Neural Network Initialization Using Color-Aware Perlin Noise

Mustafa Atak
Department of Computer Engineering

Bogazici University
mustafa.atak@std.bogazici.edu.tr
Student ID: 2020400042

January 2025

1 Introduction

Every person has heard the word machine learning at least once in the last decade. Deep Neural Networks in this field are of great importance to us and how successful they are is normally a very important detail for their application in the field they are trained for. What ensures this success is how well the model is trained. Here, of course, our biggest problem is to find a dataset for the model. Producing this good dataset is a big challenge for us. Because collecting and annotating large datasets is often impractical, expensive, or sometimes impossible. Traditional ways of initializing a network, such as He and Xavier, are math-based and this can be seen as a solution for that. However, the way they set the weights is random and does not have the patterns found in real images. That creates a big problem when there is not much data because networks find it hard to learn important features without enough examples. Furthermore, although pretraining large datasets is a popular idea, its use may generally be limited to teaching purposes (due to licensing issues) or the computational cost may be too high. These issues create the need for innovative initialization approaches that can enable effective training of neural networks with limited data while maintaining high performance.

The first choice that comes to mind when it comes to initializing neural networks is statistical approaches such as He initialization, which uses Gaussian distributions, and Xavier initialization, which uses uniform distributions to determine the initial weights. Although these approaches provide a stable foundation mathematically, they fail to understand the natural visual structures that we want to achieve as the final result. The concept of pretraining emerged to work around this situation. Still, the substantial computational resources and access to extensive data collection required to accomplish this again started to create problems for us.

This leads us to look for solutions that offer a data-logical perspective for initialization that will allow us to reduce the amount of data to be used and the time to train. In this respect, Perlin noise is an interesting alternative. Unlike purely random noise, Perlin noise

is coherent and creates patterns that share the same properties as natural textures and can therefore be useful for initializing the weights of neural networks. Such patterns offer a more regular initialization for network filters and therefore seem useful at the beginning, when very basic visual features are usually learned in the first convolution layers of a neural network.

Based on the work of Inoue et al. [1], this paper explores a new initialization method of neural networks using structured patterns of Perlin noise. The method proposed in their paper has two parts: one is a generator of noise patterns, by which Perlin noise in a controlled manner is generated; the other is the noise label generator, in which the generated noise patterns are classified according to complexity. This two-stage process constitutes a key innovation in the literature, as it accomplishes neural network initialization with no dependency on real images and simultaneously yields state-of-the-art performance. Their proposed method is especially effective in the regime of low-training data, where almost all existing initialization methods degrade in performance. They have, through extensive experiments on benchmark datasets such as CIFAR-10, CIFAR-100, Omniglot, and DTD, shown that initialization by Perlin noise leads to consistently better performance over standard initialization methods, specifically for tasks involving the recognition of textures and characters with natural structural patterns.

The remainder of this paper is organized as follows. First, we examine the achievements of the original article and compare them with traditional initialization and training techniques. Next, we will consider what we want to gain from it and why we want to do it. This will be followed by explaining and analyzing the results and their application. The last part will present our conclusions based on the results and what these results point to.

2 Existing System Model

The initialization of network parameters is a very critical factor in the training of deep neural networks that greatly influences the convergence and performances of the model. Generally, a number of traditional initialization methods have been developed in order to set the initial weight of neural networks hoping that the learning will be efficient and avoid problems associated with vanishing or exploding gradient.

2.1 Traditional Initialization Methods

2.1.1 Xavier Initialization

The aim of this scheme is to provide the same initial variance and the same propagated gradient between the layers of the network. If we say n_in for connections entering a layer and n_out for connections leaving a layer, the assignment of weights is calculated as follows:

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

This method prevents signals from diminishing and disappearing or amplifying and damaging the training of the model and works well with sigmoid or hyperbolic tangent activation functions.

2.1.2 He Initialization

He initialization was built on Xavier's ideas but he adjusted the variance for ReLU activations. Taking into account that these ReLU units output half of them as 0 because they output 0 for negative values, the He initialization variance is changed as follows:

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

By doing so, it ensures that the forward and backward propagated signals have a variance that is neither too high nor too low. This creates a system especially suitable for the ReLU-activated layers we mentioned.

2.1.3 Limitations of Traditional Methods

Although the Xavier and He initialization methods are mathematically sound and have been used in many applications, they themselves depend on random distributions that do not incorporate any domain-specific information or structural patterns found in natural images. This becomes especially problematic when we have limited data, because the starting weights may not provide us with a starting point that gives us enough information to learn meaningful features.

On top of that, these methods assume that the layers in the model are densely connected and that there is no correlation between activations, which can make it difficult to detect the complexity of Convolutional Neural Networks because spatial correlations are vital in CNNs.

2.2 Pre-training on Large Datasets

Pretraining the model is another alternative to random initialization. The model is first trained on large-scale datasets and is made to capture certain object recognition features. The original dataset is then used to bring it to a state with the desired features. This approach shows improved performance, especially for model training with small datasets, as the model learns general features beforehand and is then transferred to specific tasks.

2.2.1 Advantages of Pre-training

Feature Transferability: The initial layers of the model learn generic properties, which in general are useful across different tasks.

Improved Convergence: Starting with a pre-trained model can provide much better optimization and faster convergences.

2.2.2 Challenges with Pre-training

Data Availability and Licensing: Many of the datasets mentioned are not suitable for commercial use due to license restrictions.

Computational Resources: Training on these large datasets requires a lot of computational power and time.

Domain Mismatch: The learned features may not be optimal for different domains, which will result in loss of benefit.

2.3 Need for Alternative Initialization Approaches

Given the shortcomings of traditional initialization methods and the practical issues of pre-training, there has been an increasing demand for alternative approaches that can provide informative initial weights without a large-scale dataset. Ideally, those methods should:

Incorporate Structural Patterns: Provide initial weights representative of natural patterns in images to help to learn right from the beginning.

Be Data-Efficient: Allow for efficient training with limited data by providing a better initialization for optimization.

Avoid Dependence on External Datasets: Avoid data availability and licensing issues by not requiring any real images for initialization.

2.4 Perlin Noise as Initialization Tool

Invented by Ken Perlin and used to produce natural-looking textures in computer graphics, gradient noises called Perlin noise seem to be an alternative to take network initializations one step further with the following features:

- Structured Patterns: Unlike Gaussian or uniform noise, Perlin noise creates structured patterns similar to the textures of natural images.
- Controlled Complexity: Perlin noise can generate new patterns over a wide range of variations by changing only a few parameters.
- Area Relevance: Initialization with such patterns can enable the network to start with filters for edges and textures, which are fundamental in computer vision tasks.

These features give advantages:

- No Dependency on Real Images: Since Initialization does not require real-life data, we can grow our dataset as we wish.
- Incorporation of Natural Patterns: Perlin noise provides a much more successful starting point than other methods by enabling the learning of much more structural patterns for learning from image data. This means that the initialized weights are more informative than random weights.

2.5 Performance on Benchmark Datasets and Improvements

This method has been tested on several image categorization datasets and has shown a much better result compared to the previously mentioned methods.

- CIFAR-10 and CIFAR-100: It outperformed the benchmarks with 94.27% accuracy with ResNet152 for CIFAR-10 and 78.21% for CIFAR-100, which are very hard to beat since they are well-studied data.
- Omniglot: Showed dramatic effectiveness in tasks requiring character recognition achieving 18.71% accuracy using ResNet152.
- DTD: Achieved strong performance with 54.18% accuracy using ResNet152, showing its strength in texture recognition tasks.

Although Perlin Noise has shown very promising results, its implementation remains grayscale Perlin Noise. On top of that, they mention that incorporating color information could potentially enhance the initialization, especially for datasets with color images.

3 Proposed Color Space Extension for Perlin Noise Initialization

3.1 Motivation

Inoue et al.'s work uses grayscale Perlin noise patterns for network initialization. Though this has generated wonderful outputs, especially within the arenas of texture recognition and character recognition, the process has one main drawback: the inability to include color information during the initialization process. This becomes an issue especially when training networks on natural image databases where color is an important factor in feature extraction and image classification.

Current convolutional neural networks work on RGB images which are fed to the network through three different channels that capture different intensities of light. The current grayscale initialization technique might not be sufficient to tune the network's weights at the initial stage to accommodate color features which could in turn affect the convergence rate or the performance of the network especially when working with high-resolution color images.

3.2 Proposed Extension

We extend the Perlin noise initialization idea by embedding color information into the RGB Perlin noise generator. This way, we preserve the original paper's properties and make it color-aware for colored images. We are implementing this feature in the following way:

- Color Complexity Levels
  - Level-based system for controlling channel relationships
  - Four distinct complexity levels from grayscale to full independence
  - Base noise pattern serves as a reference for controlled variation

3.2.1 Technical Approach

The key components of the proposed color extension are:

1. Color Complexity Levels
   - Level-based system for controlling channel relationships
   - Four distinct complexity levels from grayscale to full independence
   - Base noise pattern serves as a reference for controlled variation
2. Level-Based Color Generation
   - Level 1: Single-channel generation with replication ($R = G = B$)

- Level 2: Weighted combination with mild independence from base
- Level 3: Weighted combination with moderate independence from base
- Level 4: Independent channel generation

3. Extended Label Generation

$$y_i = (n-1)M \cdot 4 + (m-1) \cdot 4 + (c-1)$$

where n, m corresponds to grid parameters, M is the grid size multiplier which is used to create distinct ranges for each grid configuration, and $c \in \{1, 2, 3, 4\}$ represents the color complexity level mentioned above.

3.3 Hypothesis

We assume that color-sensitive initialization with our approach will:

1. Improve the Performance in Classification:
   - Higher accuracy in color-dependent datasets
   - Faster convergence on training
   - More robust feature learning
2. Improve Network Efficiency:
   - Natural initialization of color-sensitive filters
   - Multi-channel architecture leveraging effectively
   - Less time training on color-based tasks
3. Generalize Better:
   - Improved transfer learning
   - More robustness across different domains
   - Better handling of color variations

3.4 Implementation Details

The implementation works on the idea of making the original Perlin noise algorithm adapt to colors as well. A level-based approach is followed to achieve channel correlation control. Key steps are:

1. Base Noise Generation: The algorithm first generates Perlin noise patterns sized $(2^n \times 2^m)$ using the grid generation method mentioned in the original paper. This is used as the base noise for color complexity in subsequent steps.
2. Color Complexity Levels: The implementation introduces four distinct levels of color complexity:
   - Level 1 (Grayscale): All RGB channels are perfectly correlated, producing grayscale-like noise patterns where $R = G = B$.

- Level 2 (Slight Variation): Channels maintain a high correlation (70%) with the base noise while introducing 30% independent variation.
- Level 3 (Moderate Independence): Reduces channel correlation to 40% with the base noise, allowing 60% independent variation.
- Level 4 (Full Independence): Each RGB channel is generated with separate Perlin noise pattern functions to make it independent.

3. Channel Generation: For each color level mentioned in Step 2, red in RGB colors takes the base noise value from Step 1. Green and blue are recalculated with this value and the weighted averages of independent Perlin patterns. The weights of these averages are adjusted according to the level and provide us with a controllable variation.

4. Normalization: The last step normalizes the generated noise to the range $[0, 1]$ so that the data distribution density remains consistent for each level.

This implementation maintains the computational efficiencies of the original paper at the same time allowing us to systematically incorporate color relationships in the initialization process of the model. Depending on the nature of the target dataset, a color complexity level can be changed to allow domain-specific optimization of the initialization process.

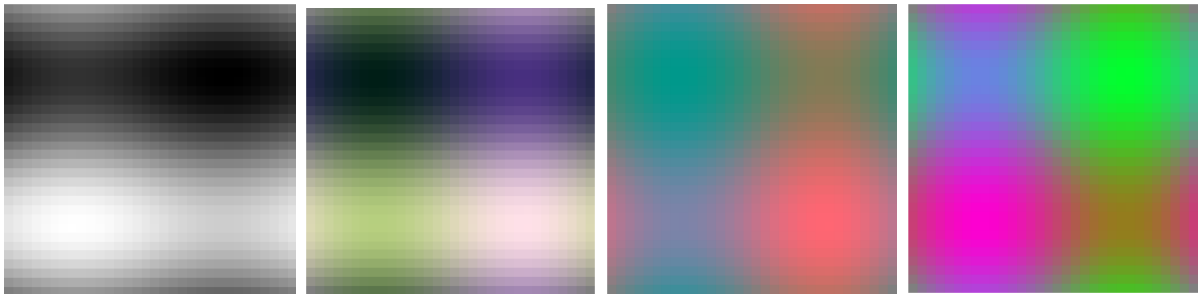

Figure 1: Visualization of Perlin noise patterns generated with different color complexity levels. From left to right: (a) Grayscale correlation, (b) High base correlation with slight variation, (c) Moderate correlation with increased variation, (d) Full channel independence.

4 Setup and Limitations

4.1 Experimental Setup

The setup used for the implementation and testing of the color-aware Perlin noise initialization method is as follows:

- Test Parameters:
  - Grid Configurations: $2^n \times 2^m$ where $n, m \in \{8, 12, 16\}$
  - Number of Samples: 500 samples per configuration
  - Color Complexity Testing:
    - Level 1: Perfect channel correlation ($R = G = B$)

    - Level 2: Base correlation 70% and 30% independent variation
    - Level 3: Reduced correlation 40% and 60% independent variation
    - Level 4: Fully independent channels
  - Image Dimensions: 32×32 pixels for compatibility with standard CNN input sizes

- Evaluation Datasets:
  - CIFAR-10: 60,000 colored images / 10 classes
  - CIFAR-100: 60,000 colored images / 100 classes
  - Omniglot: 38,300 grayscale images / 1,623 classes
  - DTD (Describable Textures Dataset): 5,640 colored images / 47 classes

- Network Architectures:
  - ResNet50: Used as a baseline in the original paper.

4.2 Computational Limitations

The implementation of the color-aware Perlin noise initialization method faced several important limitations from the outset. It was subject to technical and practical constraints:

- **Implementation Challenges:** Since the source code of the original paper was not available, we had to implement the entire system ourselves according to the explanations in the paper. Since there were too many parameters in this reimplementation that were not included in the paper, we tried to find the most optimal results by trying all of them and reproducing the logic. As a result, the initialization algorithm that emerged had difficulty producing the desired results or could not produce the result.

  During the implementation, since Resnet50 could only reach an accuracy level of 0.21 for Omniglot, we decided that the dataset was too weak for the model and decided not to conduct our experiments on it. You can access the experiments from the source code.

- **Computational Resources:** Our implementation was constrained by hardware limitations:
  - Maximum grid size of 16×16 (compared to the paper's 36×36)
  - Trained for 25 epochs to inspect short-term impacts (didn't mention in the original paper).
  - ResNet-50 model was used. All the following comparisons were made using ResNet-50. (When we use the ResNet-152 model, the grid size of the Perlin data set may need to be increased above our computational limits to see its effect.)

- **Dataset Processing:** The large-scale dataset processing capabilities implemented in the original paper cannot be achieved in our setup with limited resources. This issue affected the comprehensive evaluation of the method across different scenarios.

These difficulties made it considerably more difficult to replicate the conditions described in the paper precisely and probably prevented us from observing significant variations in outcomes. Here are the results we got when we tried to rewrite the original article:

Table 1: Performance Limitations according to the Original Paper's Implementation

Method	CIFAR-10	CIFAR-100	Omniglot	DTD
	R50	R50	R50	R50
Original Paper	93.76	77.42	17.54	55.03
Implemented Original Paper	81.06	60.91	0.21	21.60

Table 2: Observation of how the number of instances affects the quality of education. (N=16 M=16 for the grid sizes)

Method	DTD	
	50 instances	1000 instances
3 epochs pre-train[1], 10 epochs train[2]	12.87	22.13
3 epochs pre-train[1], 25 epochs train[2]	27.45	47.77

[1]Pre-train: Training the model on Perlin noise patterns for initialization. In 3 epochs the model learns the structure really well.
[2]Train: Fine-tuning the pre-trained model on the actual DTD dataset

In any case, the implementation presented here uses the best possible parameter setup to describe the behavior of color-aware Perlin noise initialization at smaller magnitudes, which is an important setting when resource constraints apply. To examine parameters such as the optimizer in detail, we have included the source code.

5 Results and Discussion

5.1 Implementation Results

We tested our color-aware Perlin noise implementation on the same datasets as the original paper and tested the importance of grids as variables. Table 2 shows us the success of the model trained with 5 epoch colored Perlin noise data and 25 epoch standard datasets depending on the grid size. When we examine the results, we see that the larger grid size generally increases accuracy performance. Therefore, it shows the highest success in all 16 datasets with the highest n and m values. This suggests that finer grid resolution allows for more nuanced noise patterns, beneficial for network initialization.

Table 3: Classification Accuracy (%) with Different Grid Configurations

Dataset	$\mathbf{2^8 \times 2^8}$	$\mathbf{2^{12} \times 2^{12}}$	$\mathbf{2^{16} \times 2^{16}}$
CIFAR-10	83.43	85.70	86.63
CIFAR-100	61.25	61.87	62.57
DTD	22.55	23.30	24.41

Also, you can take a look at Figure 2 in Appendix A to see an example training comparison. As you can see higher grid sizes have a more efficient training curve.

5.2 Comparison with Original Method

To evaluate the effectiveness of our color extension, we compared our method with the original Perlin noise initialization. Table 4 presents the comparison results.

Table 4: Performance Comparison with Original Perlin Noise Initialization

Method	**Dataset Accuracy (%)**		
	CIFAR-10	CIFAR-100	DTD
Implemented Original Method	81.06	60.91	21.60
Our Method	86.63	61.87	24.41

5.3 Discussion

The results of the implementation we call Color Aware Perlin Noise provide us with important insights in different areas. These findings can be examined under 3 main headings: performance improvements, computational considerations, and practical implications.

5.3.1 Performance Analysis

The color complexity levels used while initializing Perlin noise give us promising results on the grayscale model we implemented from paper. Especially in the CIFAR-10 dataset, it puts a significant improvement of 5.57 on the original implementation which has 86.63 and 81.06 accuracy. We can see similar but slightly smaller improvements on DTD with a 2.81 percentage point increase (24.41% vs 21.60%) and CIFAR-100 with 0.96 percentage points (61.87% vs 60.91%) improvement. The thing to note here is that since the CIFAR-100 implementation is much less successful than the implementation in the paper in the original, it is difficult to see the effect of the additions made to it.

These improvements show that working with color information during the initialization process helps the model to start from a much more successful starting point. This is especially evident in a dataset such as CIFAR-10, which is both more suitable for Resnet50 and stands out with its colorfulness.

5.3.2 Impact of Grid Configurations

Our experiments with different grid configurations ($2^8 \times 2^8$, $2^{12} \times 2^{12}$, and $2^{16} \times 2^{16}$) revealed a clear correlation between grid size and model performance. The steady improvements in model performance while training all datasets show that finer grid resolutions enable the generation of more nuanced noise patterns, which in turn provide better initialization conditions for the network.

The grid size-training curve graph of CIFAR-10 reported in Appendix A shows that the method not only helps to achieve better final accuracy but also helps to show stable and efficient training progress. This shows that the additional computation cost of increasing the grid size can be justified by the better results we get.

5.3.3 Computational Trade-offs

Despite these positive results, our implementation was subject to major computational and implementational difficulties, as we mentioned in Section 4.2. The relationship be-

tween model size and model success, as seen in Table 2, shows us that the method's full potential may only be realized with sufficient computational resources.

5.3.4 Practical Implications and Future Directions

Our findings allow us to make some important inferences for color-aware Perlin noise initialization.

1. Resource Scaling: We see that the performance of the method we applied increases with the increase in computational resources. This situation shows us that organizations with much more powerful computing power can obtain much more valuable data than the results obtained here.

2. Dataset Dependency: The fact that the effect of the results in some datasets is different from others shows that the effectiveness of this method can be task-dependent, and this method can be used much more in datasets rich in color.

3. Training Efficiency: Although the implementation requires additional computational overhead, the improved training dynamics and final accuracies show that these trade-offs can be worthwhile in many applications.

Future work could focus on the following topics: - Optimizing color complexity levels specifically for specific domains and tasks. - Researching and testing alternative color spaces other than RGB. - Using adaptive methods for color complexity levels and grid size selection. - Investigating hybrid approaches that combine color-aware Perlin noise with other techniques.

These findings add to our understanding of how neural network initialization strategies work and demonstrate that color-aware Perlin noise initialization is one promising direction in the improvement of deep learning models, especially when color information becomes important for a task at hand.

References

[1] Kenta Inoue, Takumi Kato, Hiroki Yoshimura, and Yoshiki Suzuki. Neural network initialization with perlin noise. *arXiv preprint arXiv:2101.07406*, 2021.

A Training Progression Details

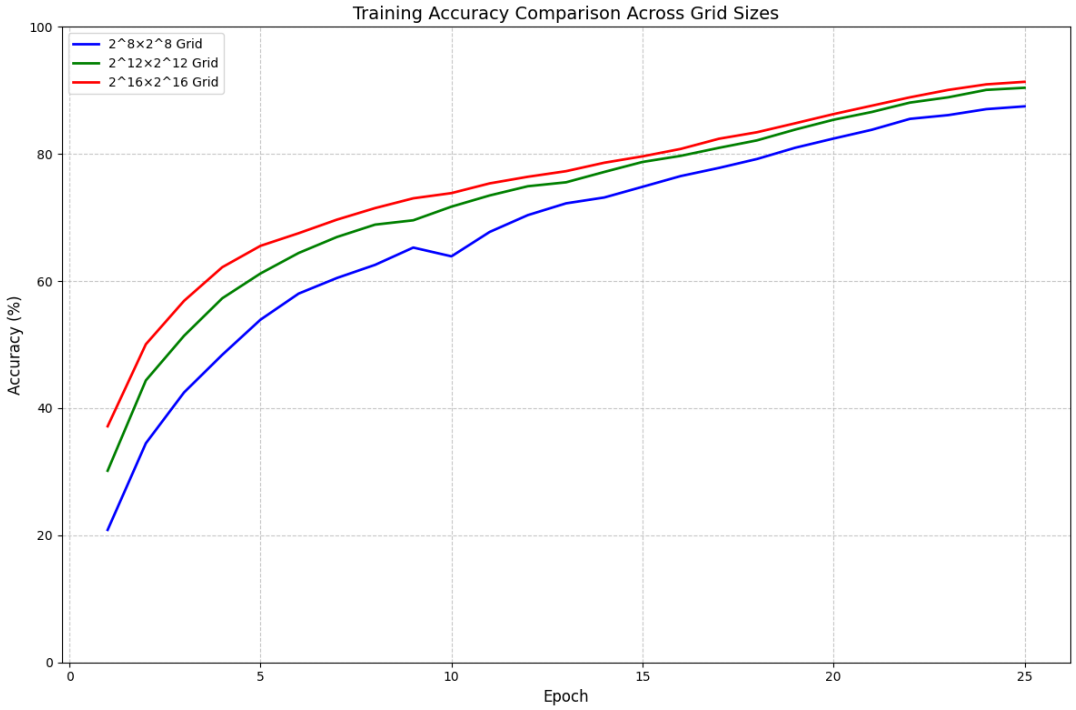

Figure 2: Comparison of grid size impacts on training for CIFAR-10.