Weight Initialization Training Stability in MLPs

Parth Agrawal

December 2025

theory-of-signal-propagation-weight-initialization

1 Theory of Signal Propagation & Weight Initialization

When we go with backpropagation, the gradients may *vanish* by becoming too small or may *explode* by becoming too large. To prevent both from happening throughout the signal propagation, we need to maintain consistent activation functions and gradients.

For a linear layer $y = Wx + b$, if we assume inputs x and weights W are independent and centered at zero:

$$Var(y) = n_{in} \cdot Var(w) \cdot Var(x)$$

where n_{in} is the number of input connections (fan-in). To maintain stability (i.e., $Var(y) = Var(x)$), we must satisfy:

$$n_{in} \cdot Var(w) = 1 Var(w) = \frac{1}{n_{in}}$$

Thus, we need a good choice of initialization and activation functions.

xavier-glorot-initialisation

2 Xavier (Glorot) Initialisation

This Initialization is used when activation functions like **sigmoid** or **tanh** are used. These functions are symmetric around zero but susceptible to saturation.

In short, Xavier initialization is a method that maintains the same variance for x and y.

Xavier initialization balances the variance for both the forward pass (fan-in) and backward pass (fan-out).

$$Var(W) = \frac{2}{n_{in} + n_{out}}$$

Weights are typically drawn from a uniform distribution within the range:

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_{in}+n_{out}}}, \frac{\sqrt{6}}{\sqrt{n_{in}+n_{out}}}\right]$$

Without specific variance signals in the sigmoid, the tanh function tends to shrink towards zero, causing a vanishing gradient.

he-kaiming-initialization

3 He (Kaiming) Initialization

He Initialization is the standard method used in cases of ReLU activation functions.

Because the ReLU function kills half the signal (outputting 0 for negative inputs), it effectively halves the variance of the forward pass. To compensate, He Initialization doubles the variance compared to the standard requirement:

$$Var(W) = \frac{2}{n_{in}}$$

Weights are initialized from a Normal distribution:

$$W \sim N\left(0, \sqrt{\frac{2}{n_{in}}}\right)$$

Xavier initialization is too small for ReLU functions, resulting in signal attenuation in later layers. He Initialization prevents that from happening.

the-dying-relu-problem

4 The Dying ReLU problem

Since the ReLU function sets all negative values to zero, this may cause some neurons to become permanently disabled.

The ReLU function is defined as $f(x) = \max(0, x)$. Its derivative is:

$$f'(x) = \{\, 1\ if x > 00 if x \leq 0$$

If a neuron's weights are updated such that the weighted sum of inputs ($Wx+b$) is consistently negative for all training examples, the gradient becomes 0. The weights will never update again, and the neuron "dies."

Leaky ReLU introduces a slight slope α (e.g., 0.01) for negative values:

$$f(x) = \max(\alpha x, x)$$

This ensures that even for negative inputs, a slight gradient flows back, allowing the neuron to recover.

According to some, using a parametric ReLU is more effective when negatives are multiplied by a specific slope.

gradient-clipping

5 Gradient Clipping

To prevent exploding gradients, which occur when the gradient product exceeds 1, we clip it.

We can either clip by a particular threshold value or by clipping by norm, which is preferred.

Clipping by Norm Rescales the entire gradient vector g if its Euclidean norm $\|g\|$ exceeds a threshold C. This preserves the direction of the gradient while limiting its magnitude.

$$if \|g\| > C g \leftarrow g \cdot \frac{C}{\|g\|}$$

This helps gradient descent to have reasonable behaviour even if the loss landscape of the model is irregular, most likely a cliff.

softmax-and-numerical-stability

6 Softmax and Numerical Stability

The softmax function is typically used as an activation function in the final layer to convert outputs into probabilities.

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Calculating e^{z_i} when z_i is large (e.g., 100) causes floating-point overflow. The Fix (Shift Invariance): Subtract the maximum value $C = \max(z)$ from all inputs before exponentiation.

$$\sigma(z)_i = \frac{e^{z_i - C}}{\sum_{j=1}^{K} e^{z_j - C}}$$

This ensures the largest exponent is $e^0 = 1$, preventing overflow without changing the output probabilities.

The softmax function was introduced in statistical mechanics through the Boltzmann distribution in Boltzmann's foundational paper (1868), which was formalized and popularised in Gibbs' influential textbook (1902).

sources

7 Sources

- Joshi, P. (2016, March 29). Understanding Xavier Initialization in Deep Neural Networks. Prateek V Joshi. Retrieved from https://prateekvjoshi.com/2016/03/29/understanding xavier-initialization-in-deep-neural-networks/
- Hlav, E. (2023, February 15). Kaiming He Initialization in Neural Networks – Math Proof. Towards Data Science. Retrieved from https://towardsdatascience.com/kaiming-he-initialization-in-neural-networks-math-proof-73b9a0d845c4/
- Educative, Inc. (n.d.). What is the dying ReLU problem? Educative. Retrieved from https://www.educative.io/answers/what-is-the-dying-relu-problem

- Bajaj, A. (2025, May 8). Understanding Gradient Clipping (and How It Can Fix Exploding Gradients Problem). Neptune.ai. Retrieved from https://neptune.ai/blog/understanding-gradient-clipping-and-how-it-can-fix-exploding-gradients-problem