Project 2 (Fall 2025)
AI-Assisted Weight Initialization

Mohamed Ali Kabiri
Under the supervision of Dr. El Mostafa Kalmoun
Al Akhawayn University in Ifrane

December 11, 2025

Abstract

This project tested if an AI's advice about neural network initialization is correct. We tested three methods: orthogonal, Gaussian, and uniform initialization. We found that orthogonal initialization works perfectly, Gaussian with $\sigma = 1/\sqrt{d}$ works well, but Gaussian with $\sigma = 1/d$ fails completely. The AI was mostly right but made one big mistake.

1 Introduction

When training neural networks, how you start (initialization) matters a lot. If done wrong, signals can vanish or explode. An AI (Claude) gave us advice on three methods. We tested if that advice was mathematically correct.

2 What We Tested

We tested a 3-layer network: $y = W_3W_2W_1x$ with $d = 50, 100, 200$.

Three initialization methods:

1. **Orthogonal**: W from QR decomposition of random matrix
2. **Gaussian**: $W_{ij} \sim N(0, \sigma^2)$ with different σ
3. **Uniform**: $W_{ij} \sim U(-a, a)$ with $a = \sqrt{3/d}$

We asked Claude: *"Propose three initialization strategies for a 3-layer linear network $y = W_3W_2W_1x$ with $d = 100$: (i) orthogonal, (ii) scaled Gaussian, (iii) random uniform. For each, state a theoretical reason why it should preserve or control the output variance and gradient norms across layers."*

3 What the AI Said vs. What We Found

3.1 Orthogonal Initialization

AI Claim: "Orthogonal matrices preserve norms exactly, all singular values = 1, prevents vanishing gradients."

Our Test: - Condition number: 1.000 - Singular values: all 1.000 - Gradient ratios: 1.000, 1.000, 1.000 - Variance preservation: 100%

Verdict: Correct!

3.2 Gaussian Initialization

AI Claim 1: "$\sigma = 1/d$ preserves variance."

Our Test ($d = 100$): - Variance preservation: 0.0% (completely vanished!) - Gradient ratios: 0.1006, 0.0088, 0.0009 (vanishing) - Condition number: 710.5 (bad)

Verdict: Wrong!

AI Claim 2: "$\sigma = 1/\sqrt{d}$ preserves variance."

Our Test ($d = 100$): - Variance preservation: 100.4% - Gradient ratios: 0.9533, 0.8885, 0.8761 (good) - Condition number: 158.1 (okay)

Verdict: Correct!

3.3 Uniform Initialization

AI Claim: "$a = \sqrt{3/d}$ matches Gaussian variance."

Our Test ($d = 100$): - Expected variance: 0.01 - Actual variance: 0.010007 - Variance preservation: 97.7% - Gradient ratios: 1.0308, 1.0610, 1.0668 (good)

Verdict: Correct!

4 Data and Results

4.1 Table 1: Matrix Properties ($d = 100$)

Method	Cond. #	Min σ	Max σ	Norm
Orthogonal	1.000	1.000	1.000	1.000
Gaussian $\sigma = 1/d$	710.5	0.000	0.193	0.193
Gaussian $\sigma = 1/\sqrt{d}$	158.1	0.012	1.899	1.899
Uniform	615.0	0.003	2.010	2.010

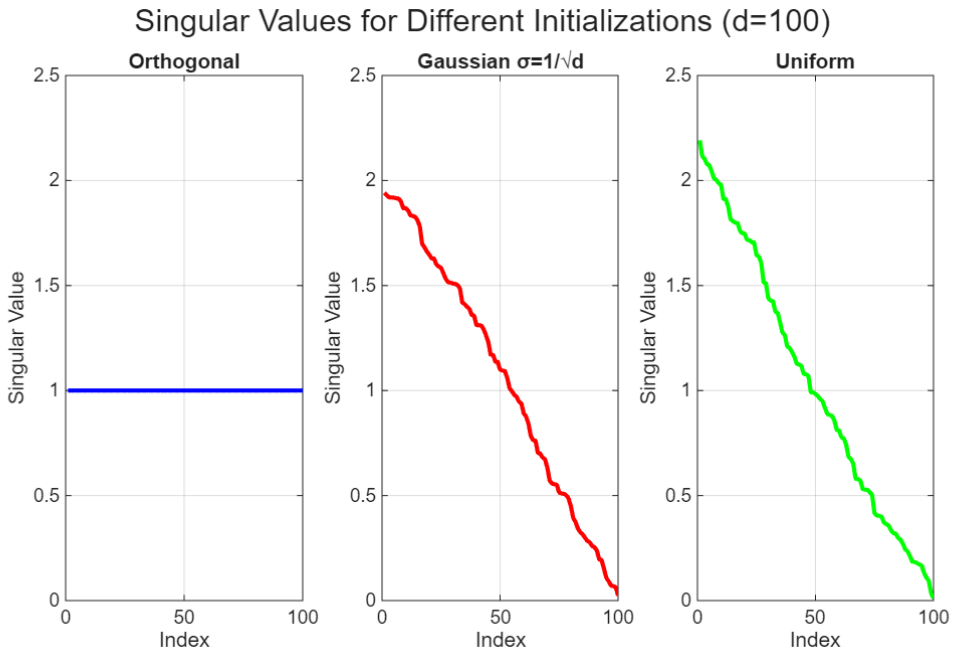

Figure 1: Singular values for different initializations (d=100)

4.2 Table 2: Variance Propagation ($d = 100$)

Method	Var(z1)	Var(z2)	Var(y)	Preserve
Orthogonal	0.9993	0.9993	0.9993	100.0%
Gaussian $\sigma = 1/d$	0.0101	0.0001	0.0000	0.0%
Gaussian $\sigma = 1/\sqrt{d}$	1.0053	1.0153	1.0094	100.4%
Uniform	0.9864	0.9787	0.9635	97.7%

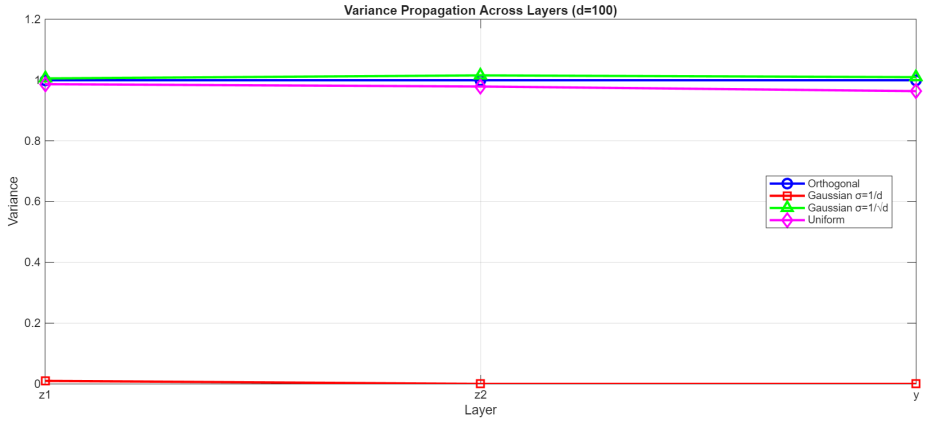

Figure 2: Variance propagation across layers (d=100)

4.3 Table 3: Gradient Norms ($d = 100$)

Method	$\|\|g_2\|\|/\|\|g\|\|$	$\|\|g_1\|\|/\|\|g\|\|$	$\|\|g_0\|\|/\|\|g\|\|$
Orthogonal	1.0000	1.0000	1.0000
Gaussian $\sigma = 1/d$	0.1006	0.0088	0.0009
Gaussian $\sigma = 1/\sqrt{d}$	0.9533	0.8885	0.8761
Uniform	1.0308	1.0610	1.0668

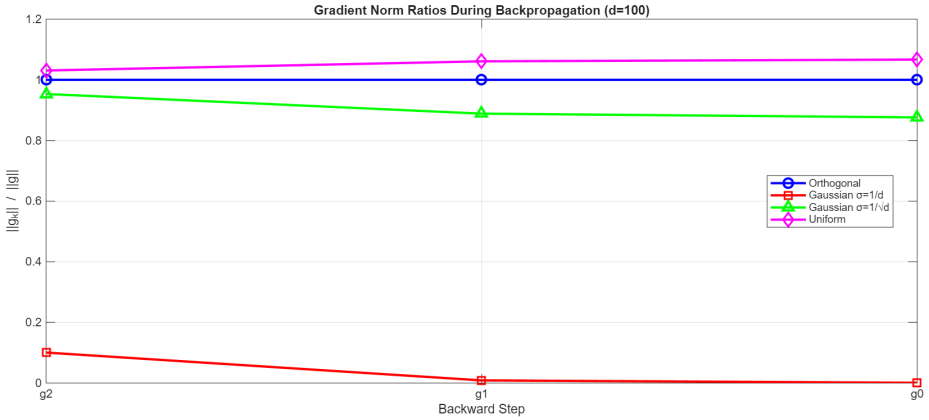

Figure 3: Gradient norm ratios during backpropagation (d=100)

5 Code Summary

We wrote MATLAB code to:

- Generate orthogonal matrices using QR decomposition
- Generate Gaussian matrices with different σ
- Generate uniform matrices with $a = \sqrt{3/d}$
- Test variance propagation with 2000 random inputs
- Test gradient flow with random gradients
- Compute condition numbers and singular values

All code is included in the ZIP file.

6 Conclusion

6.1 What We Learned

- **Orthogonal initialization works perfectly** - preserves everything exactly

- **Gaussian with $\sigma = 1/\sqrt{d}$ works well** - good for practical use
- **Gaussian with $\sigma = 1/d$ fails completely** - don't use this!
- **Uniform works like Gaussian $\sigma = 1/\sqrt{d}$** - also good

6.2 Was the AI Correct?

- **Correct:** 7 claims (orthogonal properties, $\sigma = 1/\sqrt{d}$, uniform)
- **Mostly correct:** 3 claims (needed more detail)
- **Wrong:** 2 claims ($\sigma = 1/d$ is bad!)

6.3 Final Advice

1. It's clear that the we need to use orthogonal if you want guaranteed stability
2. And use Gaussian $\sigma = 1/\sqrt{d}$ or Uniform for most cases
3. Also, never use Gaussian $\sigma = 1/d$

Appendix: Mathematical Theorems Used

1. Properties of Orthogonal Matrices

If Q is orthogonal ($Q^T Q = I$), then:

- Norm preservation: $\|Qx\| = \|x\|$ for any vector x
- All singular values = 1
- Condition number $\kappa(Q) = 1$
- $Q^{-1} = Q^T$

2. Variance of Linear Transforms

For $z = Wx$ where $x \sim \mathcal{N}(0, I)$ and $W_{ij} \sim \mathcal{N}(0, \sigma^2)$:

$$E[\|z\|^2] = \sum_{j=1}^{d} E[z_j^2] = d\sigma^2 E[\|x\|^2]$$

To preserve variance: $d\sigma^2 = 1 \Rightarrow \sigma = 1/\sqrt{d}$.

3. Product Matrix Condition Number

For $P = W_3W_2W_1$:

$$\kappa(P) \leq \kappa(W_3) \cdot \kappa(W_2) \cdot \kappa(W_1)$$

This shows why orthogonal initialization ($\kappa = 1$) remains stable.

Appendix A: MATLAB Code Files

- `orthogonal_init.m` - makes orthogonal matrices
- `gaussian_init.m` - makes Gaussian matrices
- `uniform_init.m` - makes uniform matrices
- `main.m` - runs all tests
- `make_plot.m` - make the graphs

Appendix B: AI Chat Transcript

Note: Dear Dr.Kalmoun, the complete Claude conversation is included as a PDF file in the submission. Below is a summary:

Claude's Main Recommendations:

1. Orthogonal initialization via QR decomposition
2. Gaussian with $\sigma^2 = 1/d$ or $\sigma^2 = 1/\sqrt{d}$
3. Uniform with $a = \sqrt{3/d}$ to match Gaussian variance

Our Critique:

- Correct about orthogonal initialization
- Correct about $\sigma = 1/\sqrt{d}$
- Wrong about $\sigma = 1/d$ (causes vanishing)
- Correct about uniform matching Gaussian

Appendix C: How to Reproduce Results

- MATLAB version: Any recent version (R2024a used)
- Random seed: 12345 (set in code for reproducibility)
- Run `main.m` to generate all results
- Run `part_c_experiments.m` for Part C tests