Title: The Effect of Weight Initialisation on MLP Training Stability and Performance

Dataset Used: MNIST

Student Id – 24098812 **Github:** https://github.com/Chandrikamadithati/Machine-learning

Introduction:

Setting the weights of deep neural networks is important for optimisation. Even though modern models use powerful optimisers like Adam and RMSProp, a network's inherent capacity to learn still largely depends on how the weights are initialized before training commences. Poor initialisation can cause vanishing gradients, exploding gradients, and slow or unstable convergence. The problems arise in deep networks with nonlinear activations (notably ReLU).

The impact of using three common initialisation strategies on convergence during training and its generalisation performance is investigated in this report.

1. Random Normal Initialisation.
2. Xavier (Glorot) Initialisation.
3. He Initialisation.

A MLP with three hidden layers is trained on the MNIST dataset, keeping all components of the architecture the same apart from the initialisation method. The research isolates the initialisation strategy to highlight the influence of weight scaling on gradient propagation, loss behaviour and accuracy.

Dataset Overview:

The MNIST dataset contains 70,000 images of handwritten digits (60,000 training and 10,000 test). Each image is:

- 28×28 pixels,
- grayscale,
- labelled 0–9.

MNIST remains a widely adopted benchmark because it is lightweight, easy to preprocess, and exhibits enough complexity to demonstrate training behaviour differences clearly - especially in deep fully connected networks.

Model Architecture

A three-hidden-layer MLP is constructed with:

- Dense(256) → ReLU
- Dense(256) → ReLU
- Dense(128) → ReLU
- Output Dense(10) → Softmax

Only the kernel initializer is changed between experiments:

- Model A: Random Normal

- Model B: Xavier Normal
- Model C: He Normal

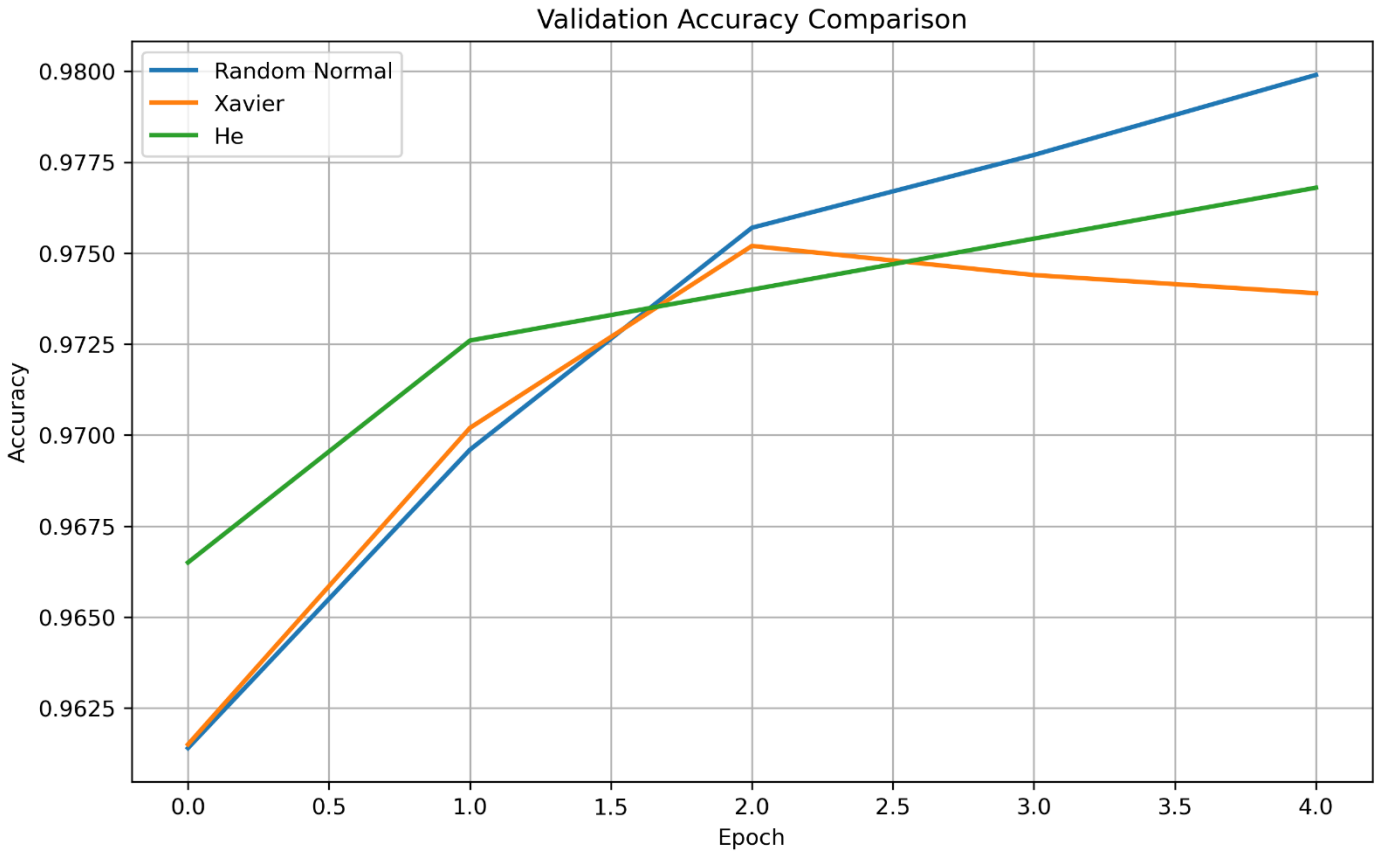
All other factors remain constant:

- Optimiser: Adam
- Loss: Sparse Categorical Crossentropy
- Batch size: 128
- Epochs: 5

This allows a controlled of convergence curves & performance

Experimental Results:

→Validation Accuracy Comparison:

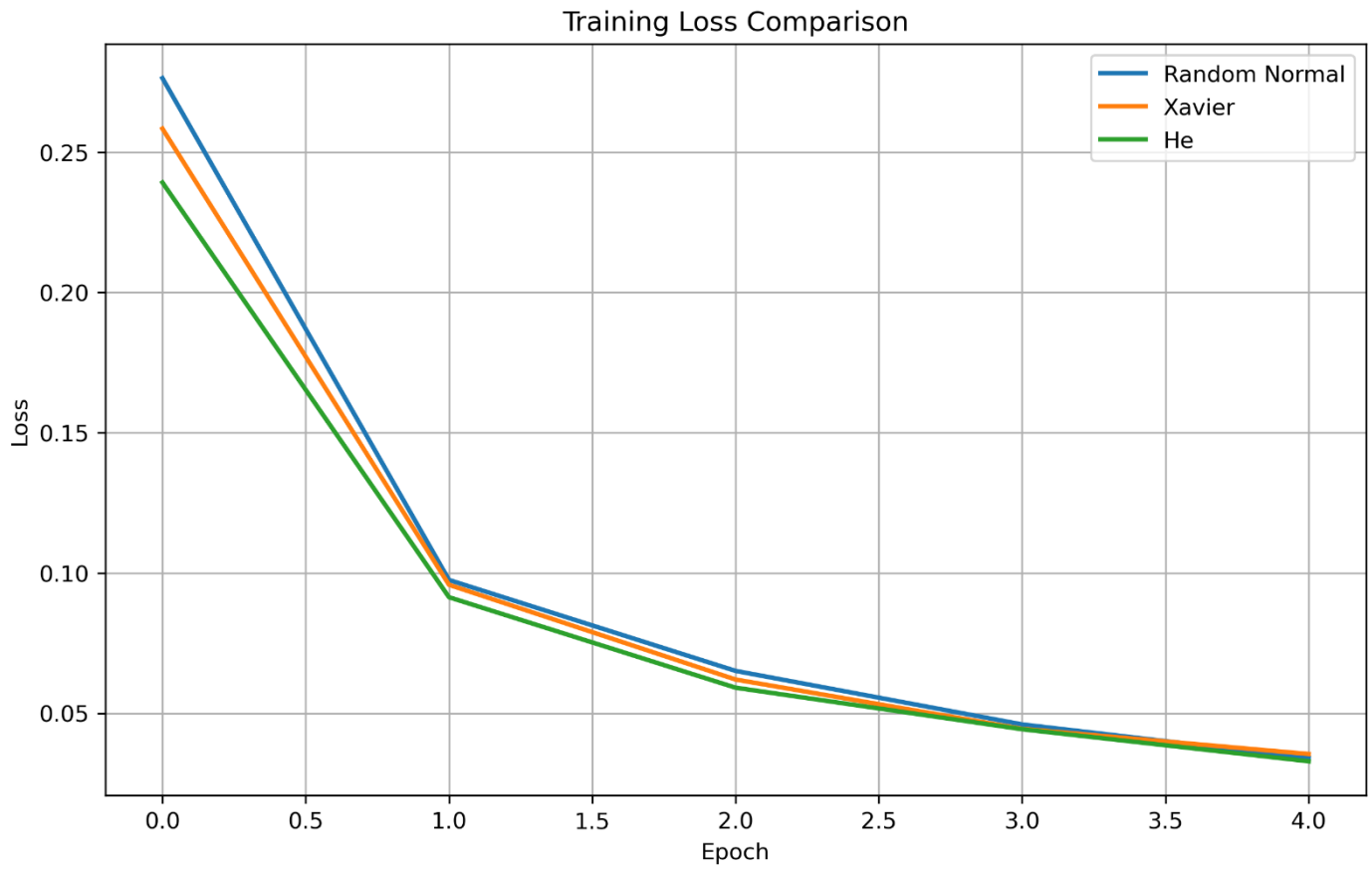


Observations:

- The Random Normal model has shown the least accuracy and varies widely in its performance.
- Xavier does a fair job but trends upward more smoothly.
- He Initialisation is better than both. It has the highest accuracy early and it is also stable.

This is what theory suggests: He is scaling matches activation behaviour of ReLU.

→Training Loss Comparison:

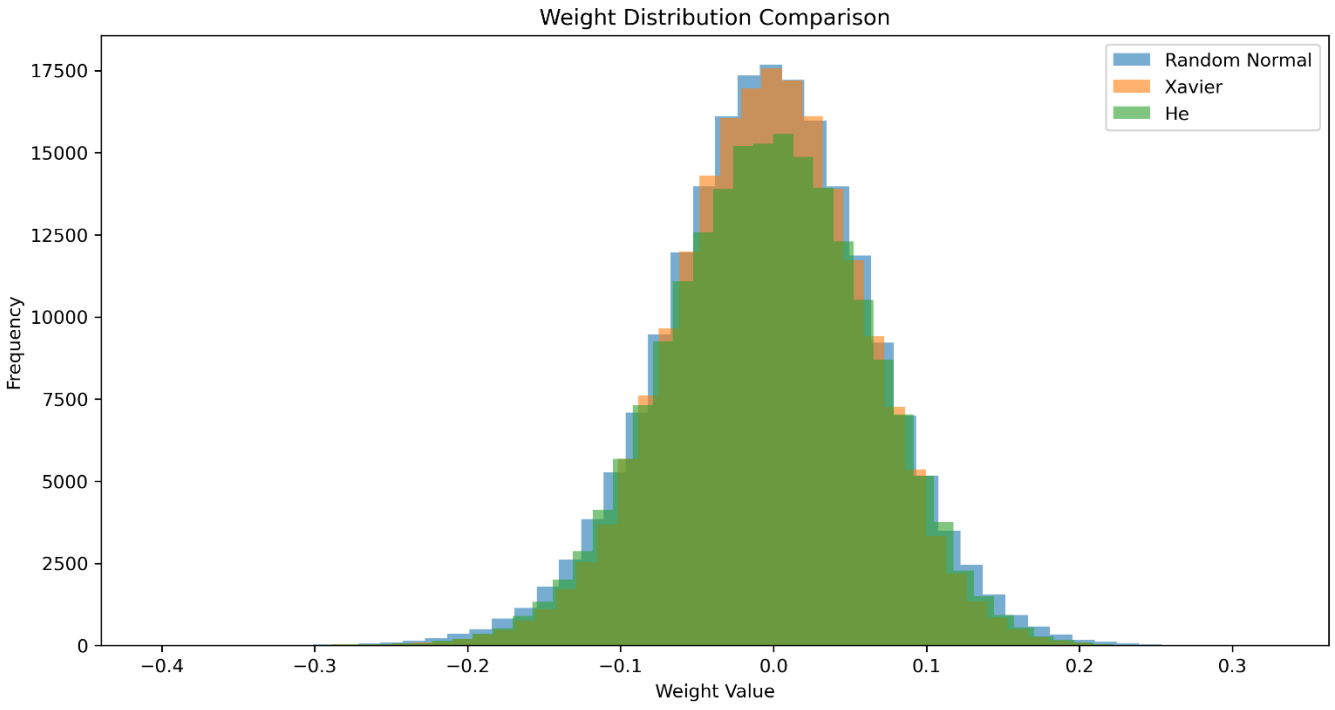


Analysis.

- The random normal has noisy behaviour and gradient flow instability.
- Xavier is more stable but plateaus earlier than He.
- He displays too smooth and sharp a decrease in loss, which describes good learning.

The mismatch in weight variance (like Random Normal) can obstruct and cause issues in learning.

→Initial Weight Distribution Comparison:



Interpretation.

- Weights produced by Random Normal are tightly clustered around zero.
- Xavier weighs layers in proportion to fan-in/fan-out.
- He spreads weights a little further to adjust for ReLU activation effects.

Different from these leads to differences in forward activation and gradients.

Discussion:

The results demonstrate a consistent relationship between initialisation strategy and training dynamics. Below, we analyse the findings in depth.

→Why Random Normal Initialisation Performs Poorly

Random Normal does not consider layer connectivity. As a result:

a. Activations shrink in deeper layers

The variance decreases exponentially, causing:

- near-zero activations,
- weak gradients,
- slow learning.

b. Exploding variance in some layers

If the initial variance is slightly high:

- activations blow up,
- gradients become unstable,
- regularisation becomes ineffective.

This explains the oscillating loss curve observed.

→Why Xavier Initialisation Improves Stability

Xavier balances the variance of activations:

$$Var(W) \approx \frac{1}{fan\_avg}$$

Benefits.

- gradient variance remains stable.
- deeper layers avoid saturation,
- convergence improves.

However, Xavier was originally designed for tanh, not ReLU. The variance only sees partial stabilization because of Xavier initialization, which is due to the activation distribution not being symmetric.

→Why Initialisation Performs Best

Initialisation is mathematically tailored for ReLU:

$$Var(W) = \frac{2}{fan\_in}$$

This compensates for the fact that ReLU discards negative values.

Resulting advantages:

1. Stronger gradients early in training
2. Stable forward activations
3. No variance shrinkage
4. Faster descent in loss
5. Consistent performance across layers

This leads to the smoothest loss curve and highest accuracy.

-→Gradient Flow Analysis.

The three-layer architecture amplifies the differences.

- The use of random normal gradients causes vanishing in deeper layers and slow learning.
- Xavier has a moderate gradient preservation and is reasonably stable.
- He → optimal gradient preservation → fast and stable.

His larger initial variance improves gradient propagation, especially in ReLU-using networks.

→Impact on Feature Representation.

MLPs don't generate spatial feature maps according to CNNs but the learned representations are still a function of accurate weight scaling.

Better initialisation produces.

- more discriminative neuron activations.
- better separation of digit classes.
- faster learning of high-level abstractions.

He initialisation causes more meaningful internal activation patterns.

→Implications for Modern Deep Learning.

Modern architectures overwhelmingly rely on.

- He initialization for ReLU-based networks.
- Xavier initialization for tanh/sigmoid networks.

ResNet model requires suitable initialisation (even with batch normalisation) for very deep architecture.

A strong algorithm won't be able to fix the behaviour if not correctly initialised.

Conclusion:

The weight initialization of the neural network plays an important role in determining its training behaviour. The experiment shows that Random Normal, Xavier and He initialisation function similarly in the same MLP architecture.

- Random Normal suffers from vanishing/exploding gradients,
- Xavier stabilises learning but is not optimised for ReLU,
- He initialization makes training most stable and efficient.

His method helps to improve learning speeds while making the networks behave more effectively. These results are consistent with current best practices, indicating why initialisation matters when designing deep learning models.

References

- Glorot, X., & Bengio, Y. (2010). Understanding the Difficulty of Training Deep Feedforward Neural Networks.
- He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning.
- Keras Documentation – Initialisers.
- LeCun, Y. — MNIST Dataset.