Activation Functions

1 激活函数作用

激活函数在神经网络中起着至关重要的作用，它们引入非线性因素，使得神经网络能够学习和表示复杂的函数。激活函数的主要作用包括：

- 引入非线性：激活函数使得神经元的输出不再是输入的线性组合，从而能够学习更复杂的特征。
- 控制输出范围：激活函数可以限制神经元的输出范围，例如 Sigmoid 函数将输出限制在 (0, 1) 之间。
- 提供梯度信息：在反向传播中，激活函数的导数提供了梯度信息，使得网络能够更新权重。

2 不同激活函数

2.1 Sigmoid 函数

Sigmoid 函数公式如下：

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

Sigmoid 函数图像如图1所示。Sigmoid 函数的输出范围在 (0, 1) 之间，常用于二分类问题的输出层。

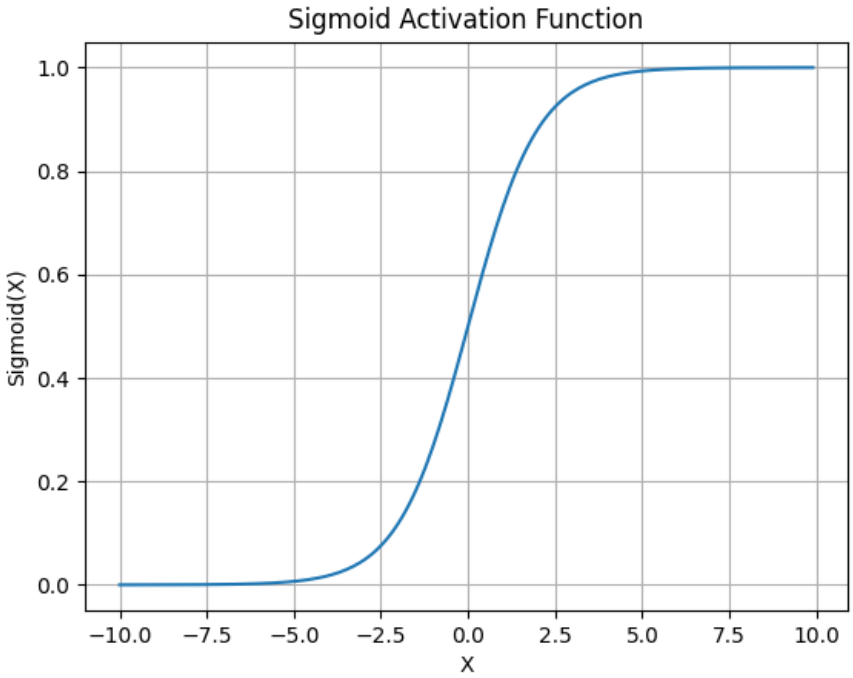

图 1: Sigmoid 函数的图像

2.2 Tanh 函数

Tanh 函数公式如下：

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

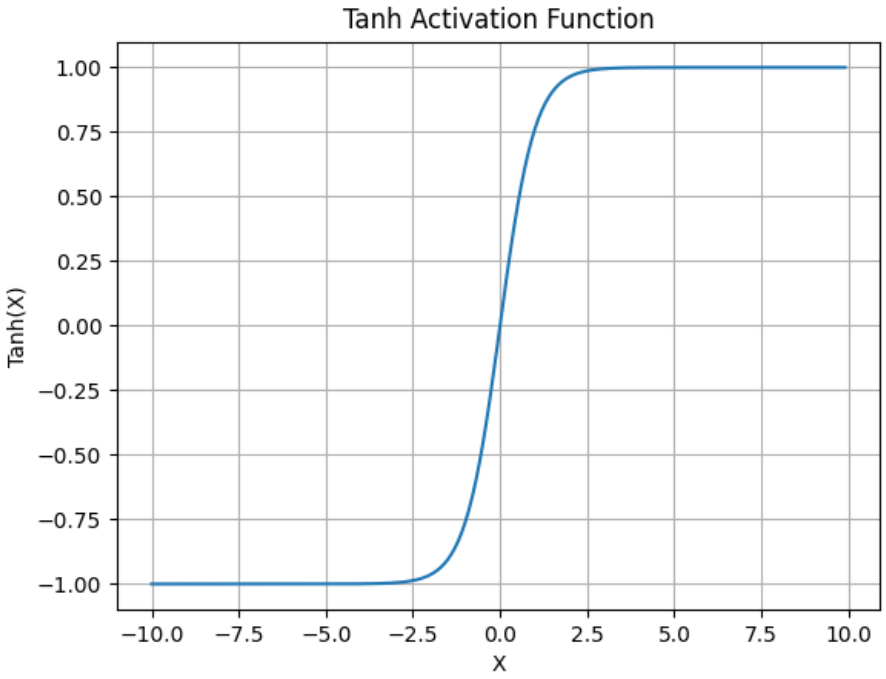

图 2: Tanh 函数的图像

Tanh 函数图像如图2所示。Tanh 函数的输出范围在 (-1, 1) 之间，相较于 Sigmoid 函数，Tanh 函数的输出范围更广，且在 0 附近的梯度更大，有助于缓解梯度消失问题。

2.3 ReLU 函数

ReLU 函数（Rectified Linear Unit）公式如下：

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$$

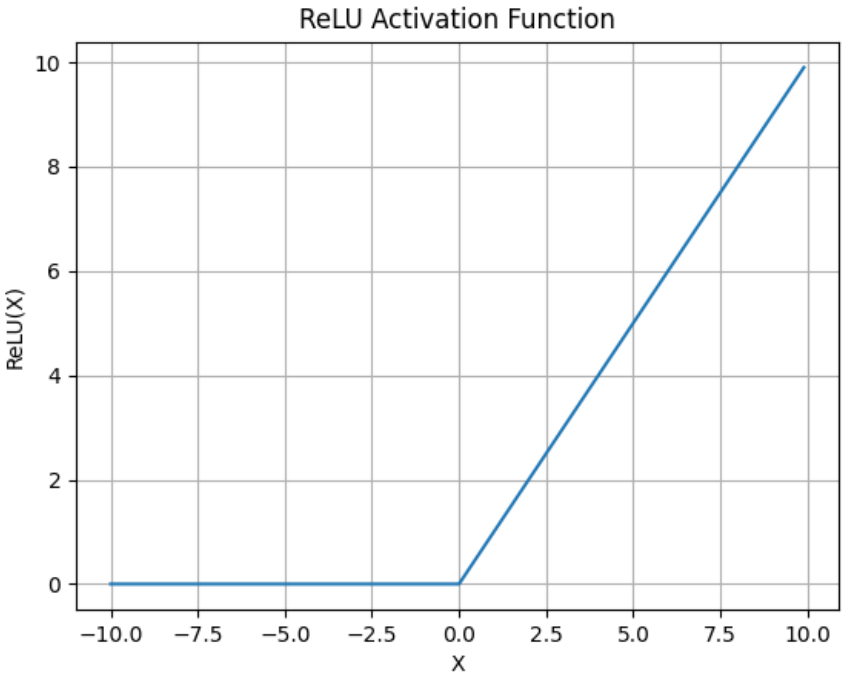

图 3: ReLU 函数的图像

ReLU 函数图像如图3所示。ReLU 函数的优点是计算简单，且在正区间具有较大的梯度，有助于加速网络的收敛。然而，ReLU 函数在负区间的梯度为 0，可能导致部分神经元“死亡”，即在训练过程中永远不会被激活。

2.4 Leaky ReLU 函数

Leaky ReLU 函数公式如下：

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases}$$

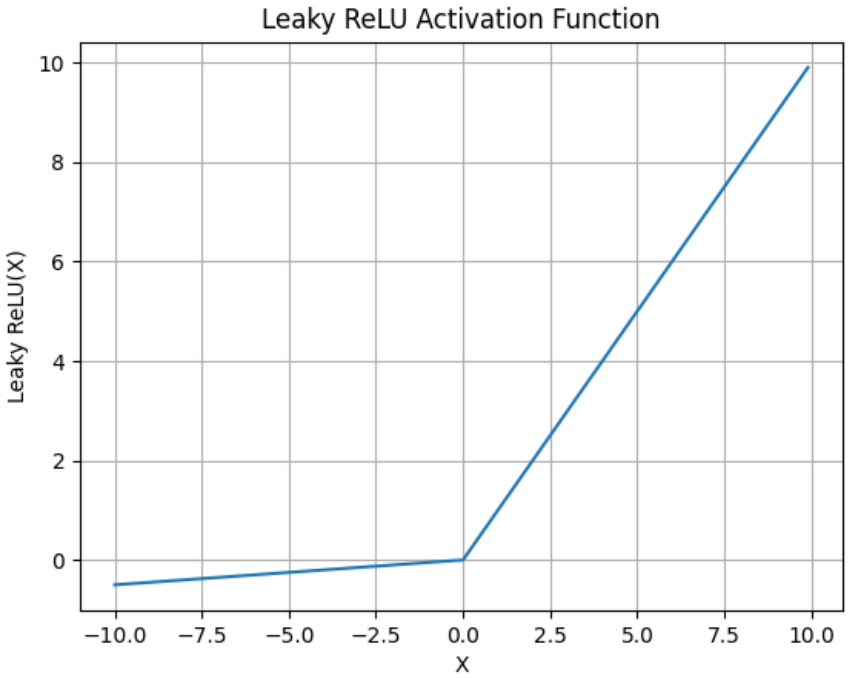

图 4: Leaky ReLU 函数的图像

Leaky ReLU 函数图像如图4所示，其中 $\alpha = 0.05$。Leaky ReLU 函数在负区间引入了一个小的斜率（通常为 0.01），从而避免了 ReLU 函数的“死亡”问题，使得神经元在负区间也能有非零梯度。

2.5 ELU 函数

ELU 函数公式如下：

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

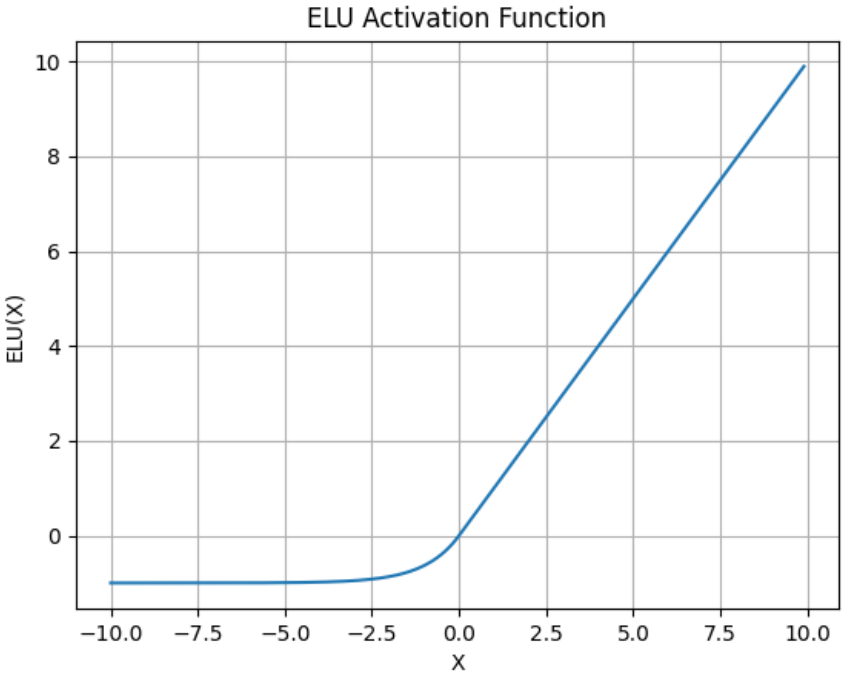

图 5: ELU 函数的图像

ELU 函数图像如图5所示，其中 $\alpha = 1$。ELU 函数在负区间的输出是指数函数的形式，这样可以使得负区间的输出更平滑，且在负区间也有非零梯度，从而避免了 ReLU 函数的“死亡”问题。

2.6 PRELU 函数

PRELU 函数（Parametric ReLU）是 Leaky ReLU 的一个变种，其公式如下：

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases}$$

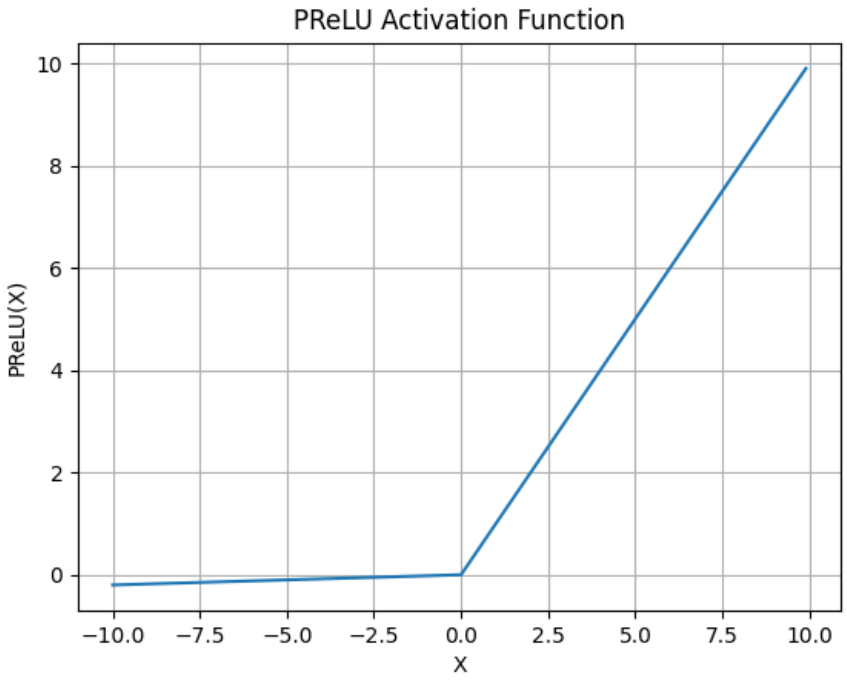

图 6: PRELU 函数的图像

PRELU 函数图像如图6所示，此处图中 $\alpha = 0.02$。与 Leaky ReLU 不同的是，PRELU 函数中的 α 是一个可学习的参数，这样可以使得网络在训练过程中自适应地调整负区间的斜率，从而提高模型的表达能力。

2.7 Swish 函数

Swish 函数公式如下：

$$f(x) = x \cdot \text{sigmoid}(x) = \frac{x}{1 + e^{-x}}$$

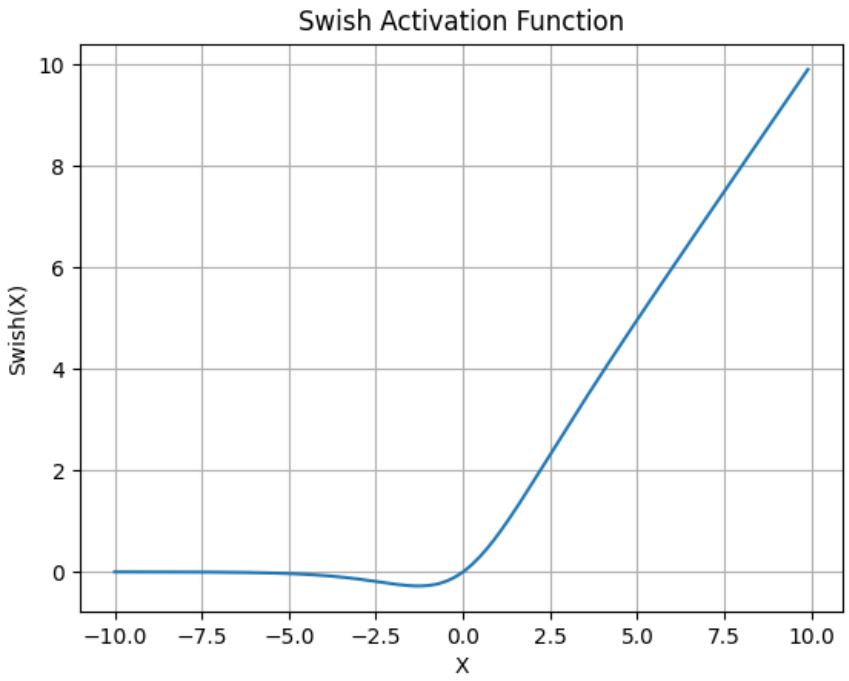

图 7: Swish 函数的图像

Swish 函数图像如图7所示。Swish 函数是一个平滑的非线性函数，其在正区间的输出与 ReLU 类似，但在负区间的输出更平滑，且具有非零梯度。Swish 函数在某些任务上表现优于 ReLU 和其他激活函数。

2.8 Softplus 函数

Softplus 函数公式如下：

$$f(x) = \ln(1 + e^x)$$

Softplus 函数图像如图8所示。Softplus 函数是 ReLU 函数的平滑版本，其在正区间的输出与 ReLU 类似，但在负区间的输出更平滑，且具有非零梯度。Softplus 函数在某些任务上表现良好，但计算复杂度较高。

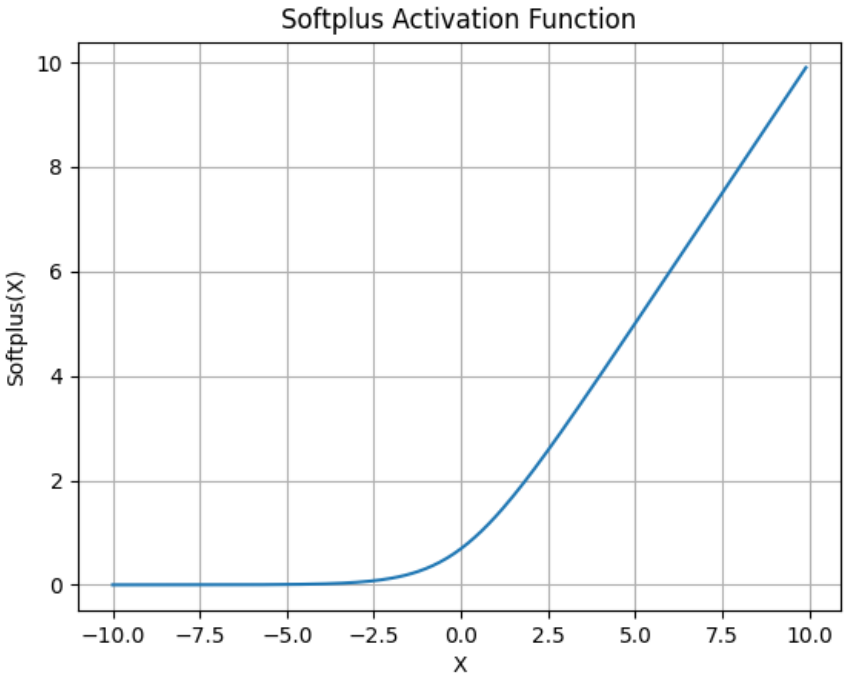

图 8: Softplus 函数的图像

2.9 LogSigmoid 函数

LogSigmoid 函数公式如下：

$$f(x) = \ln\left(\frac{1}{1+e^{-x}}\right) = -\ln(1+e^{-x})$$

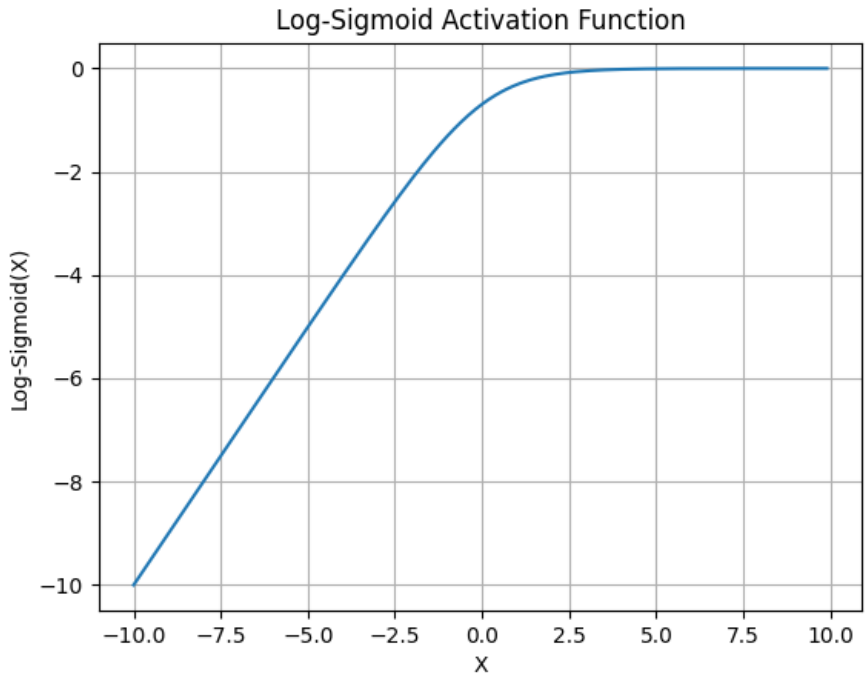

图 9: LogSigmoid 函数的图像

LogSigmoid 函数图像如图9所示。LogSigmoid 函数是 Sigmoid 函数的对数形式，其输出范围在 (-∞, 0) 之间。LogSigmoid 函数在某些任务上表现良好，尤其是在需要对数概率输出的场景中。

2.10 RReLU 函数

RReLU 函数（Randomized ReLU）是 Leaky ReLU 的一个变种，其公式如下：

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases}$$

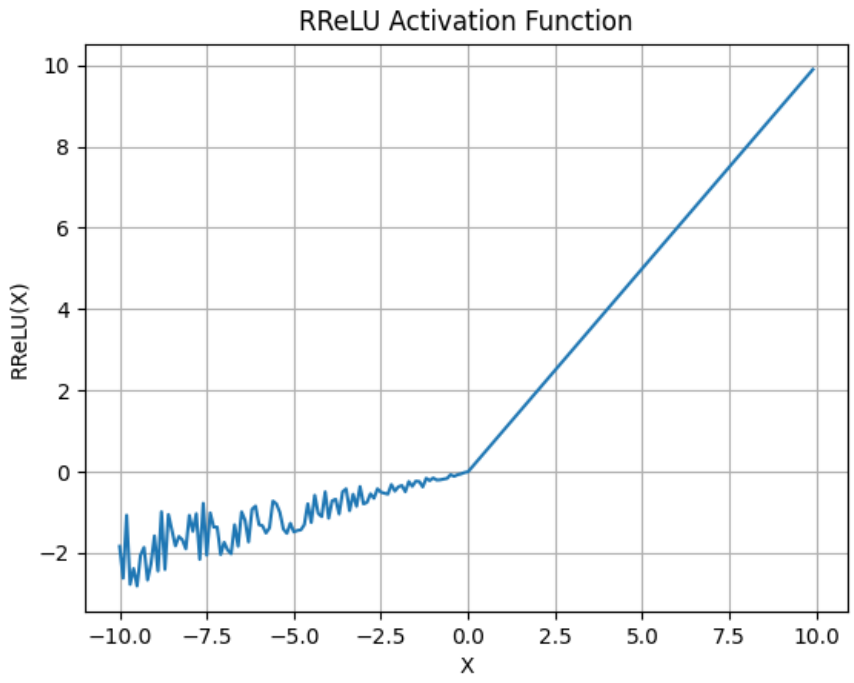

图 10: RReLU 函数的图像

RReLU 函数如图10所示。在训练过程中 α 是对均匀分布 $U(lower, upper)$ 进行随机采样，在评估阶段 α 取一个固定值，如 $\frac{lower+upper}{2}$。RReLU 函数的图像与 Leaky ReLU 类似，但在负区间的斜率是随机的，这样可以增加模型的多样性，防止过拟合。

2.11 SELU 函数

SELU 函数（Scaled Exponential Linear Unit）是 ELU 函数的一个变种，其公式如下：

$$f(x) = \lambda * (max(0, x) + min(0, \alpha * (e^x - 1))) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{if } x \leq 0 \end{cases}$$

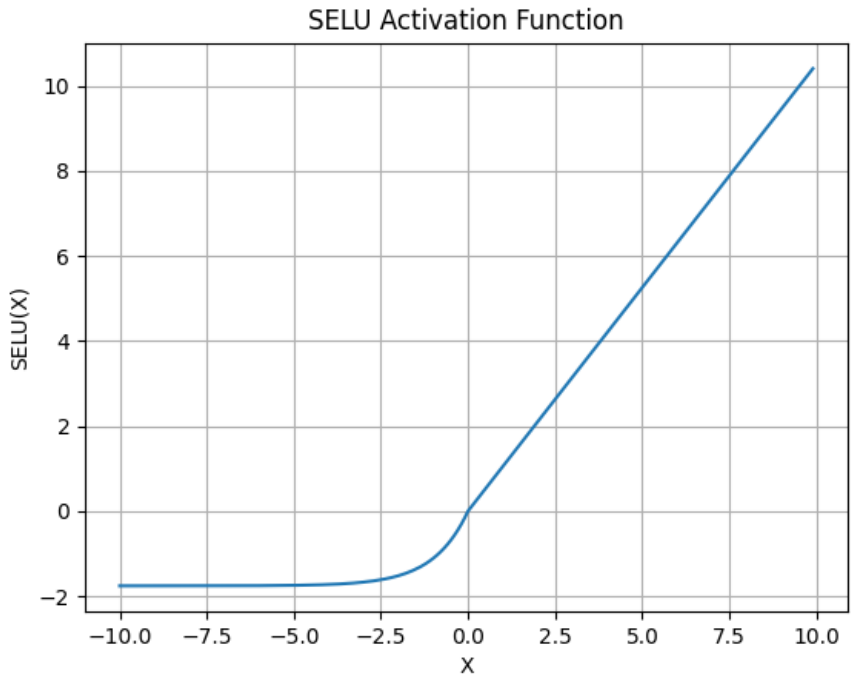

图 11: SELU 函数的图像

SELU 函数图像如图11所示，其中 $\lambda \approx 1.0507$ 和 $\alpha \approx 1.6733$。SELU 函数在正区间的输出与 ReLU 类似，但在负区间的输出更平滑，且具有非零梯度。SELU 函数的一个重要特点是它具有自归一化的性质，可以使得网络在训练过程中保持均值为 0 和方差为 1，从而加速收敛。

2.12 CELU 函数

CELU 函数（Continuously Differentiable Exponential Linear Unit）是 SELU 函数的一个变种，其公式如下：

$$f(x) = max(0, x) + min(0, \alpha(e^{\frac{x}{\alpha}} - 1)) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^{\frac{x}{\alpha}} - 1) & \text{if } x \leq 0 \end{cases}$$

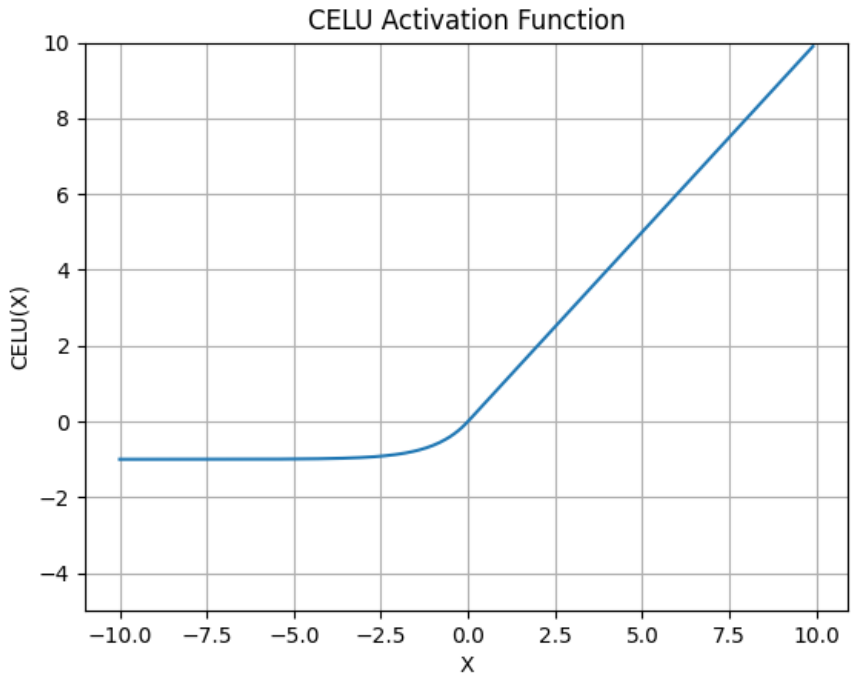

图 12: CELU 函数的图像

CELU 函数图像如图12所示，图中 $\alpha = 1.0$。CELU 函数在正区间的输出与 ReLU 类似，但在负区间的输出更平滑，且具有非零梯度。CELU 函数的一个重要特点是它具有连续可微性，可以使得网络在训练过程中更稳定。

2.13 GELU 函数

GELU 函数（Gaussian Error Linear Unit）是另一种激活函数，其公式如下：

$$f(x) = x \cdot \Phi(x) = x \cdot \frac{1}{2}\left(1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right)$$

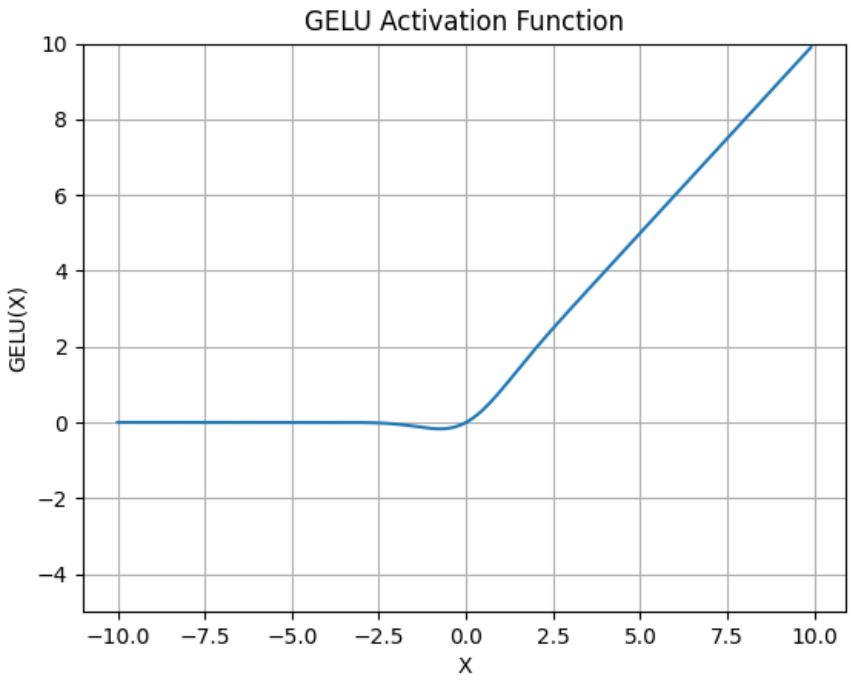

图 13: GELU 函数的图像

GELU 函数图像如图13所示，其中 $\Phi(x)$ 是高斯分布的累积分布函数。GELU 函数在正区间的输出与 ReLU 类似，但在负区间的输出更平滑，且具有非零梯度。GELU 函数的一个重要特点是它具有随机失活的性质，可以使得网络在训练过程中更稳定。

3 参考

激活函数的作用与分类

PyTorch 文档 - 非线性激活函数