

在 R Markdown 文档中使用中文

谢益辉 邱怡轩 于淼

目录

1	GAN	1
2	DCGAN	3
2.1	步长卷积和微步卷积	4
2.2	批标准化	6
2.3	全连接层	8
2.4	激活函数	8
3	工作量评估	13

1 GAN

生成式对抗网络 GAN 是一种生成式模型，灵感来自于零和博弈的思想。整个系统由两部分组成，生成模型 G 和判别模型 D，通过两者的博弈，来使生成模型学习到数据的分布。生成模型仿照真实数据样本的分布，从一个随机噪声 z 中生成一个输入并训练自己骗过判别模型，使之认为其生成的输入都是真实的，而判别模型则试图区分真实输入与生成的输入。原始的 GAN 中，生成模型和判别模型都为多层感知机，结构如图 1。

GAN 的目标函数如图 2，是一个极小极大博弈问题。

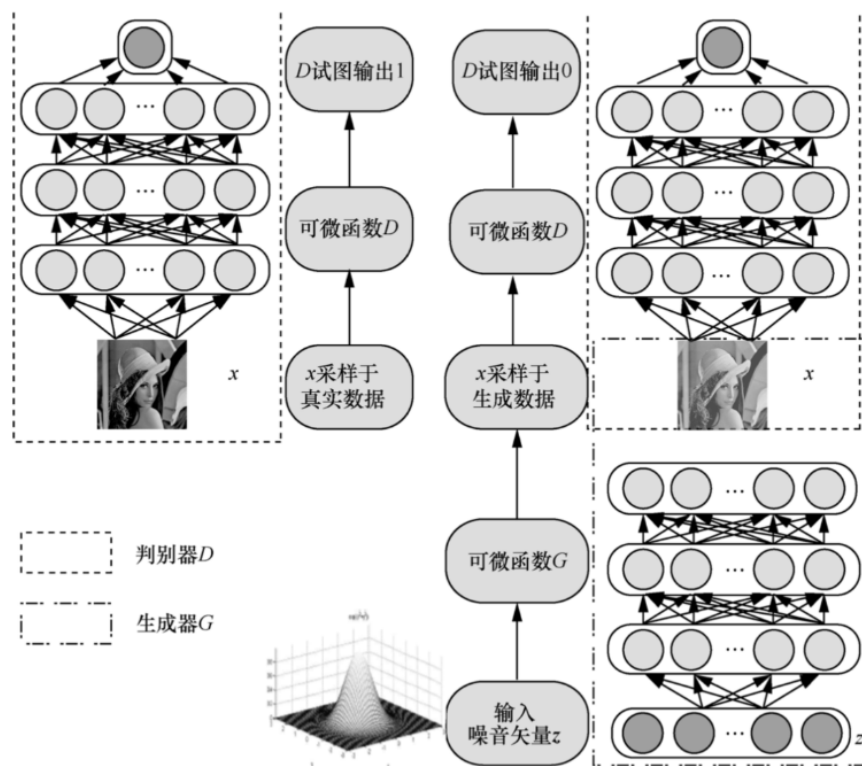


图 1: GAN 的结构

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

图 2: GAN 的目标函数

其中, P_{data} 是真实数据分布, P_z 是噪声分布, $G(z)$ 是生成模型根据随机噪声 z 生成的模拟真实数据的假数据, $D(x)$ 是判别模型判断真实数据为真的概率, $D(G(z))$ 是判别模型判断造假数据为真的概率。对于判别模型 D , 它的目标是将真实数据判断为真, 将生成模型生成的数据判断为假, 也就是使式子取值尽可能大。对于生成模型 G , 它的目标是生成的数据能够欺骗判别模型, 使式子取值尽可能小。在两个模型对抗的过程中, 各自的生成能力和判别能力都在提高。训练过程中, 交替地对生成模型和判别模型进行训练。固定生成模型 G , 通过梯度下降优化判别模型 D , 再固定判别模型 D , 优化生成模型 G , 直到达到纳什均衡。

2 DCGAN

在 DCGAN 的生成模型和判别模型中, 使用卷积神经网络 (CNN) 来代替传统 GAN 中的多层感知机。对于生成模型 G , 它的输入是一个 100 维的向量 z 。生成模型的第一层是一个全连接层, 将 100 维的向量变为 $4 \times 4 \times 1024$ 的输出, 从第二层开始, 使用微步卷积 (fractional-strided convolution) 来进行上采样, 逐渐减少通道数, 最后输出为 $64 \times 64 \times 3$ 的矩阵。生成模型的结构如图 3。

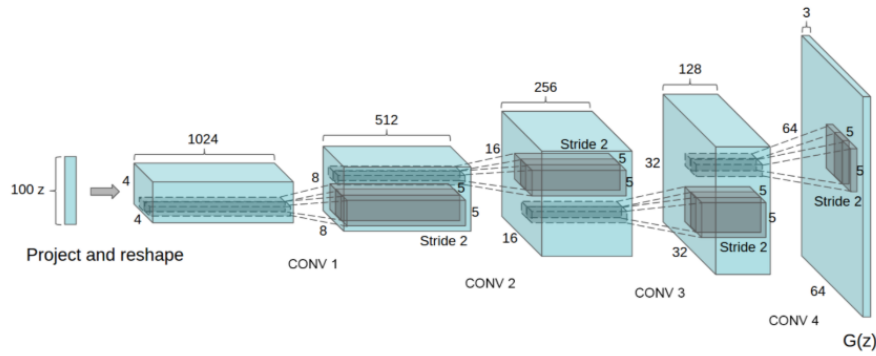


图 3: 生成模型的结构

判别模型 D 的输入是一张图像, 经过步长卷积下采样, 逐渐增加通道数, 最后得到的卷积特征通过全连接层, 输出一个值用来判断图像是真实数据或生成的数据, 结构如图 4。

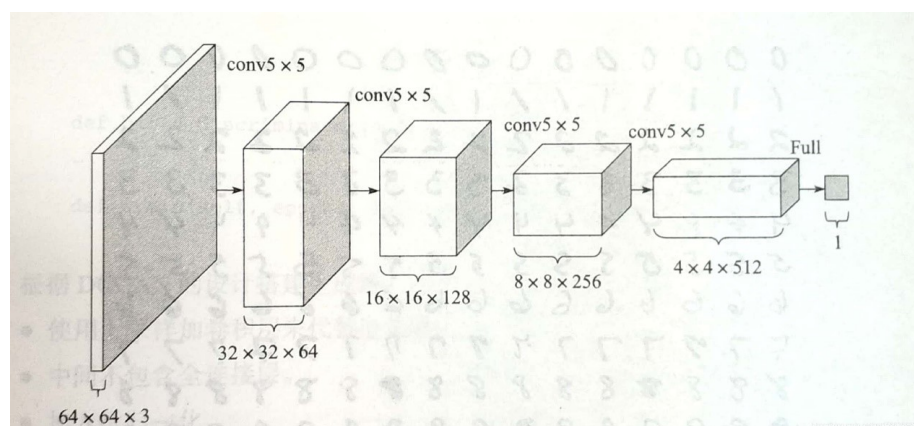


图 4: 判别模型的结构

2.1 步长卷积和微步卷积

在 DCGAN 中，没有使用池化层，而是以步长卷积来进行下采样。在生成模型中，使用了微步卷积（fractional-strided convolutions）来进行上采样。上采样即去卷积，目的是将经过池化层以后缩小的矩阵扩大到一定的大小，包含转置卷积和微步卷积两种方法。以将 3×3 的矩阵扩大到 5×5 为例，去卷积的过程如图 5。

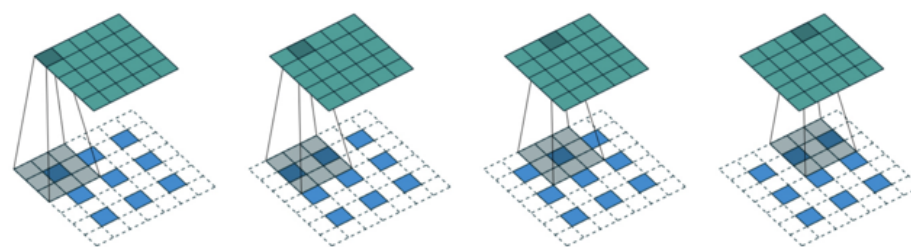


图 5: 去卷积

而转置卷积和微步卷积的主要差别在于填充的方式不同，如图 6。

在判别模型中，使用了步长卷积（strided convolution），步长即滤波器每次移动的距离。以 7×7 的矩阵为例，步长为 2 的情况下，最后会得到 3×3 的输出。如图 7。

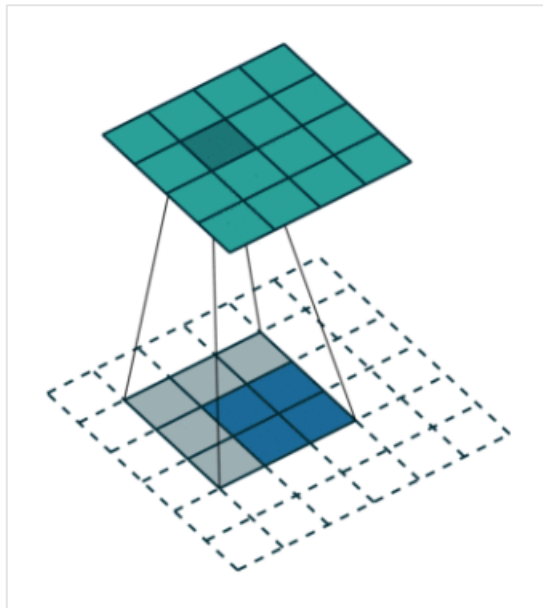


图4 转置卷积

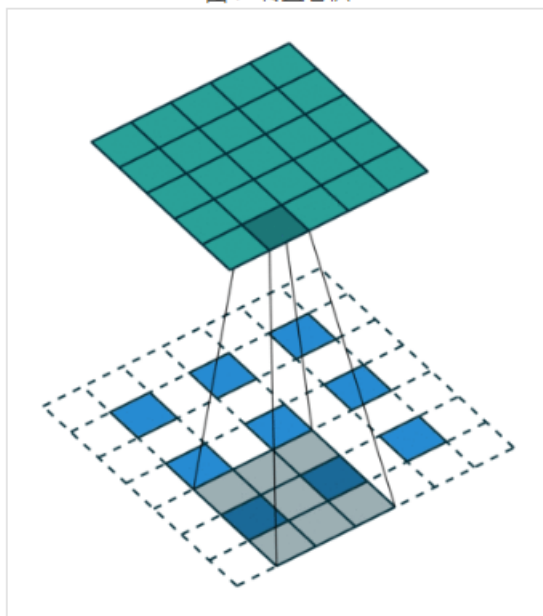
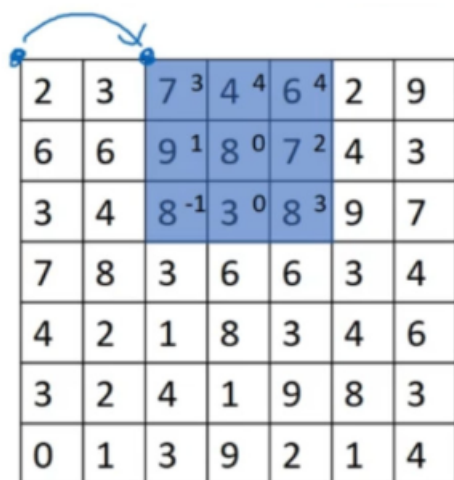


图5 微步卷积

图 6: 转置卷积和微步卷积



2	3	7 ³	4 ⁴	6 ⁴	2	9
6	6	9 ¹	8 ⁰	7 ²	4	3
3	4	8 ⁻¹	3 ⁰	8 ³	9	7
7	8	3	6	6	3	4
4	2	1	8	3	4	6
3	2	4	1	9	8	3
0	1	3	9	2	1	4

图 7: 步长卷积

2.2 批标准化

批标准化通过一定的规范化手段，将每一层的输入变换到均值为 0，标准差为 1 的标准正态分布，可以使训练变得稳定，帮助解决不好的初始化带来的问题，使梯度可以在更深的模型中传递。这是因为整体分布逐渐向非线性的激活函数的取值区间的上下限两端趋近时，会导致反向传播时底层神经网络梯度消失，而批标准化使激活函数输入值落座在对输入较敏感的区域。同时，梯度变大意味着学习收敛速度快，能加快训练速度。为了保证网络的表达能力，对变换后，满足均值为 0，标准差为 1 的 x 再次进行操作 $y = \text{scale} * x + \text{shift}$ ， scale 和 shift 两个参数通过训练学习到。批标准化的具体操作流程如图 8。

GAN 可能出现崩溃问题，生成器总是生成同样的样本点，无法继续学习。以图 9 为例，右图的十个团簇代表 mnist 数据集的 10 个模式，生成模型如果只能生成其中的几个数字而遗漏其他模式，便是出现了模式崩溃。批标准化可以避免这种问题。

在每一层应用批标准化会导致震荡和不稳定性，所以在生成模型的输出层和判别模型的输入层不使用批标准化。

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_1 \dots x_m\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{ mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{ mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{ scale and shift}$$

图 8: 批标准化

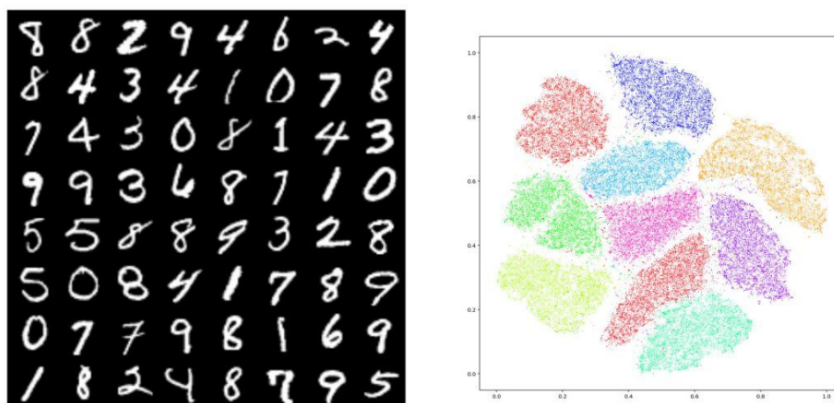


图 9: 模式崩溃

2.3 全连接层

长久以来，全连接层一直是 CNN 的标配结构，在常见的卷积神经网络中，传统的做法是在负责对图像进行特征提取的卷积层后设置全连接层，之后再继续进行激活分类。但近年来也有去除全连接层的趋势，比如使用全局平均池化来替代全连接层。

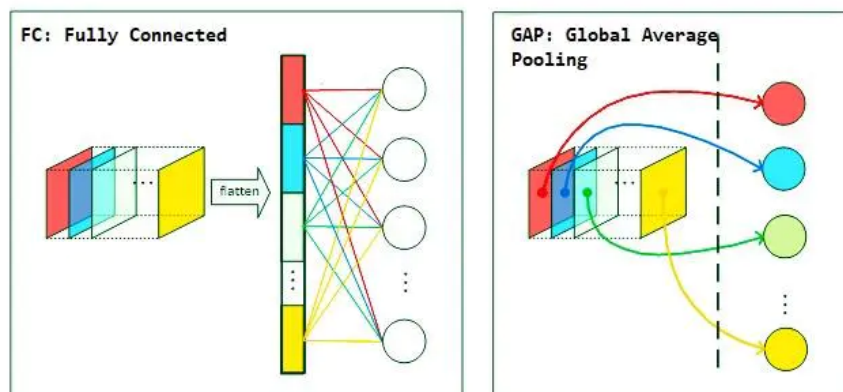


图 10: 全连接层

其主要思想即对于输出的每一个通道的特征图的所有像素计算一个平均值，用该数值代表对应的特征图。

相较于全连接层，全局平均池化可以接受任意尺寸的图像，降低了参数量，由于全局平均池化层没有参数，也可防止在该层过拟合，同时，整合了全局空间信息，还可以更好地将类别与最后一个卷积层的特征图对应起来。但在增加了模型稳定性的同时，全局平均池化有可能降低收敛速度。因此，最终选择了直接使用卷积层连接生成器和判别器的输入层以及输出层。

2.4 激活函数

激活函数的主要功能是提供网络的非线性建模能力，如果没有激活函数，网络只能表达线性映射，可以认为只有加入了激活函数之后深度神经网络才具备了分层的非线性映射学习能力。DCGAN 在生成器网络中选用 ReLU 作为激活函数，最后一层使用 Tanh。Tanh 即双曲正切函数，它的输出和输入

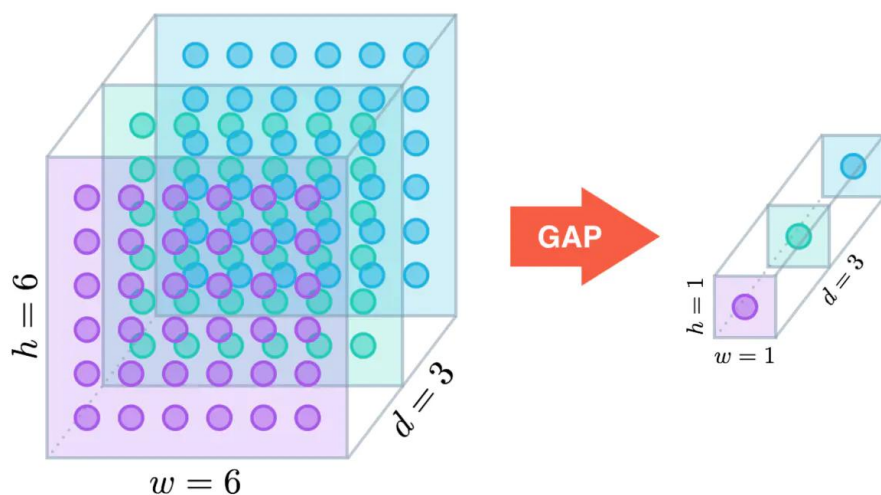


图 11: 全局平均池化

能够保持非线性单调上升和下降关系，输出以 0 为中心，比 Sigmoid 函数收敛速度更快，但还是没有改变 Sigmoid 函数最大的问题，即由于饱和性产生的梯度消失。

$$f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

图 12: tanh 激活函数

而 ReLU 有效缓解了梯度消失的问题，提供了神经网络的稀疏表达能力，且相较于 Tanh，在 SGD 中能快速收敛。

GAN 在判别器网络中选用了 Maxout 作为激活函数，它的拟合能力非常强，具有 ReLU 的所有优点，即线性、不饱和性，同时不会导致神经元死亡，但它会导致整体参数的数量激增。

与此相对，DCGAN 选用了 LeakyReLU 作为激活函数，LeakyReLU 也解决了神经元死亡的问题，同时，实现更为简单，在实验中表现也更好。

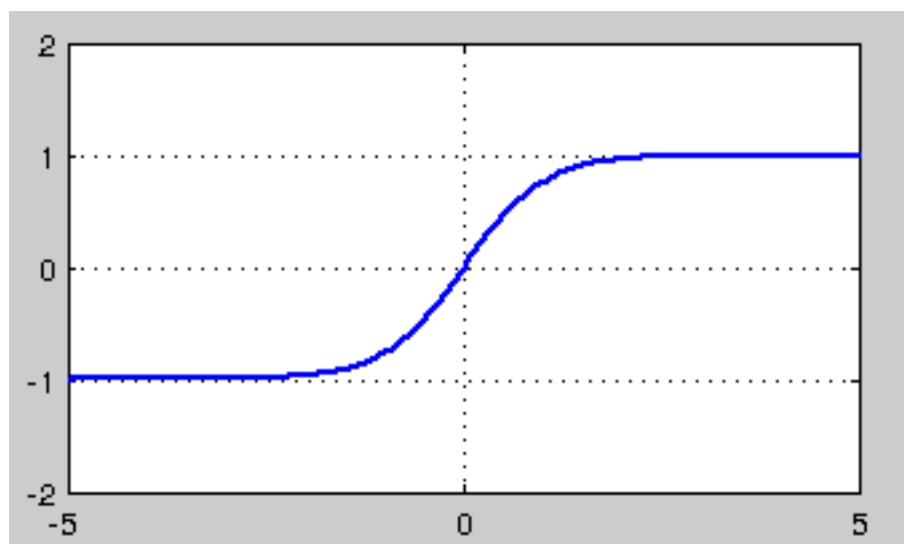


图 13: tanh 激活函数

$$f(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases}$$
$$f(x) = \max(0, x)$$

图 14: ReLU 激活函数

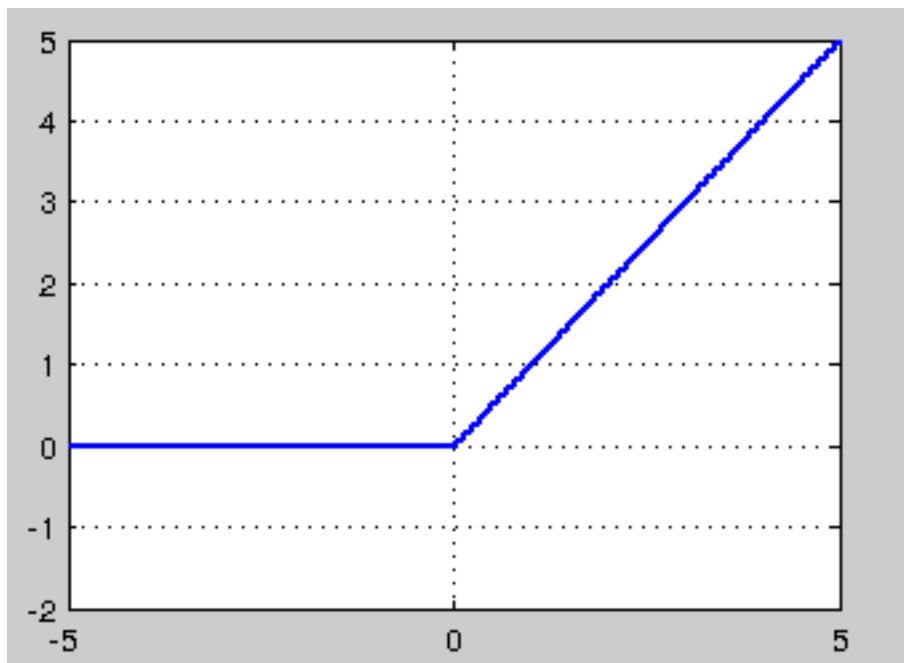


图 15: ReLU 激活函数

$$f(x) = \max(w_1^T x + b_1, w_2^T x + b_2, \dots, w_n^T x + b_n)$$

图 16: Maxout

$$f(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha(e^x - 1), & \text{if } x < 0 \end{cases}$$

图 17: LeakyReLU

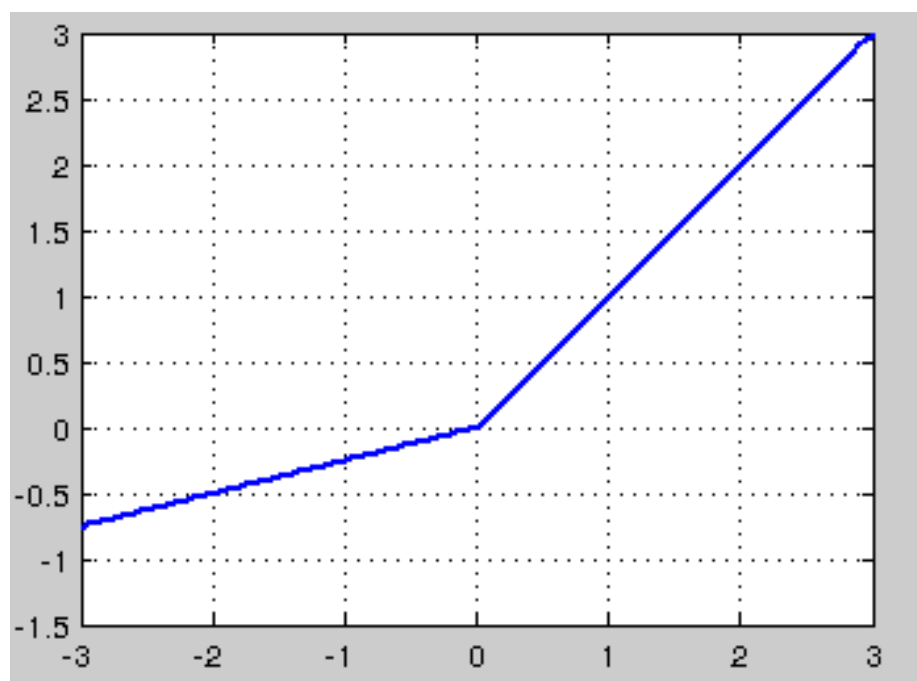


图 18: LeakyReLU

3 工作量评估

总工作量任务 | 工作量 (人天) | —— | —— | 使用 TensorFlow 搭建模型 | 6 | 获取和处理数据集 | 5 | 实现 Demo | 12

工作量分解任务 | 补充说明 | 需求分析 | 开发 | 测试 | —— | —— | —— | —— | —— | —— | 搭建生成模型 | 使用 TensorFlow 搭建 DCGAN 的生成器 | 无 | 王宸 | 张逸涵 | 搭建判别模型 | 使用 TensorFlow 搭建 DCGAN 的判别器 | 无 | 张岑媛 | 王宸 | 直接获取数据集 | 训练使用的数据集包括 LSUN 数据集, ImageNet 1k 和 celebA 数据集 | 无 | 石睿欣 | 宋普琦 | 构造数据集 | 爬取网上社区的图片, 通过 openface 进行修剪来构造数据集 | 无 | 张逸涵 | 石睿欣 | 预处理数据集 | 调整图像大小, 进行图像标准化处理 | 无 | 黄蕙茹 | 张宏年 | 训练模型 | 通过优化目标函数训练模型 | 无 | 宋普琦 | 张岑媛 | 调整模型参数 | 调整参数进行多次训练, 在验证集和测试集上进行测试来找到性能最好的参数 | 无 | 张宏年 | 宋普琦 | 实现 Demo 前端 | 前端需要实现一个网页, 主要功能为展示生成的图像, 并能显示其对应的输入 z ; 前端网页使用 vue 框架来实现 | 张逸涵 | 石睿欣 | 王宸 | 实现 Demo 后端 | 后端需要提供相应接口, 使网页能够获取图像与对应的输入 | 张岑媛 | 王宸 | 黄蕙茹

项目进度计划任务名称 | 耗时 (天) | 开始 | 结束 | —— | —— | —— | —— | DCGAN | 31 | 2020-9-26 | 2020-10-26 | 项目分析 | 5 | 2020-9-26 | 2020-9-30 | 搭建模型 | 7 | 2020-10-8 | 2020-10-14 | 获取与处理数据集 | 5 | 2020-10-10 | 2020-10-14 | 实现 Demo | 14 | 2020-10-13 | 2020-10-26