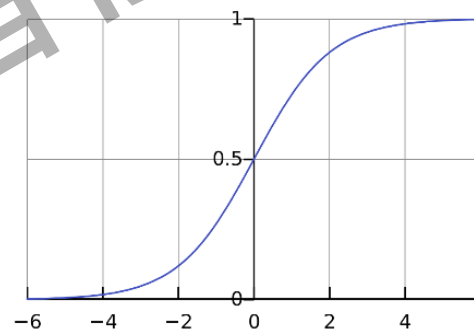
$$\blacksquare f(x) = W_3(W_2(W_1x + b_1) + b_2) + b_3 = W_*x + b_*$$

■ 激活函数：非线性的形式，给神经网络增加表达能力

■ 我们之前学习过的sigmoid激活函数

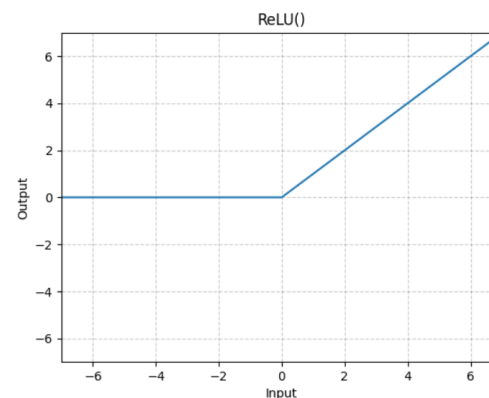
$$\blacksquare \text{公式: } \sigma(h) = \frac{1}{1 + e^{-h}}$$

■ 计算复杂，且正负饱和区存在梯度消失的风险



■ CNN中更为常用的是ReLU激活函数

$$\blacksquare \text{公式: } \text{ReLU}(x) = (x)^+ = \max(0, x)$$

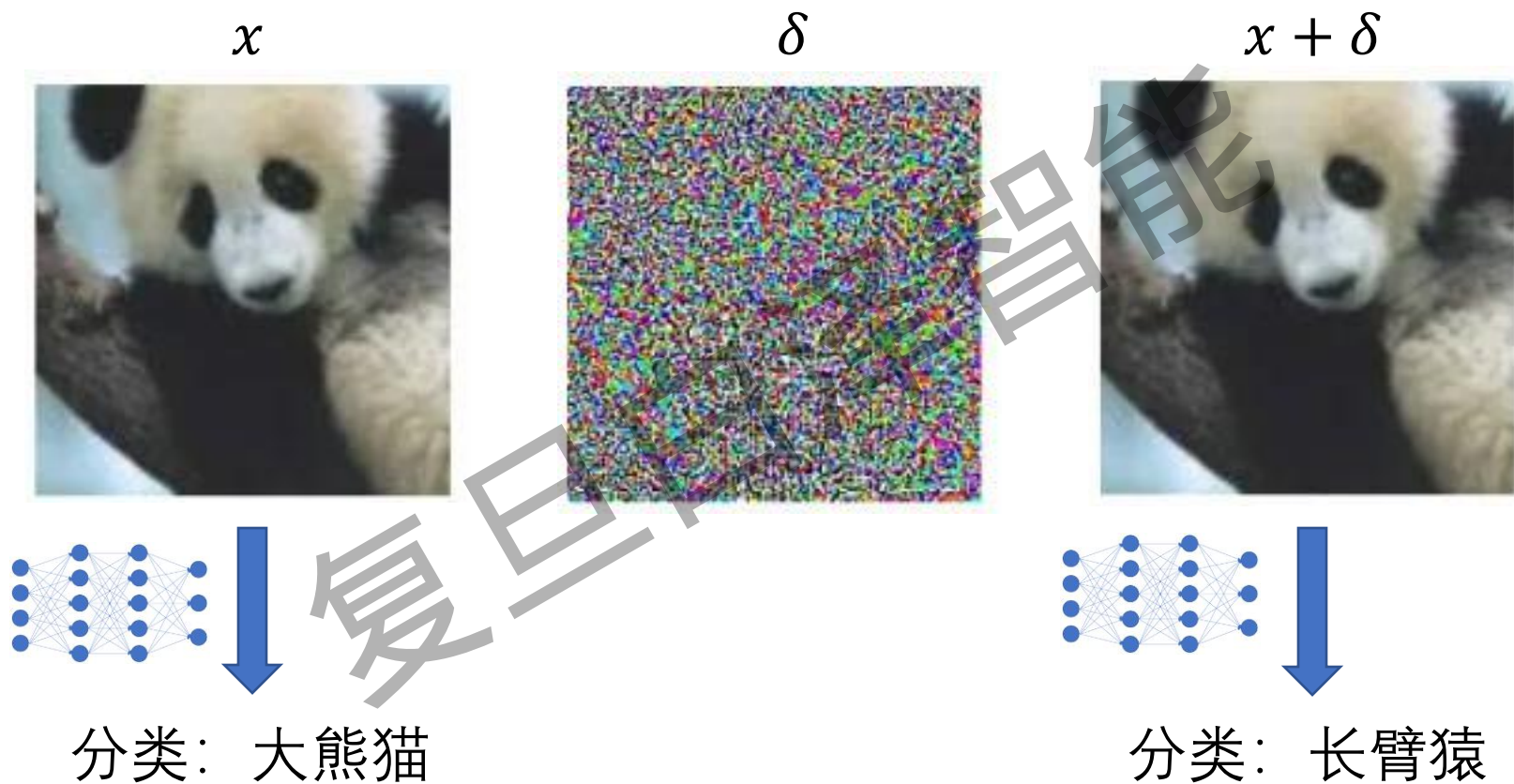




Q&A

复旦翻译智能

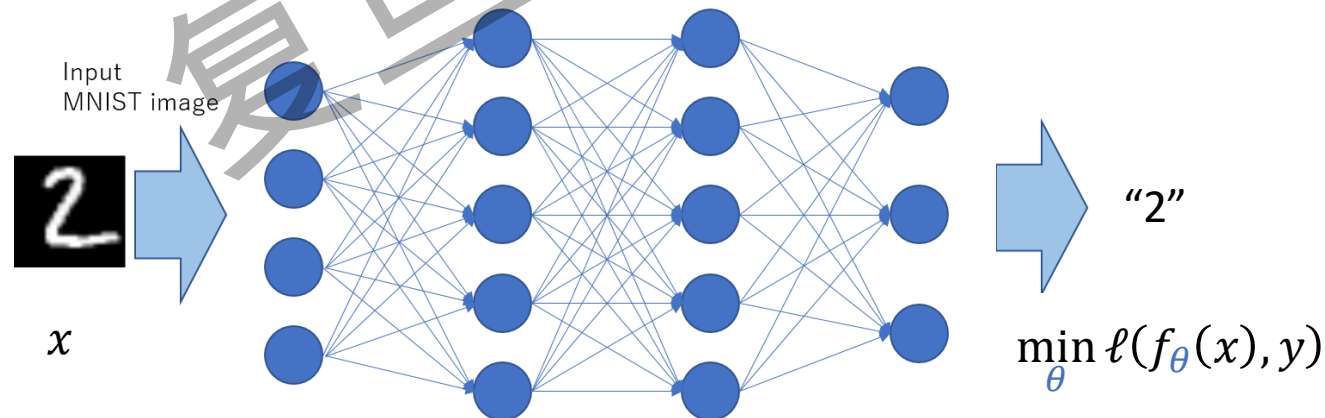
对抗样本



- 如何实现上述攻击?

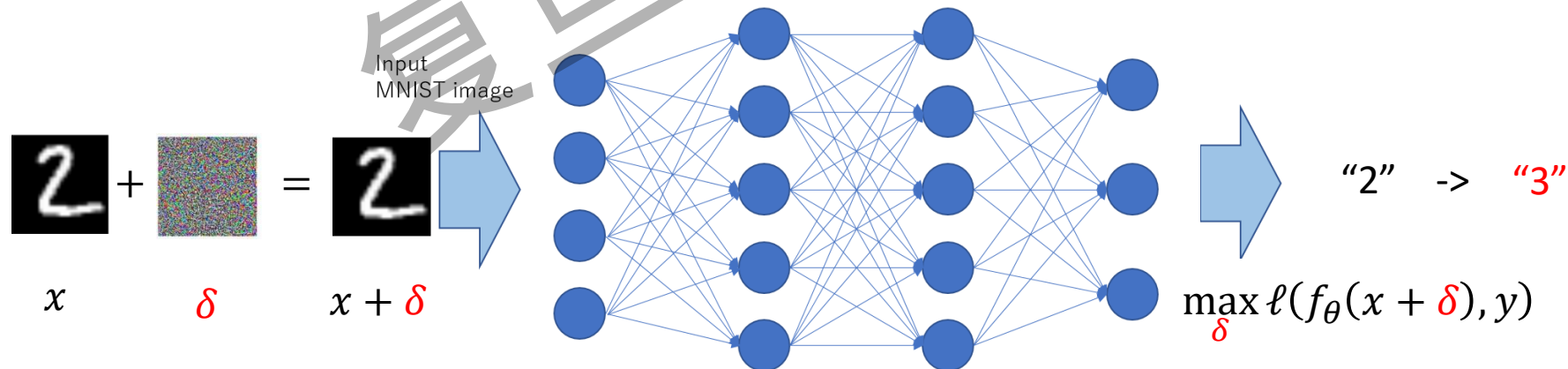
回顾：最小化分类损失

- 回忆模型学习目标：
- $\min_{\theta} \ell(f_{\theta}(x), y)$
- 给定数据点的情况下，通过改变 θ ，最小化损失函数
- 目标：让模型的分类更加准确



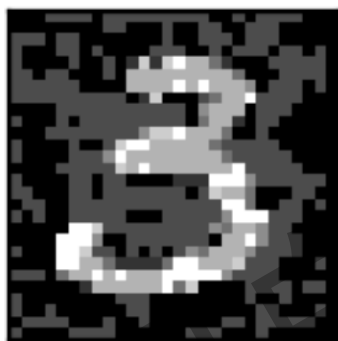
对抗样本：最大化分类损失

- 对抗样本攻击目标：
- $\max_{\delta} \ell(f_{\theta}(x + \delta), y)$
- 给定数据点和模型的情况下，通过改变 δ ，最大化损失函数
- 目标：让模型的分类出错



对抗样本：不可见扰动

- 如果对 δ 没有限制，加扰动的图片可能非常离谱



- 解决方案：

- $$\max_{\delta} \ell(f_{\theta}(x + \delta), y), \quad \text{s.t.} \quad \|\delta\|_{\infty} \leq \epsilon$$

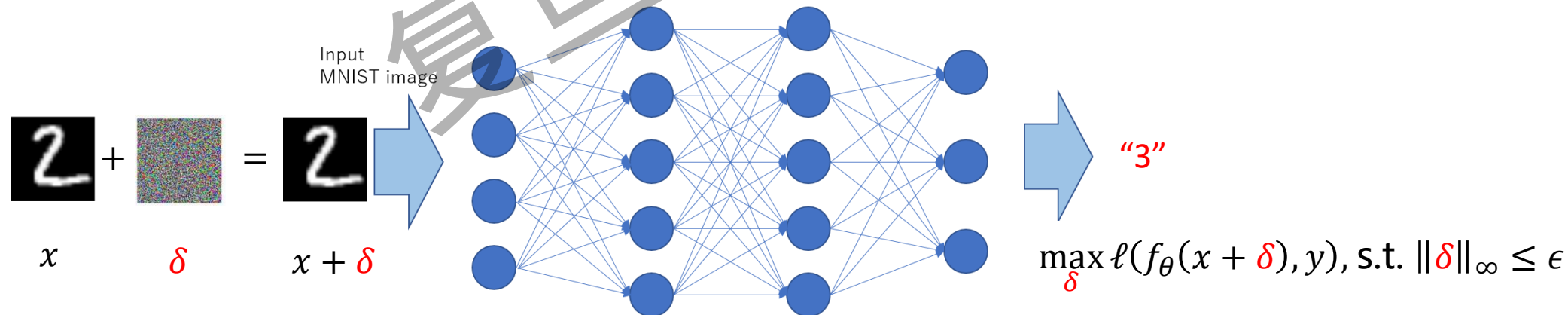
- 每个维度扰动绝对值都小于 ϵ

对抗样本：问题建模

- 对抗样本攻击目标：

- $$\max_{\delta} \ell(f_{\theta}(x + \delta), y), \quad \text{s.t. } \|\delta\|_{\infty} \leq \epsilon$$

- 如何完成上述优化？



回顾：最小化分类损失的解法

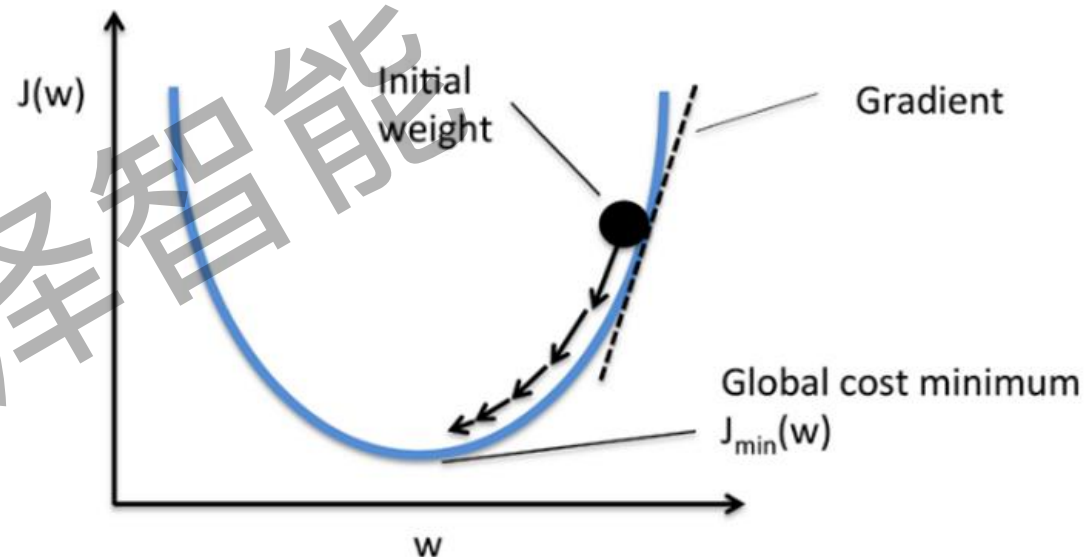
- 回忆梯度下降算法：

- $\min_{\theta} \ell(f_{\theta}(x), y)$

- 通过一步迭代：

- $\theta = \theta - \alpha \cdot \nabla_{\theta} \ell(f_{\theta}(x), y)$

- 可以离最优解越来越近，即最小化损失函数



对抗样本：扰动的优化

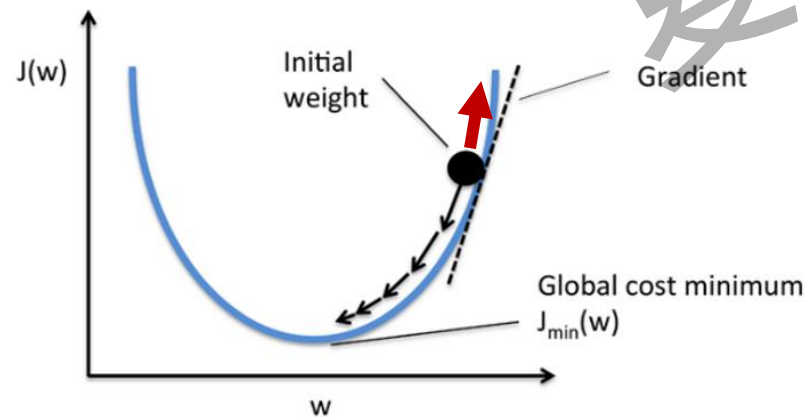
- 如果做反向的更新：

- $\theta = \theta + \alpha \cdot \nabla_{\theta} \ell(f_{\theta}(x), y)$

- 会离最优解越来越近

- 等同于在最大化损失函数

- $\max_{\theta} \ell(f_{\theta}(x), y)$



$$\max_{\delta} \ell(f_{\theta}(x + \delta), y), \quad \text{s.t.} \quad \|\delta\|_{\infty} \leq \epsilon$$

- 那如果把更新做到输入上呢？

- $\tilde{x} = x + \alpha \cdot \nabla_x \ell(f_{\theta}(x), y)$

- 就等同于在优化：

- $\max_x \ell(f_{\theta}(x), y)$

即可得到 $\delta = \alpha \cdot \nabla_x \ell(f_{\theta}(x), y)$

下一步是如何让 $\|\delta\|_{\infty} \leq \epsilon$

对抗样本：FGSM

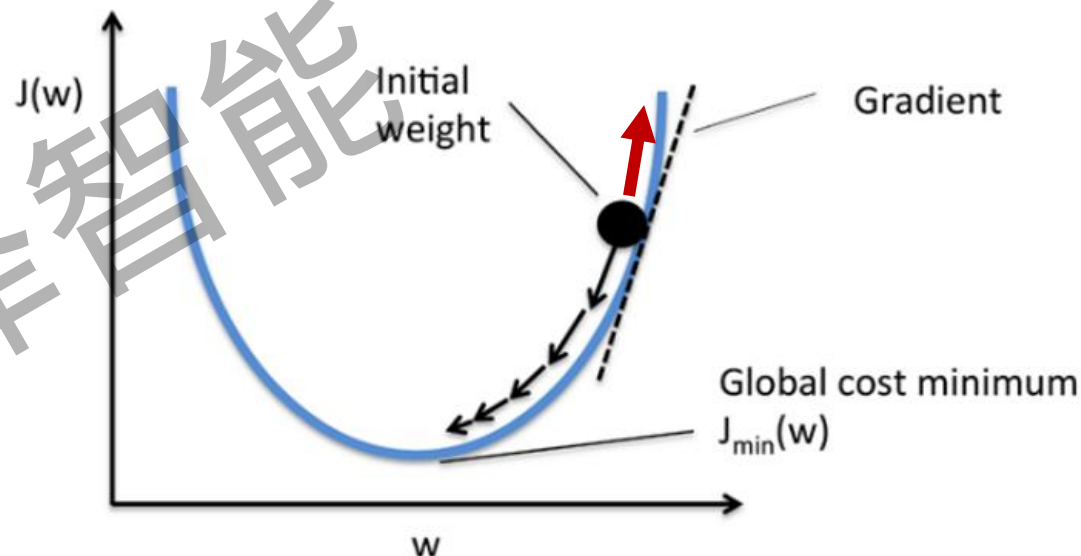
- 对抗样本攻击目标：

- $$\max_{\delta} \ell(f_{\theta}(x + \delta), y), \text{ s.t. } \|\delta\|_{\infty} \leq \epsilon$$

- 对抗样本生成过程：

- $$\delta = \epsilon \cdot \text{sign}(\nabla_x \ell(f_{\theta}(x), y))$$

- $$\text{sign}(\delta_i) = \begin{cases} +1, & \delta_i > 0 \\ -1, & \delta_i \leq 0 \end{cases}$$

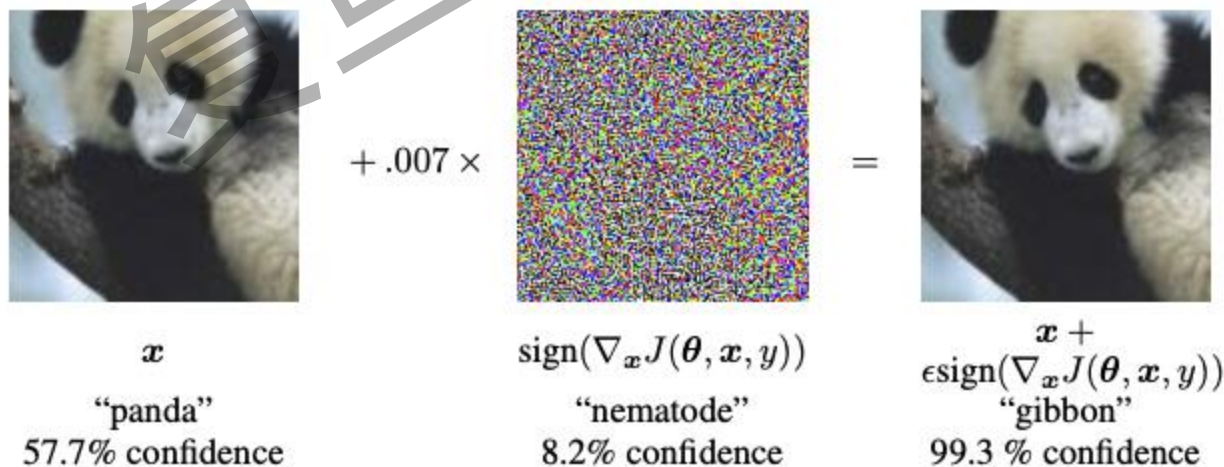


核心思想：

- 通过固定参数 θ ，对 x 做一步**反向**梯度下降，最大化损失函数
- 通过sign函数来满足对 δ 的约束

对抗样本：FGSM攻击流程

- 给定一个训练好的模型 f_θ
- 对于给定的样本 x 以及对应的标签 y ，生成对抗样本：
- $$\tilde{x} = x + \epsilon \cdot \text{sign}(\nabla_x \ell(f_\theta(x), y))$$
- 输入进模型，验证 $f_\theta(\tilde{x})$ 的预测结果





Q&A

复旦翻译智能

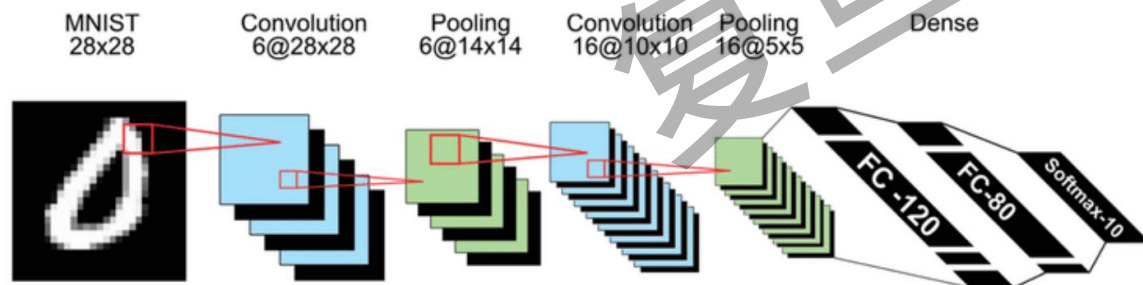


实验部分

CNN与对抗样本

任务1：实现卷积神经网络LeNet5

- 上一周我们在MNIST上实现了MLP
- 本周请在MNIST上实现LeNet5
 - 完成模型结构的定义
 - 完成模型的前向传播过程
- 尽可能按下图复现
 - 基于PyTorch内置模块



```
##### Model Definition #####
# TODO: Week 5, Task 1
class LeNet5(nn.Module):
    def __init__(self):
        super(LeNet5, self).__init__()
        # TODO: 定义模型的结构

    def forward(self, x):
        # TODO: 在这里定义前向传播过程。注意输入x的形状

        return 0
```

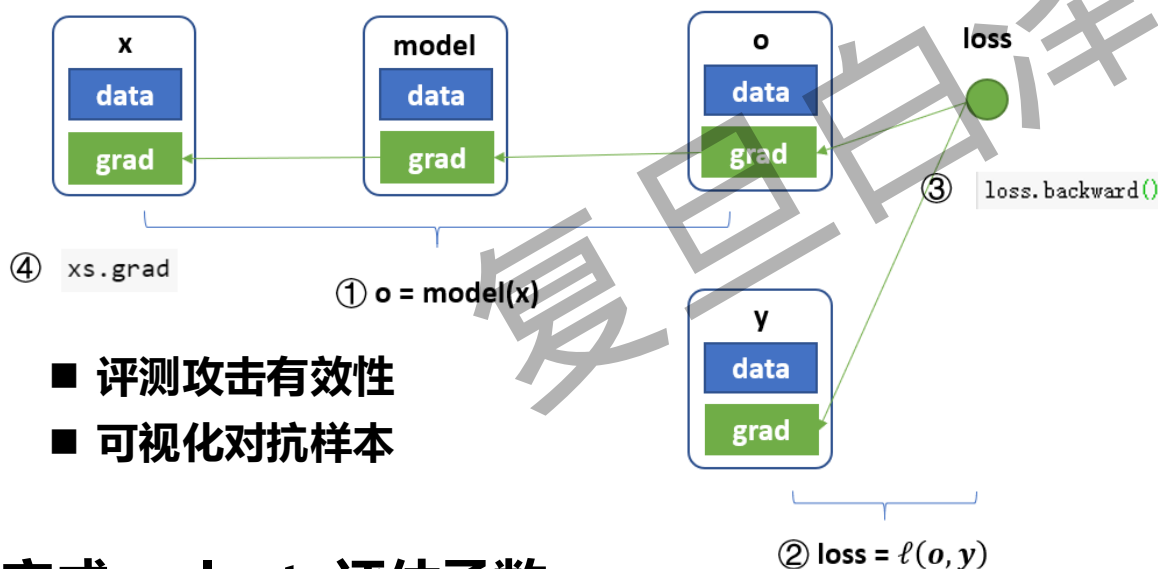
任务2：实现FGSM对抗攻击

■ 针对你训练好的LeNet5模型进行对抗攻击

■ 完成FGSM算法的实现

$$\tilde{x} = x + \epsilon \cdot \text{sign}(\nabla_x \ell(f_\theta(x), y))$$

`xs.requires_grad = True`



- 评测攻击有效性
- 可视化对抗样本

■ 完成evaluate评估函数

```
##### Adversarial Attacks #####
# TODO: Week 5, Task 2
def fgsm(imgs, epsilon, model, criterion, labels):
    model.eval()

    adv_xs = imgs.float()
    adv_xs.requires_grad = True

    # TODO: 模型前向传播, 计算loss, 然后反传

    # TODO: 得到输入的梯度、生成对抗样本

    # TODO: 对抗扰动做截断, 保证对抗样本的像素值在合理域内

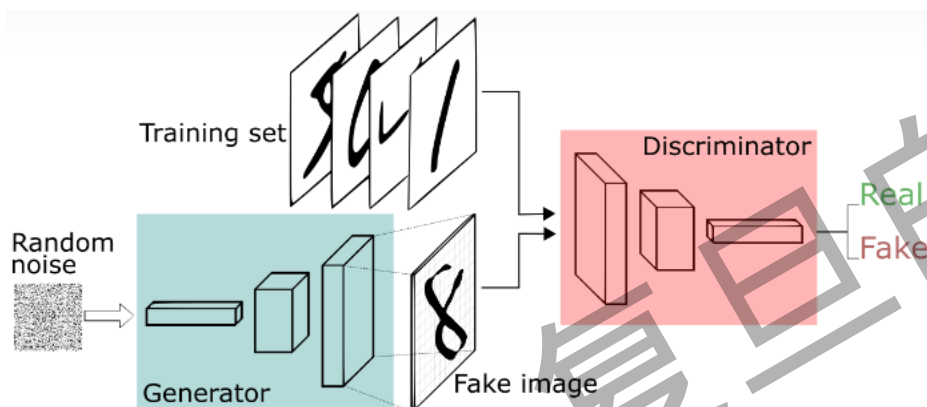
    model.train()

    return adv_xs.detach()
```

Bonus: 实现生成对抗网络GAN

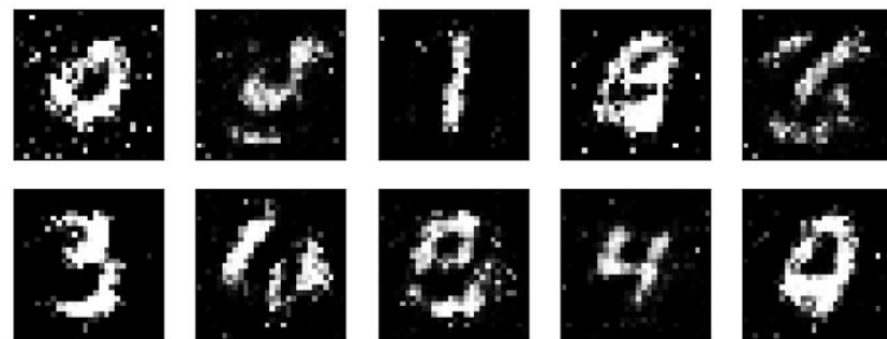
■ 在MNIST上实现GAN

- 定义Generator、Discriminator的结构与前向传播行为
- 制定两者的训练目标，并完成训练



$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

助教训了30个epoch后的生成图片：



- 自主学习相关材料，要求深刻理解！！



Q&A

復旦白澤智能