The impact of Initialization and Learning Rate Scheduling in Deep Learning Optimization

Pang Liu
jeff.pang.liu@gmail.com

October 27, 2025

Project Report Content and Overview

The objective of this project is to analyze the impact of two critical hyperparameters on deep neural network training: **Weight Initialization** and **Learning Rate Schedules**.

We will use two different models for this analysis:

1. A deep CNN architecture, **VGG-19**, to evaluate initialization methods.
2. A residual network, **ResNet-34**, to evaluate learning rate schedulers.

The baseline hyperparameter settings for these experiments are as follows:

- **VGG-19 (Initialization Exp):** Optimizer: SGD, Epochs: 50.
- **ResNet-34 (LR Schedule Exp):** Optimizer: Adam, Initialization: Kaiming Uniform, Epochs: 100.

I will report this research in the following order:

- **1 - Weight Initialization Comparison (VGG-19)**

  In this section, I will first compare **Kaiming vs. Xavier** initialization (using normal distribution) to demonstrate the importance of activation-aware initialization methods. Then, I will compare **Kaiming Uniform vs. Kaiming Normal** to analyze the impact of the chosen distribution.

- **2 - Learning Rate Scheduling Comparison (ResNet-34)**

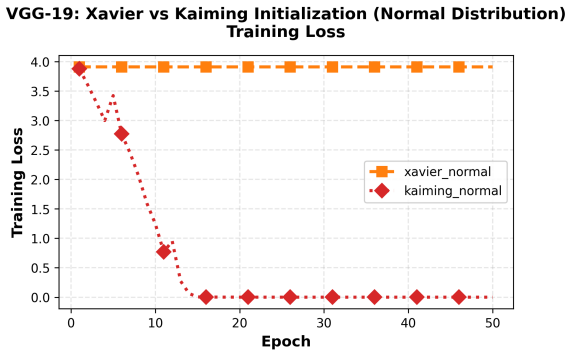
  In this section, I will compare five different LR schedulers: **Constant**, **Exponential**, **MultiStep**, **Cyclic**, and **Cosine Annealing**. The analysis will be conducted in a "tournament" style to find the best-performing scheduler.

- **3 - Conclusion**

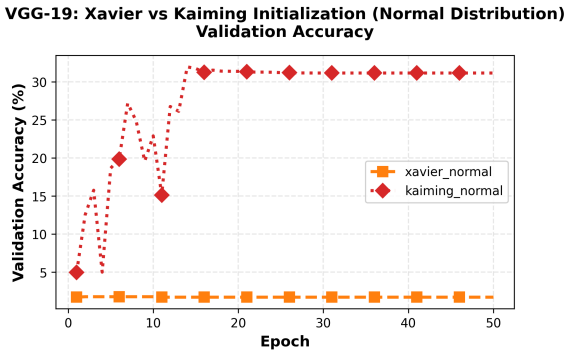
  A final summary of the experimental results, highlighting the best-performing methods identified in the experiments.

1 Weight Initialization Comparison (VGG-19)

1.1 Comparison of Initialization Methods (Xavier vs. Kaiming)



(a) Training Loss vs. Epochs (Xavier Normal vs. Kaiming Normal)



(b) Validation Accuracy vs. Epochs (Xavier Normal vs. Kaiming Normal)

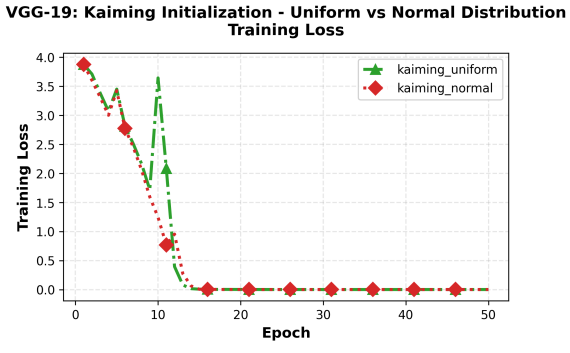
From the plots above, we can observe a **dramatic performance difference** between the two initialization methods.

The **Kaiming Normal initialization** (red diamond line) enables the model to start learning immediately. Its training loss (Figure a) drops rapidly towards zero, and its validation accuracy (Figure b) successfully climbs to over 30%.

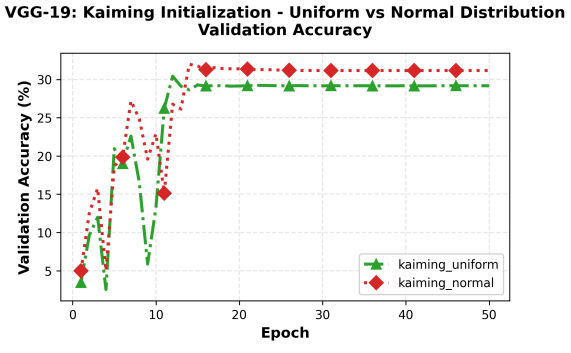
In contrast, the **Xavier Normal initialization** (orange square line) **completely fails to train the network**. As seen in both plots, its training loss is a straight line, and its validation accuracy is also a straight line that has no change.

This result is expected and aligns with what we have learned. Kaiming initialization is specifically designed for networks using ReLU activation functions, which VGG-19 primarily uses. It prevents vanishing or exploding gradients by scaling the weights correctly. Xavier initialization, designed for sigmoid or tanh activations, is unsuitable for ReLU. This mismatch causes the failure, completely halting the learning process, which is exactly what our experiment shows.

1.2 Comparison of Distribution (Uniform vs. Normal)



(a) Training Loss vs. Epochs (Kaiming Uniform vs. Kaiming Normal)



(b) Validation Accuracy vs. Epochs (Kaiming Uniform vs. Kaiming Normal)

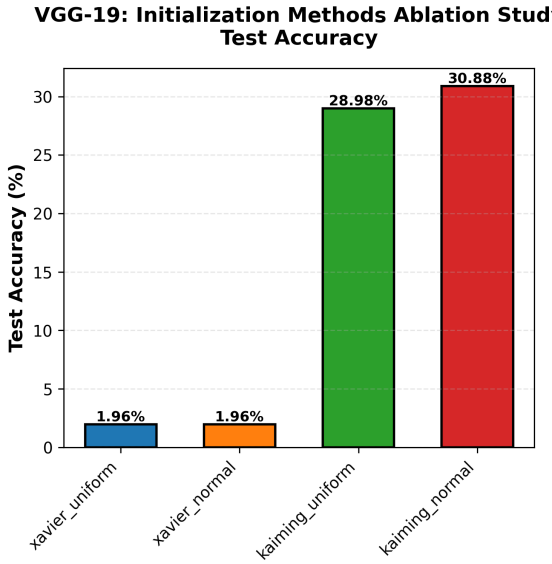
Based on the results from Section 1.1, we furthermore selected the Kaiming initialization method to compare its Uniform and Normal distribution variants.

From the training plots, we can see that while **both methods ultimately enable the network to train, their training dynamics are quite different.** The **Kaiming Normal** (red diamond line) demonstrates a relatively smooth and stable convergence path.

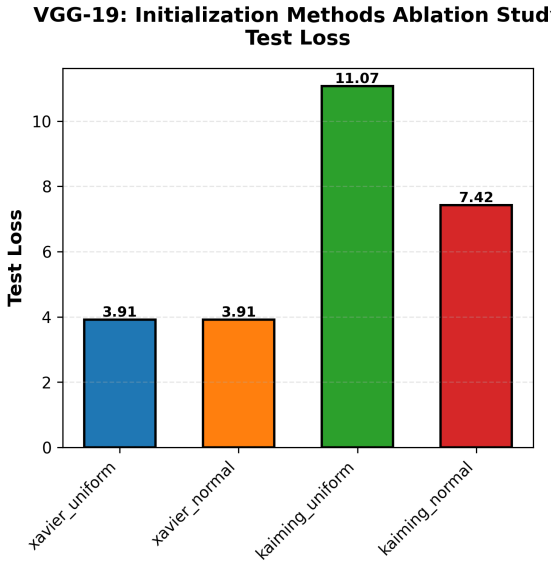
In contrast, the **Kaiming Uniform** (green triangle line) exhibits **significant instability** during the initial training phase (around epochs 5-10). We can clearly observe a **large punctuation in its training loss** (Figure a) and a corresponding **sharp drop and recovery in its validation accuracy** (Figure b).

Although both methods eventually converge, this initial instability of Kaiming Uniform suggests it may struggle to find an optimal solution as effectively as Kaiming Normal, and its unstability might cause more unknown risks, therefore in this experiment Kaiming Normal performs better and is suggested to use firstly.

1.3 Ablation Study: Final Test Performance



(a) Final Test Accuracy (Initialization Methods on VGG-19)



(b) Final Test Loss (Initialization Methods on VGG-19)

Table 1: VGG-19 Initialization Method Performance Summary

Method	Test Accuracy (%)	Test Loss
Kaiming Normal	30.88	7.4210
Kaiming Uniform	28.98	11.0686
Xavier Uniform	1.96	3.9130
Xavier Normal	1.96	3.9130

The bar charts and Table 1 summarize the final test performance of all four methods.

The results confirm the conclusions we drew from our training plots. Kaiming Normal is the definitive winner, achieving the highest test accuracy of 30.88%. Kaiming Uniform follows at 28.98%.

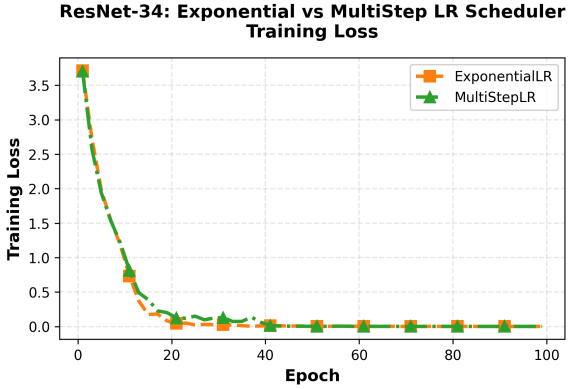
Furthermore, it is critical to observe the Test Loss (Figure b). While Kaiming Normal's accuracy is only 2% higher than Kaiming Uniform's, its final test loss is significantly lower (7.42 vs 11.07). This suggests that the model trained with Kaiming Normal is not only more

accurate but also more confident in its predictions, indicating superior generalization. This strong result correlates with the more stable training dynamic we observed in Section 1.2.

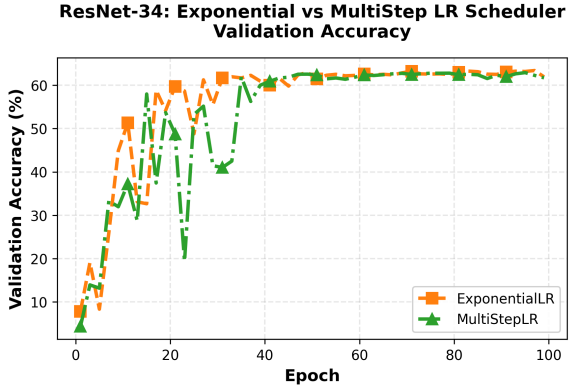
As observed during training, both Xavier methods are complete failures, achieving only 1.96% accuracy. This is equivalent to random guessing on this dataset. This experiment clearly demonstrates that proper initialization is not just a minor optimization but a critical requirement for training deep networks like VGG-19. A wrong choice (e.g., using Xavier for ReLU) will result in the model failing to learn at all.

2 Learning Rate Scheduling Comparison (ResNet-34)

2.1 Fixed Schedule Comparison (Exponential vs. Multistep)



(a) Training Loss vs. Epochs (Exponential vs. Multistep)



(b) Validation Accuracy vs. Epochs (Exponential vs. Multistep)

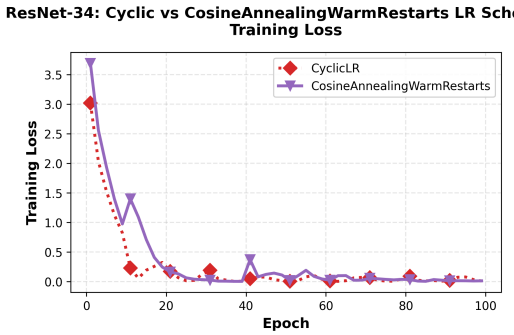
This comparison evaluates two "fixed-decay" schedulers: **ExponentialLR** (orange square line) and **MultiStepLR** (green triangle line).

From the Training Loss plot (Figure a), the two methods perform almost identically, with their curves highly overlapping.

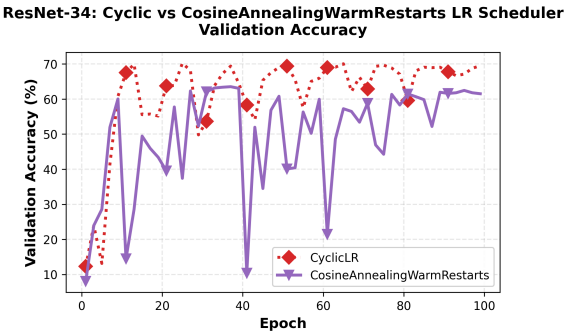
The Validation Accuracy plot (Figure b) reveals their different training dynamics. The ExponentialLR curve is relatively smooth. In contrast, MultiStepLR exhibits **significantly larger and more frequent fluctuations**.

Although ExponentialLR appears more stable during validation, the final test accuracy from our ablation study (see Table 2) reveals that **MultiStepLR is the winner of this comparison**, achieving **61.62%** accuracy, slightly outperforming ExponentialLR's **60.98%**. This indicates the validation-time volatility did not negatively impact its final generalization.

2.2 Cyclic Schedule Comparison (Cyclic vs. Cosine Annealing)



(a) Training Loss vs. Epochs (Cyclic vs. Cosine Annealing)



(b) Validation Accuracy vs. Epochs (Cyclic vs. Cosine Annealing)

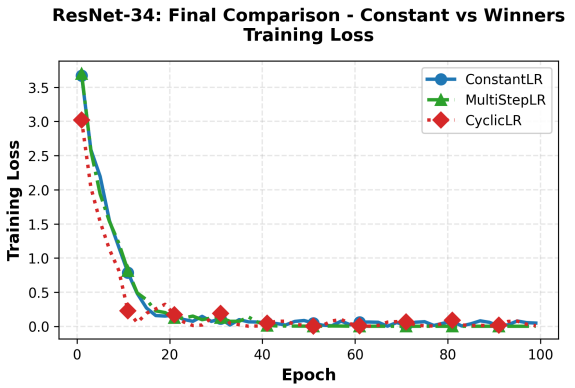
This section compares two schedulers that use cycles and restarts: **CyclicLR** (red diamond line) and **CosineAnnealingWarmRestarts** (purple inverted triangle line).

The validation accuracy plot shows a **striking performance difference**. The CyclicLR curve climbs steadily and maintains a high accuracy, demonstrating a stable path to a good minimum.

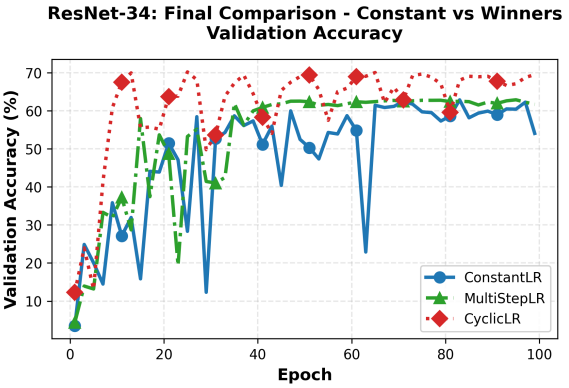
In contrast, CosineAnnealingWarmRestarts shows **extremely punctuate**. This is a direct consequence of its design, the Warm Restarts abruptly reset the learning rate to its maximum value. This large LR spike "kicks" the optimizer out of its current position, causing the validation accuracy to plummet. While the model recovers as the LR anneals, these repeated, violent shocks prevent it from settling into a high-quality minimum.

The final test results (Table 2) confirm this. **CyclicLR is the clear winner of this comparison**, achieving **69.45%** accuracy, which is decisively better than the volatile CosineAnnealingWarmRestarts (62.09%).

2.3 Final Comparison (Constant vs. Winner 1 vs. Winner 2)



(a) Training Loss vs. Epochs (Constant LR vs. Winner 1 vs. Winner 2)



(b) Validation Accuracy vs. Epochs (Constant LR vs. Winner 1 vs. Winner 2)

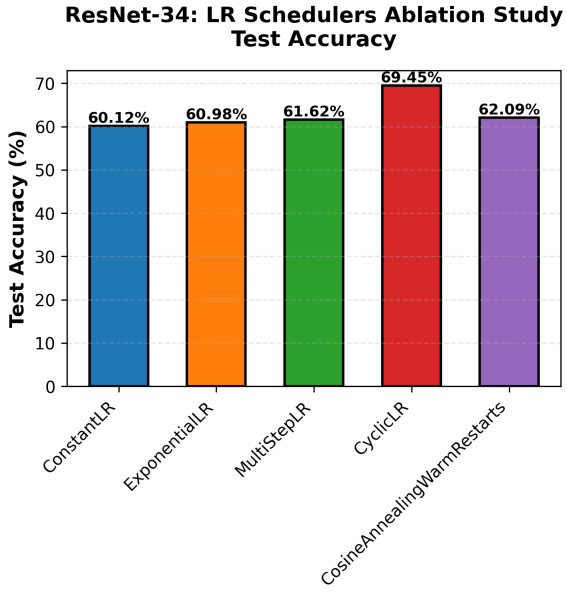
This final comparison pits the baseline (**ConstantLR**, blue circle line), the best fixed scheduler from 2.1 (**MultiStepLR**, green triagle line), and the best cyclic scheduler from 2.2 (**CyclicLR**, red diamond line) against each other.

(Note: MultiStepLR is chosen as 'Winner 1' because its final **test accuracy** of 61.62% was higher than ExponentialLR's 60.98%, as shown in the final ablation study (Section 3.4), despite its volatile validation curve.)

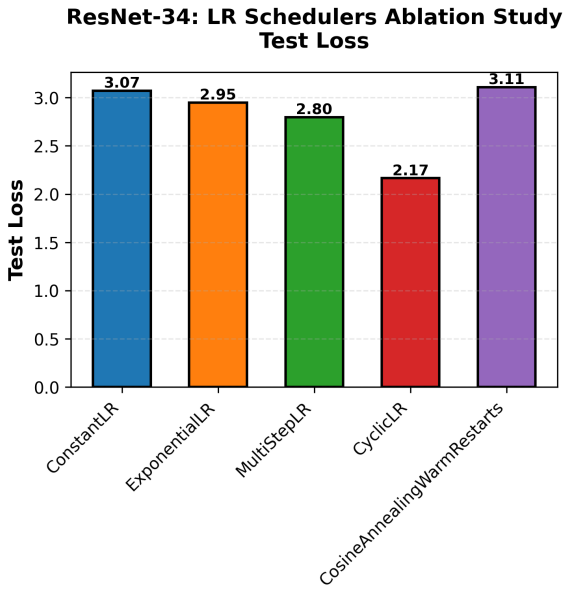
Although the training loss (Figure a) is not significantly different between the schedulers in the final epochs, the validation accuracy (Figure b) tells a clearer story. Both schedulers, MultiStepLR and CyclicLR, clearly outperform the baseline ConstantLR. In fact, the ConstantLR curve is not only the lowest in terms of accuracy but also shows obvious peaks and valleys, indicating an unstable training path.

Comparing the two winners, CyclicLR shows a much better overall performance. Its accuracy curve climbs far higher than MultiStepLR, which appears to hit a performance plateau after epoch 60. This demonstrates CyclicLR's superior ability to find a better, more generalizable minimum.

2.4 Ablation Study: Final Test Performance



(a) Final Test Accuracy (Learning Rate Schedulers on ResNet-34)



(b) Final Test Loss (Learning Rate Schedulers on ResNet-34)

Table 2: ResNet-34 LR Scheduler Performance Summary

Scheduler	Test Accuracy (%)	Test Loss	vs Baseline
CyclicLR	69.45	2.1654	+9.33%
CosineAnnealingWarmRestarts	62.09	3.1062	+1.97%
MultiStepLR	61.62	2.7977	+1.50%
ExponentialLR	60.98	2.9458	+0.86%
ConstantLR (Baseline)	60.12	3.0683	+0.00%

The final table shows the performance of five different learning rate (LR) schedulers on a ResNet-34 model. The ConstantLR scheduler was used as the baseline, or starting point, and it achieved a test accuracy of 60.12%. The results clearly show that the CyclicLR scheduler performed the best by a large margin. It reached a much higher test accuracy of 69.45% and had the lowest test loss. This was a significant 9.33% improvement over the baseline. The other schedulers, like CosineAnnealingWarmRestarts and MultiStepLR, also performed better than the baseline, but their improvements were much smaller (only about 1-2%). In summary, this test indicates that CyclicLR was by far the most effective scheduling method.

3 Conclusion

From the experiments in this project, we can draw two main conclusions:

(1) On Initialization (VGG-19): The choice of weight initialization is critical and non-negotiable. For a deep network using ReLU activations like VGG-19, using an inappropriate method (like Xavier, designed for tanh/sigmoid) will lead to **complete training failure** (1.96% accuracy). **Kaiming initialization is essential for training to succeed**. Among its variants, **Kaiming Normal (30.88% accuracy)** performed slightly better than Kaiming Uniform (28.98%) on this task.

(2) On Learning Rate Scheduling (ResNet-34): The LR schedule is a powerful tool for improving optimization. While standard decay methods (Exponential, MultiStep) offered only marginal gains over a constant LR, the **CyclicLR scheduler dramatically outperformed all other methods** (69.45% accuracy). This represents a **massive +9.33% improvement** over the baseline (60.12%). This result strongly suggests that dynamically oscillating the learning rate is highly effective for training deep residual networks, likely by helping the optimizer traverse the complex loss landscape and escape suboptimal local minima.