ReLU Variants

Just Got Best Resource:

[ReLU Activation Function and Its Variants (pythonkitchen.com)](https://www.pythonkitchen.com/relu-activation-function-and-its-variants/)

https://www.pythonkitchen.com/relu-activation-function-and-its-variants/#:~:text=Variants%20of%20ReLU%201%201.%20Leaky%20ReLU%20%28LReLU%29,4%204.%20Scaled%20Exponential%20Linear%20Unit%20%28SELU%29%20

The ReLU is best choice for Hidden Layers there are many benefits

The Dying ReLU Problem

Kuch Neurons ka Output 0 Hojata hai, we call this neuron as Dead Neuron

If 50% se zyaada Neurons Dead Neuron Hojaye, because of this we will not able to Capture Patterns in Data Very Well

In Worst Case Scenario it can happen that 100% neurons are dead

So If Dying ReLU Problem Hua more than 50% of Neurons mai then capture nahi krpayenge Patterns ko

How Dying ReLU Problem Happens?

Agar kisi particular Neuron mai Weighted Sum 0 hojaye then

ReLU ka Derivative 0 hojata hai

& ReLU ka Derivative is used in Update Rule so that’s why Update 0 hojata hai

& Then Weights Update hote nahi hai.

The "dying ReLU" problem is a phenomenon that can occur during the training of neural networks using the Rectified Linear Unit (ReLU) activation function. The problem is characterized by certain neurons always outputting zero for all inputs during training, effectively becoming "dead" and not contributing to the learning process. Here are the key aspects of the dying ReLU problem:

1. **Zero Output for Negative Inputs:**
   - ReLU sets negative input values to zero. If a neuron consistently receives negative inputs during training, it may always output zero, and its weights may not be updated.
2. **Dead Neurons:**
   - Once a neuron becomes inactive (outputs zero for all inputs), it is often referred to as a "dead neuron." Dead neurons do not participate in the learning process and essentially become useless in terms of contributing to the model's representation.
3. **Negative Gradients:**
   - During backpropagation, if the gradient flowing through a ReLU neuron is consistently negative, the weights associated with that neuron may be updated in such a way that the neuron always outputs zero.
4. **Saturated Neurons:**
   - Neurons that always output zero are effectively saturated, and the issue is analogous to the vanishing gradient problem observed with the sigmoid and tanh activation functions.
5. **Layer Impacts:**
   - The impact of the dying ReLU problem can extend beyond individual neurons. If a substantial portion of neurons in a layer becomes inactive, it can affect the overall representational capacity of that layer, potentially limiting the expressive power of the network.

The "dying ReLU" problem occurs during the training of neural networks when certain ReLU (Rectified Linear Unit) neurons become inactive and consistently output zero for all inputs. This phenomenon can be attributed to several factors, and understanding how it happens involves considering both the characteristics of the ReLU activation function and the dynamics of the training process. Here are some key factors contributing to the dying ReLU problem:

1. **Zero Output for Negative Inputs:**
   - ReLU activation sets all negative input values to zero. If a neuron consistently receives negative inputs during training, its output will always be zero.
2. **Vanishing Gradient:**
   - During backpropagation, the gradients flowing through a ReLU neuron are dependent on the input. If the input to a ReLU neuron is consistently negative, the gradient for that neuron becomes zero. Consequently, the weights associated with that neuron may not be updated, leading to the neuron's inability to learn.
3. **Incorrect Weight Initialization:**
   - Poor weight initialization can contribute to the dying ReLU problem. If weights are initialized in such a way that a significant number of neurons receive predominantly negative inputs, those neurons may become inactive and fail to update their weights during training.
4. **Inappropriate Learning Rates:**
   - Inappropriate learning rates can lead to convergence issues. If the learning rate is too high, it might cause weights to be updated excessively, leading to oscillations and convergence problems. On the other hand, if the learning rate is too low, the weights may not be updated sufficiently, potentially contributing to the dying ReLU problem.
5. **Data Distribution:**
   - The nature of the input data can also play a role. If a large portion of the training data leads to negative inputs for certain neurons, those neurons may become inactive.
6. **Network Architecture:**
   - The architecture of the neural network, including the number of layers and the number of neurons in each layer, can influence the occurrence of the dying ReLU problem. Deeper networks may be more susceptible, especially if weights are not initialized properly.

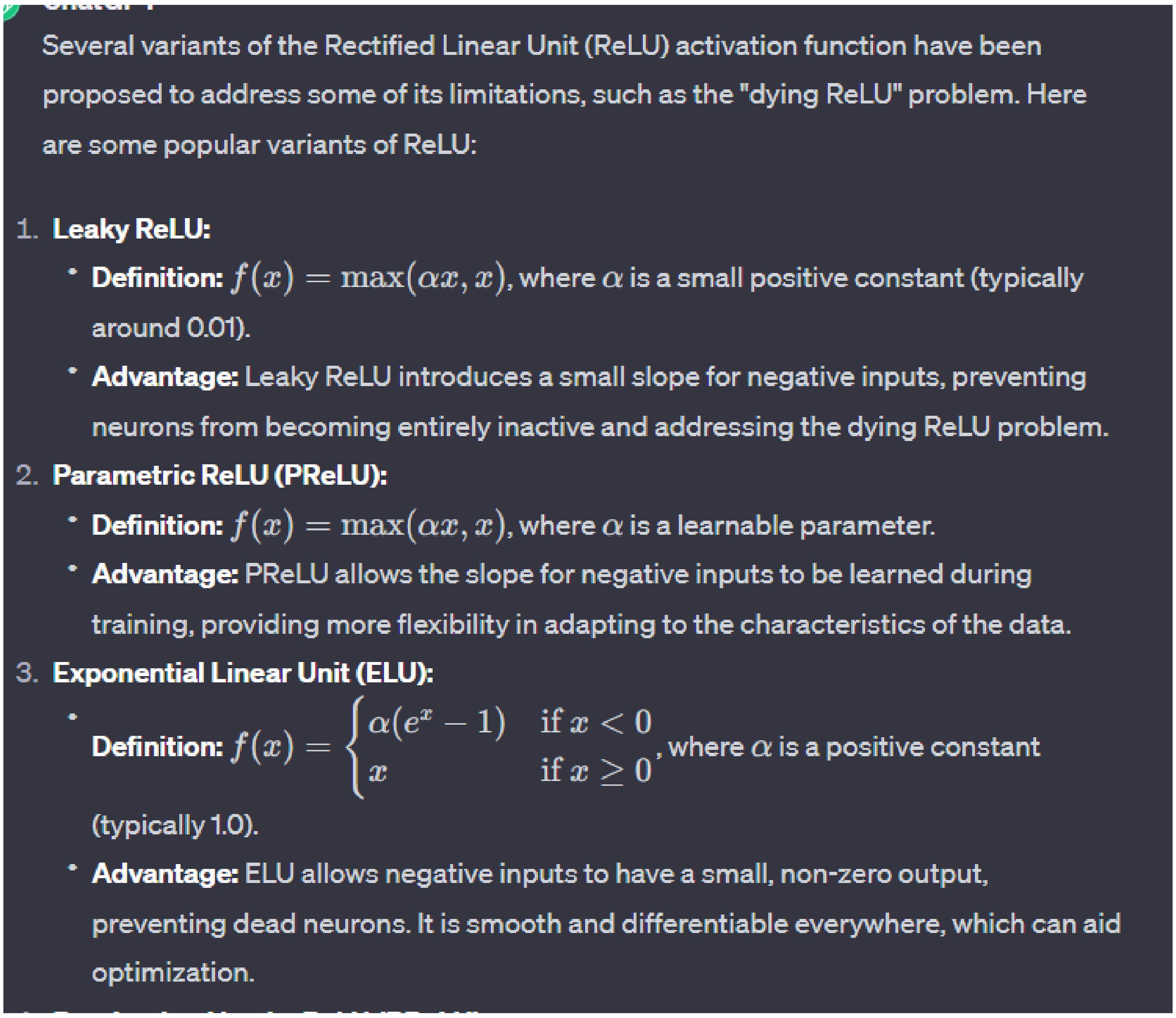
The main issue with ReLU is that all the negative values become zero immediately, which decreases the ability of the model to fit or train from the data properly.

That means any negative input given to the ReLU activation function turns the value into zero immediately in the graph, which in turn affects the resulting graph by not mapping the negative values appropriately. This can however be easily fixed by using the different variants of the ReLU activation function, like the leaky ReLU and other functions discussed earlier in the article.

Solutions

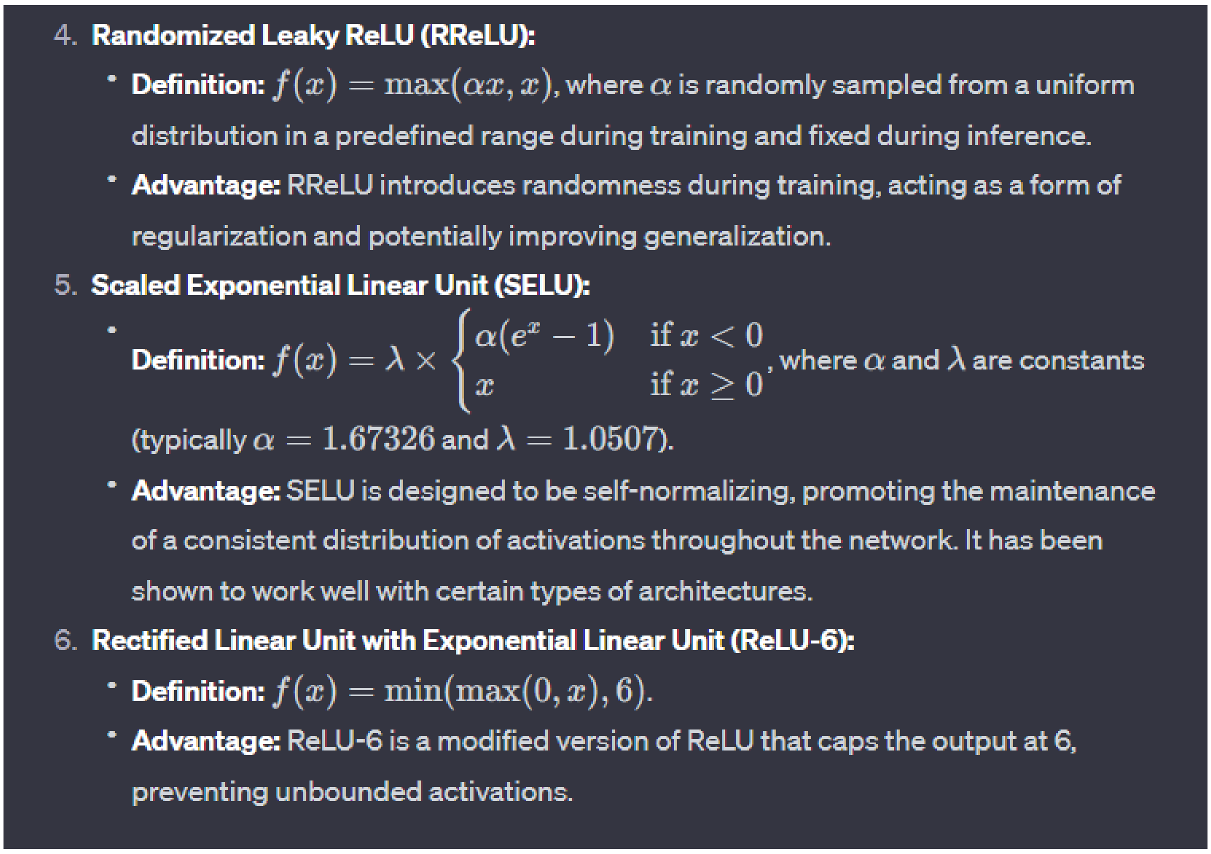
1.) Set Low Learning Rates
2.) Set Bias with positive Value -> 0.01 Experimentally proven
3.) Don't use ReLU , use ReLU Variants

ReLU esa function hai jo jaise hie negative hota hai then RelU ka derivative bhi negative hota hai & ReLu derivative is used in update rule becoz of which whole term is becoming 0.



Several variants of the Rectified Linear Unit (ReLU) activation function have been proposed to address some of its limitations, such as the "dying ReLU" problem. Here are some popular variants of ReLU:

1. **Leaky ReLU:**
   - **Definition:** $f(x) = \max(\alpha x, x)$, where α is a small positive constant (typically around 0.01).
   - **Advantage:** Leaky ReLU introduces a small slope for negative inputs, preventing neurons from becoming entirely inactive and addressing the dying ReLU problem.
2. **Parametric ReLU (PReLU):**
   - **Definition:** $f(x) = \max(\alpha x, x)$, where α is a learnable parameter.
   - **Advantage:** PReLU allows the slope for negative inputs to be learned during training, providing more flexibility in adapting to the characteristics of the data.
3. **Exponential Linear Unit (ELU):**
   - **Definition:** $f(x) = \begin{cases} \alpha(e^x - 1) & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$, where α is a positive constant (typically 1.0).
   - **Advantage:** ELU allows negative inputs to have a small, non-zero output, preventing dead neurons. It is smooth and differentiable everywhere, which can aid optimization.



4. **Randomized Leaky ReLU (RReLU):**
   - **Definition:** $f(x) = \max(\alpha x, x)$, where α is randomly sampled from a uniform distribution in a predefined range during training and fixed during inference.
   - **Advantage:** RReLU introduces randomness during training, acting as a form of regularization and potentially improving generalization.
5. **Scaled Exponential Linear Unit (SELU):**
   - **Definition:** $f(x) = \lambda \times \begin{cases} \alpha(e^x - 1) & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$, where α and λ are constants (typically $\alpha = 1.67326$ and $\lambda = 1.0507$).
   - **Advantage:** SELU is designed to be self-normalizing, promoting the maintenance of a consistent distribution of activations throughout the network. It has been shown to work well with certain types of architectures.
6. **Rectified Linear Unit with Exponential Linear Unit (ReLU-6):**
   - **Definition:** $f(x) = \min(\max(0, x), 6)$.
   - **Advantage:** ReLU-6 is a modified version of ReLU that caps the output at 6, preventing unbounded activations.

Variants of ReLu

1.Linear Variants

2.Non Linear Variants

1.Linear Variants

Linear variants mai hum linear transformation lagate hai,

a.) ***Leaky ReLu***

b.) ***Parametric ReLu***

2.Non Linear

a.) Elu

b.) Selu

<u>Leaky ReLU</u>