How does the order of the training data influence the result?

Maximilien Gridel, Juliette Le Béchec and Réza Machraoui
Optimization for Machine Learning, EPFL, Switzerland

***Abstract*—The order of training examples is pivotal for efficient learning in modern neural networks. This report compares traditional random shuffling with structured strategies, such as Curriculum and Reverse Curriculum Learning, Self-Paced Learning, Hard-example Mining, and Stratified Monte Carlo Sampling on different datasets. By tracking accuracy and sample efficiency, the study clarifies the influence of the difficulty order of training samples on noisy datasets.**

I. Introduction

This project focuses on the impact of the order in which training data is presented on the performance of machine learning models. It was greatly inspired by previous scientific studies [1] and [2] that highlight the importance of data presentation order over two specific model performance : Curriculum Learning and Self-Paced Learning. We therefore decided to investigate further in the subject, and compare the influence of various data ordering strategies on the final accuracy of several models trained on different datasets (a toy dataset, MNIST, CIFAR) on which we applied different types and levels of noise (Gaussian and Impulse noise)

II. Models and Methods

Despite the increasing use of machine learning models, standard training procedures typically predominate. Recent studies [1], [2] suggest that more deliberate ordering strategies can significantly impact the performance of predicted models. However, these approaches often focus on specific datasets or models, and a comprehensive understanding of how different ordering techniques perform across diverse datasets and noise conditions is still lacking.

Our project aims to address this gap by evaluating various strategies listed below. To perform the most complete analysis possible, we explored various datasets (a toy dataset, MNIST, and CIFAR) on which we applied different noise levels allowing us to assign a difficulty score to individual data points more accurately. By doing so, we seek to uncover patterns and practical guidelines on when and why certain ordering methods lead to better or worse model accuracy.

Follows a description of the parameters chosen for each model.

A. *Toy Dataset*

1) Base Case: We used the model of logistic regression implemented with scikit-learn to provide a simple baseline. The data was randomly shuffled and split into 80% training and 20% testing sets.

2) Curriculum Learning: Difficulty was computed by the distance between each data point and the regression line fitted on the entire dataset, then the values were normalized to have a mean of zero. The underlying model used is linear regression.

3) Self-Paced Learning: The model adaptively includes harder samples over time by comparing current sample losses to a threshold. A StandardScaler was applied to normalize input features.

4) Hard-Example Mining: Using logistic regression, only the most challenging 25% of the training samples were retained for training.

5) Reverse Curriculum Learning: The samples were sorted by normalized difficulty in descending order before training. Logistic regression was again used as the base model.

6) Stratified Monte-Carlo Sampling: To ensure balanced exposure to all difficulty levels, normalized difficulty scores were divided into bins. For each stratum, a fixed number of samples were randomly selected. A logistic regression classifier was trained on the stratified sample.

B. *MNIST Dataset*

The MNIST dataset consists of grayscale images of handwritten digits, each 28×28 pixels with 1 channel [3]. We tested 9 versions: a clean dataset and 8 noisy variants : 4 with Gaussian noise and 4 with Impulse noise (at levels 0.1 to 0.4). Each point was randomly assigned a noise level, with an equal number of points at each difficulty level. Our CNN model (called "SmallCNN") captured meaningful performance distinctions. A random seed was used for coherency and reproducibility. In every MNIST experiments, learning rate was 0.001

1) Data Loading: MNIST was fetched with sklearn.datasets, tensorised, and split into 60k/10k train/test. Gaussian noise came from scaled random values, while impulse noise replaced random pixels with extremes, capped at 0.4 to keep digits legible.

2) Base Case: In the plain setup, we trained SmallCNN using randomly shuffled samples for one epoch. The model used cross-entropy loss.

3) Curriculum Learning (CL): For plain MNIST, difficulty was defined as the Euclidean distance from each image to the centroid of its class [4]. Training was staged across five batches of increasing difficulty to help the model progressively reach more ambiguous examples.

In the noisy versions, difficulty corresponded to discrete noise levels. Cumulative CL retained all examples from prior

stages, helping reinforce easier patterns as noise increased. Strict CL used only the samples from the current noise level, creating abrupt shifts between stages.

4) Self-Paced Learning: The model gradually includes training samples based on their loss, without predefined difficulty scores. It allows the model to avoid being overwhelmed and build confidence progressively.

In noisy datasets, training was extended to 10 epochs to give the model more adaptation time. An adapted threshold filtered out overly noisy examples until the model was ready to allow smoother transitions.

5) Hard-Example Mining: Only the 25% most difficult samples, based on normalized difficulty scores to test the model's ability to generalize from the most ambiguous data.

In noisy variants, we selected samples with noise level superior or equal to 0.3.

6) Reverse Curriculum Learning: The most difficult samples were trained first, then easier ones.

In the cumulative version, each stage retained earlier hard examples, preventing the model from forgetting patterns learned under noise. Strict Reverse CL trained on only one noise level per stage.

7) Stratified Monte-Carlo Sampling: To ensure balanced difficulty coverage, the dataset was partitioned into strata based on normalized difficulty and sampled each uniformly without reusing previously drawn samples to reduce variance.

In noisy variants, strata were defined by range of noise levels. Each stage trained only on one specific noise level, maintaining balance across levels.

C. *CIFAR Dataset*

The CIFAR-10 dataset consists of RGB images (3 channels) of size 32×32 pixels [5]. As with MNIST, the analysis was performed on a baseline and 8 noisy variants with Gaussian or Impulse noise applied at 4 difficulty levels (0.05, 0.1, 0.15, and 0.2). Noise was applied on existing points to keep a reasonable number of data points. Due to its higher complexity, the training phase used between 10 and 15 epochs. A random seed was used for coherency and reproducibility. The learning rate in every CIFAR experiments was 0.001. As the methodology largely mirrors that of MNIST, mainly the differences will be noted.

1) Data Loading: Data were loaded from torchvision.datasets.CIFAR10 and converted into tensors. The training set includes 50,000 images and the test set 10,000.

For the noisy datasets, difficulty was generated by applying increasing levels of Gaussian and Impulse noise to the dataset in a similar way than MNIST. Noise was added per channel to preserve the RGB format.

2) Base Case (Plain Dataset): SmallCNN was trained on shuffled CIFAR for 10 epochs to ensure convergence.

3) Curriculum Learning (CL): Sample difficulty was computed as the distance to class centroids in flattened image space [4]. Training was structured into 5 stages over 10 epochs, with samples increasing in difficulty.

In noisy datasets, Strict CL limited each stage to one specific noise level, potentially creating abrupt learning transitions.

4) Self-Paced Learning: Self-paced training lasted 15 epochs. The model selected enables training set to focus on samples within its current capacity.

In noisy settings, training used an initial threshold $\lambda = 2.25$ with an increment of 5 per epoch, allowing gradual expansion of the training set.

5) Hard-Example Mining: We selected the top 40% most difficult examples, with 20,000 training samples over 50,000.

For noisy data, samples with noise level ≥ 0.15 were retained across combined datasets, totaling 17,280 hard samples out of 43,200. This filtered dataset emphasized the model's ability to learn under noise stress.

6) Reverse Curriculum Learning: Cumulative Reverse CL retained previously seen hard examples in later stages to mitigate forgetting. Strict Reverse CL limited each stage to a unique noise level and did not retain prior samples.

7) Stratified Monte-Carlo Sampling: The dataset was divided into strata based on difficulty, with approximately equal sample counts drawn from each. This helped assess model performance under uniformly distributed difficulty exposure.

In noisy datasets, one noise level was sampled per stage, ensuring controlled evaluation across levels.

III. Results

Only the most significant results are introduced and represented, and all the results can be found in the notebook.

A. ***Toy Dataset*** *(TABLE I, Fig 1. & 2).*

1) Base Case: Random sampling achieves remarkable results, with a final accuracy of 82% and an overall 8% increase. Despite a few downs, increasing difficulty seems to improve the accuracy of the model.

2) Self-Paced Learning: Fig 2. illustrates that the best result is obtained when training over Self-Paced Learning, with an improvement of accuracy close to 35% after 6 iterations.

3) Hard-Example Mining: Focusing exclusively on the most difficult samples achieves comparable accuracy to random sampling while requiring significantly less training data.

4) Stratified Monte-Carlo Sampling: Variance reduction techniques achieve strong performance early on, particularly with low-difficulty samples. Performance stagnates beyond the 50% difficulty threshold: most of the useful information seems to be captured in the less challenging portions of the dataset.

B. ***MNIST Dataset*** *(TABLE II, Fig. 3 to 8)*

1) Base Case: Trained on fully shuffled batches, SmallCNN attains 98.82% test accuracy on clean MNIST, 98.58% with additive Gaussian noise, and 97.67% under Impulse noise.

2) Curriculum Learning (CL): On clean MNIST, CL reaches 97% accuracy after only 30,000 samples, then flattens. For the noisy variants, the cumulative version rise smoothly as more images are added, whereas strict CL's accuracy decreases when encountering harder examples, especially for impulse noise. Both reach 98% final accuracy, except for the strict variant under Impulse Noise ($\approx 95\%$).

3) Hard-Example Mining: Accuracy on clean MNIST increases significantly by 0.7 over 10 epochs. Both noisy variants reach over 98% of accuracy in only a few epochs. The added noise produces a far denser set of initial misclassifications, giving hard-example mining a large supply of informative "difficult" samples to replay from the very first pass.

4) Reverse Curriculum Learning: As expected, accuracy starts low but increases as the samples introduces get easier, to reach 99% on plain MNIST. With Impulse noise, Cumulative Reverse variant starts the lowest to mark the greatest improvement of all (15%). The tendency persists under Gaussian noise, although less pronounced.

5) Stratified Monte-Carlo Sampling: Clean MNIST reaches 98% accuracy after the first stage, then stagnates. Noisy datasets's accuracy improve at first, although slips sligthly towards the end. They achieve a slightly lower ending accuracy.

C. ***CIFAR-10 Dataset*** *(TABLE III, Fig. 9 to 14)*

1) Base Case: On Vanilla dataset, SmallCNN reaches 72% accuracy. Performance is slightly reduced with Gaussian noise but significantly drops with Impulse noise (56%). In all three cases, accuracy improves gradually and stabilizes around the 4^{th} or 5^{th} epoch. This suggests that even without any data ordering strategy, the model learns reasonably well.

2) Curriculum Learning (CL): In the vanilla case, CL's accuracy sharply increases of 18% over the first three stages, followed by a slight drop. Both noise variants show steady improvement, particularly for the Impulse noise.

3) Self-Paced Learning: Across all settings, this method produces rapid gains in the first few epochs, then stabilizes. Its impact is doubling or even tripling the initial accuracy, showing that gradually including harder samples based on loss is an effective strategy.

4) Hard-Example Mining: This approach shows modest improvement in the vanilla case, but performs better with Gaussian and Impulse noise with consistent but limited gains. Performance stabilizes by epoch 6 for Impulse noise. This suggests that most of the useful learning signal is concentrated in a subset of difficult samples.

5) Reverse Curriculum Learning: This technique appears counterproductive in the vanilla setting: accuracy decreases by 10% as simpler examples are introduced. However, under Gaussian noise both variants improve continuously and achieve the best final performance. With Impulse noise, there is an initial boost when trained on hard examples, but accuracy drops afterwards, especially for Strict Reverse CL. This may indicate that the easier examples disrupt the model's previously acquired robustness.

6) Stratified Monte-Carlo Sampling: Under Gaussian noise, under-performance gap compared to other methods improves rapidly and narrows by the 5th training stage. This shows that most of the relevant information is already captured early on, so the benefit of further data decreases as training progresses. Under Impulse noise, accuracy increases steadily over stages, but drops slightly when encountering harder examples.

7) Comments: The difficulty heuristics employed (derived from centroid distance and addition of noise) may not reliably reflect true semantic difficulty in high-dimensional datasets like CIFAR. As a result, strategies based on these heuristics did not consistently outperform random sampling.

IV. Discussion

Overall, results suggest that selectively training on informative samples can lead to efficient learning. While traditional random sampling offers reasonable baseline performance, more structured approaches often lead to faster convergence and better generalization, particularly under noisy conditions. While Curriculum Learning enables the model to reach high accuracy with fewer training samples, especially in moderately complex datasets, Self-Paced Learning outperforms all methods independently of the type and level of Noise attributed both in convergence time and final accuracy. Hard-example mining performs well although demonstrates a steady slight increase. Reverse Curriculum Learning lags behind when forced to master the hardest examples first, but recovers when easier examples are later introduced, especially in the cumulative version. Stratified sampling provides fast convergence but limited long-term gains.

These findings suggest that intelligent data ordering is a simple but powerful tool to improve training efficiency and robustness, particularly when dealing with corrupted or ambiguous data.

V. Summary

Gradually introducing harder examples enables more effective learning compared to immediate exposure to the full dataset. Our experiments demonstrate that the order in which training data is presented significantly influences model accuracy, especially under noisy conditions.

Across all datasets (Toy, MNIST, and CIFAR), Self-Paced Learning consistently outperforms other strategies by adaptively including more difficult samples based on model feedback, leading to faster and more stable accuracy gains. Curriculum Learning also proves highly effective, particularly in simpler or moderately noisy contexts, where its structured progression from easy to hard samples helps accelerate early learning.

Hard-Example Mining shows that focusing on the most difficult examples alone can yield comparable performance using fewer data points, though it may limit generalization if not combined with easier instances. In contrast, Reverse Curriculum Learning performs poorly when hard examples dominate early stages but recovers when easier data are reintroduced, especially in cumulative variants. Lastly, Stratified Monte Carlo Sampling provides a balanced compromise, promoting rapid early learning by reducing sampling variance, though its improvements stabilize in later training stages.

In summary, our findings confirm that data ordering is a powerful tool to improve training efficiency and generalization in neural networks. Future research could explore hybrid strategies that combine multiple orderings or use model-driven signals to adaptively adjust the curriculum in real time.

APPENDIX

The most interesting results are shown in the graphs below.

A. *Toy Dataset*

Method	Accuracy
Base	0.82
Curriculum Learning	0.82
Self-Paced Learning	**0.85**
Reverse Curriculum Learning	0.83
Hard-Example Mining	0.82
Stratified Monte-Carlo Sampling	0.83

TABLE I: Final Accuracy on Toy Dataset

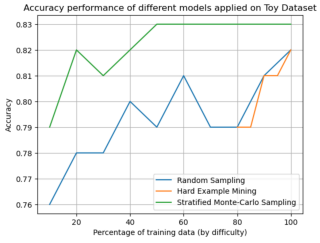

Fig. 1: Comparison of Validation Accuracy accross models, Toy Dataset

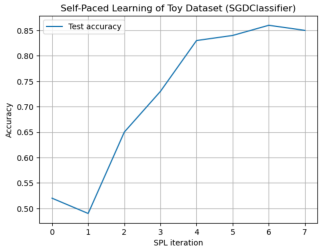

Fig. 2: Validation Accuracy of Self-Paced Learning, Toy Dataset

B. *MNIST Dataset*

Noise Type	Method	Variant	Accuracy
Vanilla	Base	–	0.9822
	Curriculum Learning	–	0.9481
	Self-Paced Learning	–	0.7623
	Reverse Curriculum Learning	–	**0.9579**
	Hard-Example Mining	–	0.3961
	Stratified Monte-Carlo Sampling	–	0.9813
Gaussian Noise	Base	–	0.9858
	Curriculum Learning	Cumulative	0.9873
	Curriculum Learning	Strict	0.9866
	Self-Paced Learning	–	**0.9892**
	Reverse Curriculum Learning	Cumulative	0.9866
	Reverse Curriculum Learning	Strict	0.9813
	Hard-Example Mining	–	0.9752
	Stratified Monte-Carlo Sampling	–	0.9731
Impulse Noise	Base	–	0.9767
	Curriculum Learning	Cumulative	0.9765
	Curriculum Learning	Strict	0.9743
	Self-Paced Learning	–	**0.9799**
	Reverse Curriculum Learning	Cumulative	0.9765
	Reverse Curriculum Learning	Strict	0.9444
	Hard-Example Mining	–	0.9786
	Stratified Monte-Carlo Sampling	–	0.9561

TABLE II: Final Accuracy on MNIST Dataset

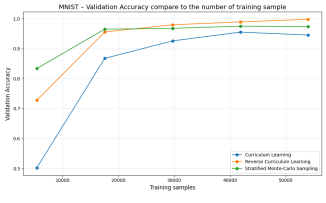

Fig. 3: Validation Accuracy vs Training Samples, MNIST (Vanilla)

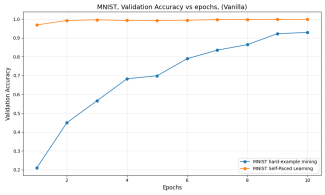

Fig. 4: Validation Accuracy vs Epochs, MNIST (Vanilla)

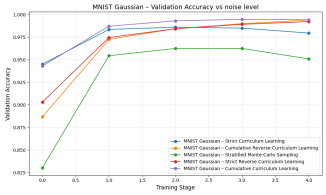

Fig. 5: Validation Accuracy vs Stage, MNIST (Gaussian Noise)

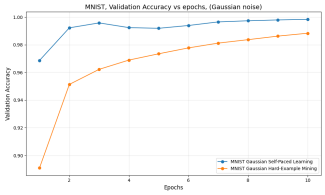

Fig. 6: Validation Accuracy vs Epochs, MNIST (Gaussian Noise)

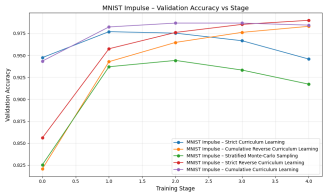

Fig. 7: Validation Accuracy vs Stage, MNIST (Impulse Noise)

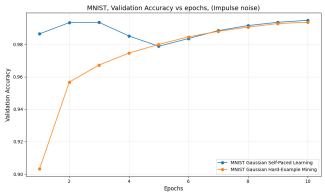

Fig. 8: Validation Accuracy vs Epochs, MNIST (Impulse Noise)

C. CIFAR Dataset

Noise Type	Method	Variant	Accuracy
Vanilla	**Base**	–	**0.7199**
	Curriculum Learning	–	0.5859
	Self-Paced Learning	–	0.6872
	Reverse Curriculum Learning	–	0.5220
	Hard-Example Mining	–	0.5994
	Stratified Monte-Carlo Sampling	–	0.6323
Gaussian Noise	**Base**	–	**0.7126**
	Curriculum Learning	Cumulative	0.6470
	Curriculum Learning	Strict	0.6547
	Self-Paced Learning	–	0.6946
	Reverse Curriculum Learning	Cumulative	0.6605
	Reverse Curriculum Learning	Strict	0.6651
	Hard-Example Mining	–	0.6348
	Stratified Monte-Carlo Sampling	–	0.6094
Impulse Noise	Base	–	0.5531
	Curriculum Learning	Cumulative	0.5343
	Curriculum Learning	Strict	0.5359
	Self-Paced Learning	–	**0.5601**
	Reverse Curriculum Learning	Cumulative	0.4918
	Reverse Curriculum Learning	Strict	0.4713
	Hard-Example Mining	–	0.5317
	Stratified Monte-Carlo Sampling	–	0.5213

TABLE III: Final Accuracy on CIFAR Dataset

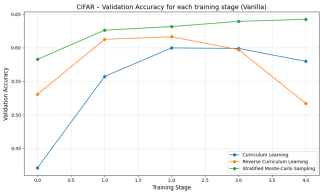

Fig. 9: Validation Accuracy vs Stage, CIFAR (Vanilla)

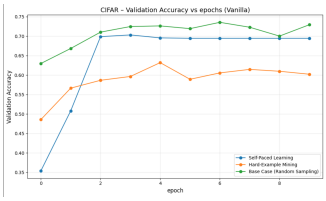

Fig. 10: Validation Accuracy vs Epochs, CIFAR (Vanilla)

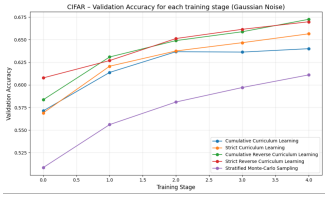

Fig. 11: Validation Accuracy vs Stage, CIFAR (Gaussian Noise)

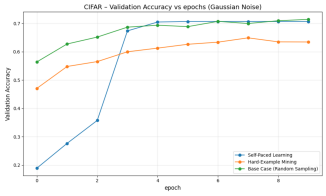

Fig. 12: Validation Accuracy vs Epochs, CIFAR (Gaussian Noise)

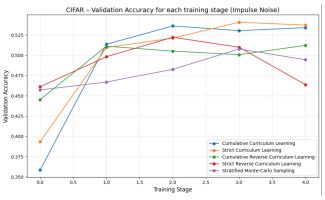

Fig. 13: Validation Accuracy vs Stage, CIFAR (Impulse Noise)

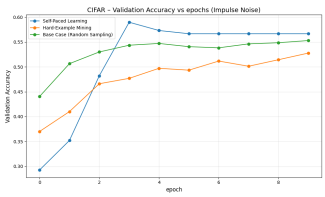

Fig. 14: Validation Accuracy vs Epochs, CIFAR (Impulse Noise)

REFERENCES

[1] R. C. Yoshua Bengio, Jérôme Louradour and J. Weston, "Curriculum learning," in *Proceedings of the 26th International Conference on Machine Learning (ICML)*, 2009. [Online]. Available: https://www.researchgate.net/publication/221344862_Curriculum_learning

[2] B. P. M. Pawan Kumar and D. Koller, "Self-paced learning for latent variable models," in *Advances in Neural Information Processing Systems (NIPS)*, 2010. [Online]. Available: https://www.researchgate.net/publication/221620417_Self-Paced_Learning_for_Latent_Variable_Models

[3] J. Brownlee, "How to develop a convolutional neural network from scratch for mnist handwritten digit classification," 2019. [Online]. Available: https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-from-scratch-for-mnist-handwritten-dig

[4] R. Ramasubramanian, A. Ghosh, A. Vyas, and P. Rai, "Difficulty-based sampling in curriculum learning," 2024. [Online]. Available: https://arxiv.org/abs/2402.07352

[5] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html