Project Title

Role of Weight Initialization: Xavier vs He vs Random Uniform

Rahul Chhabra - 25M0820
Viraj Ashar - 25M0756
Shreyash Thok - 25M0761
Prince - 25M0809
Kamal - 25M0814

CS725 FML - IIT Bombay

November 26, 2025

Problem Statement

- Goal: Train the same CNN on CIFAR-10 to see how weight initialization affects learning — comparing naive random uniform against activation-aware Xavier (for Tanh) and He (for ReLU) in terms of convergence speed, stability (activations/gradients), and final validation accuracy.
- Focus: How weight initialization affects our prediction and the final performance.
- Metrics: Accuracy, convergence speed (epochs to target), activation variance, etc

System and Setup

- Model: *CNN* with three 3×3 conv blocks ($32\rightarrow64\rightarrow128$ channels) plus a final linear classifier for CIFAR-10, run with ReLU/Tanh, using naive uniform init intially and Xavier/He init later
- Datasets: CIFAR-10 (50k/10k), standard train/val splits.
- Hardware: *T4 GPU Google Colab*.
- Framework: *Py*Torch

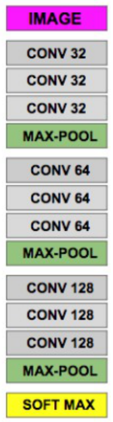
Initialization Formulas (Math)

- Xavier / Glorot: $w \sim \mathcal{U}\left(-\sqrt{\frac{6}{\text{fan}_{\text{in}}+\text{fan}_{\text{out}}}},\ \sqrt{\frac{6}{\text{fan}_{\text{in}}+\text{fan}_{\text{out}}}}\right)$ or $w \sim \mathcal{N}\left(0,\ \sqrt{\frac{2}{\text{fan}_{\text{in}}+\text{fan}_{\text{out}}}}\right)$.
- He / Kaiming (ReLU-family): $w \sim \mathcal{U}\left(-\sqrt{\frac{6}{\text{fan}_{\text{in}}}},\ \sqrt{\frac{6}{\text{fan}_{\text{in}}}}\right)$ or $w \sim \mathcal{N}\left(0,\ \sqrt{\frac{2}{\text{fan}_{\text{in}}}}\right)$.
- fan_{in} = inputs to a neuron; fan_{out} = outputs from a neuron. Choice of normal vs. uniform matches PyTorch defaults.

Model Diagram

CNN schematic: 3x3 conv blocks, activation, etc.

Reference: On weight initialization in deep neural networks



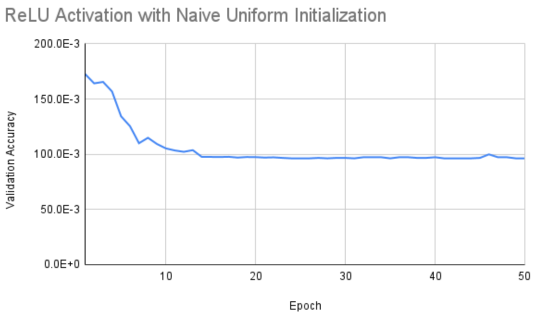
Challenges: Naive Uniform Init (CIFAR-10)

- ReLU, bound=5: activations/gradients blow up, loss in millions, val acc 18.16%; no learning.
- ReLU, bound=1: activations/gradients shrink up, loss in millions, val acc 17.26%; no learning.
- ReLU, bound=0.005: activations/gradients $\to$ 0, loss stuck at 2.302, val acc 10.06%; dead network.
- Tanh, bound=5: activations/gradients blow up, loss in millions, val acc 11.72%; no learning.
- Tanh, bound=1: unstable, high val loss, val acc 11.32%, gradients noisy.
- Tanh, bound=0.005: 68.81% val acc; activations drift/saturate, loss rises later.

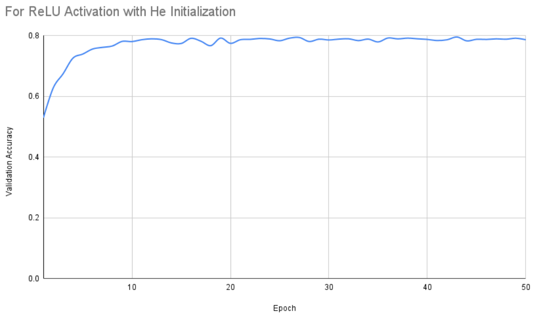
Improvements: Xavier (tanh) and He (ReLU)

- Accuracy: He+ReLU - 79.5%; Xavier+Tanh - 77.3%; Naive ReLU - 17.26%; Naive Tanh - 68.81%.
- Convergence: He/Xavier drop val loss fast; naive ReLU explodes or stays at 2.302; naive Tanh slows then plateaus.
- Gradients: He/Xavier steady; naive ReLU tiny or massive; naive Tanh is varying too much.
- Bottom line: use He for ReLU, Xavier for Tanh.

Metric 1: Validation Accuracy for ReLU

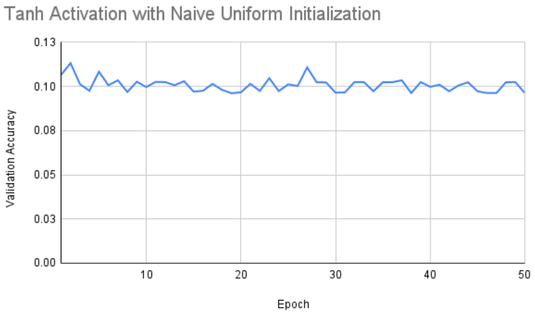


Naive Uniform

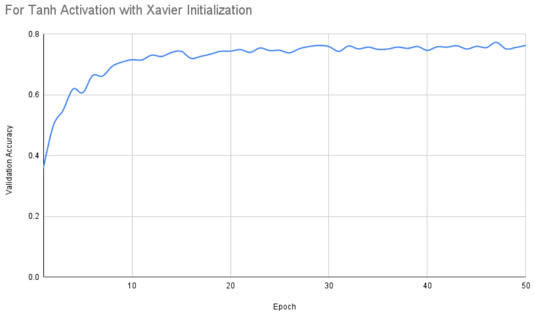


He

Metric 1: Validation Accuracy for tanh

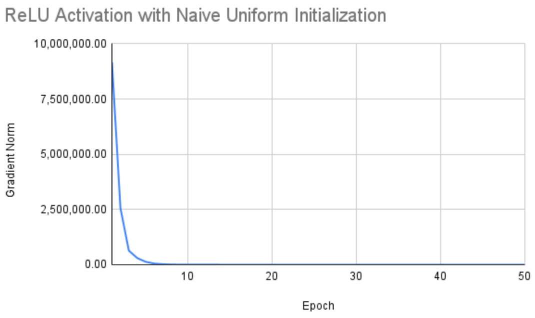


Naive Uniform

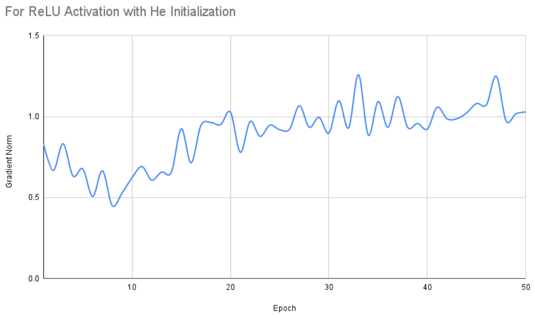


Xavier

Metric 2: Gradient Norm for ReLU

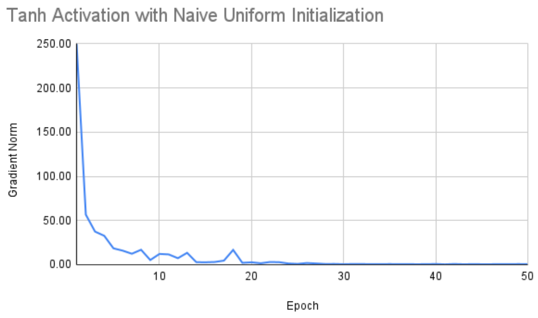


Naive Uniform

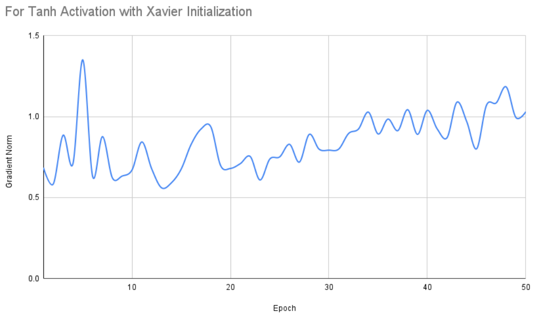


He

Metric 2: Gradient Norm for tanh



Naive Uniform



Xavier

Conclusion

- He + ReLU: 79.5% val accuracy; fast, stable. Best choice.
- Xavier + Tanh: 77.3% val accuracy; steady.
- Naive ReLU: large bound explodes; tiny bound sits at 17.26% (no learning).
- Naive Tanh: 68.81% ceiling; slower, noisier, val loss drifts.
- Overall: use He for ReLU, Xavier for Tanh. Naive uniform is either unstable or too weak.

Project Code

GitHub Repository: github.com/ashar-viraj/glorot-vs-he