# MITIGATING SIMPLICITY BIAS IN NEURAL NETWORKS: A FEATURE SIEVE MODIFICATION, REGULARIZATION, AND SELF-SUPERVISED AUGMENTATION APPROACH

**Gaurav Joshi***, **Rachit Verma** *
Department of Computer Science
Indian Institute of Technology Gandhinagar
Gandhinagar, Gujarat, 382355, India
{joshigaurav,vermarachit}@iitgn.ac.in

**Parth Narendra Shah***
Department of Electrical Engineering
Indian Institute of Technology Gandhinagar
Gandhinagar, Gujarat, 382355, India
shahparth@iitgn.ac.in

## ABSTRACT

Neural networks (NNs) are known to exhibit simplicity bias, where they tend to prioritize learning simple features over more complex ones, even when the latter are more informative. This bias can result in models making skewed predictions with poor out-of-distribution (OOD) generalization. To address this issue, we propose three techniques to mitigate simplicity bias. One of these is a modification to the Feature Sieve method. In the second method we utilize neuronal correlations as a penalizing effect to try and enforce the learning of different features. The third technique involves a novel feature-building approach called Self-Supervised Augmentation. We validate our methods' generalization capabilities through experiments on a custom dataset.

## 1 INTRODUCTION

Motivated by the need to understand generalization in deep learning, there has been a surge of studies focusing on the function classes favored by current training techniques for large neural networks (Morwani et al., 2023; Zhang et al., 2022), (Zhang et al., 2021). A growing hypothesis suggests that deep learning methods prefer learning simple functions over the data. While this inductive bias helps prevent overfitting and improves in-distribution generalization in many cases, it proves inadequate in certain scenarios. Neural networks exhibit a bias for simple features: given two features with equal predictive power on the training set, gradient-based methods often cause the network to prioritize learning the simpler features. This preference can reduce robustness to adversarial samples and hinder OOD generalization.

**Definition 1.1 (Low dimensionality simplicity bias).** A model $f : \mathbb{R}^d \to \mathbb{R}^c$ with inputs $x \in \mathbb{R}^d$ and outputs $f(x) \in \mathbb{R}^c$ (e.g., logits for $c$ classes), trained on a distribution $(x, y) \sim \mathcal{D}$ satisfies low dimensional simplicity bias if there exists a projection matrix $P \in \mathbb{R}^{d\times d}$ satisfying:

- $\text{rank}(P) = k \ll d$,
- $f(Px^{(1)} + P_\perp x^{(2)}) \approx f(x_1) \quad \forall (x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}) \sim \mathcal{D}$,
- An independent model $g$ trained on $(P_\perp x, y)$ where $(x, y) \sim \mathcal{D}$ achieves high accuracy.

We identify key limitations in existing methods and propose solutions, validated through experiments on a custom dataset. Our contributions include a modified feature sieve method for better forgetting gradient flow across layers, a novel regularization term to reduce excessive Neuronal Correlation within batches, a feature mapping technique called Self-Supervised Augmentation that enhances complex feature learning through forced reconstruction, and the creation of a novel CIFAR10-MNIST dataset designed for robust evaluation of our methods.

*All authors contributed equally to this work.

# 2 METHODOLOGY

## 2.1 WEIGHTED FORGETTING SIEVE

The Feature Sieve in Tiwari & Shenoy (2023) employs a 'forgetting gradient' to encourage lower layers to discard simpler features in favor of more complex ones, but its effectiveness is limited by gradient flow constraints. To address this, we propose attaching sieves to multiple layers, enabling direct propagation of forgetting gradients to their respective layers while blocking their influence on lower layers to stabilize training. Additionally, we introduce a weighted forgetting loss, where lower layers receive stronger forgetting gradients, following a decreasing power series on the weights. To manage computational costs, this attachment can be restricted to a fixed number of lower layers, ensuring efficiency without excessive overhead. An illustration of this concept can be found in Figure 1b, and the pseudocode for this algorithm can be found in Appendix section A.2.

## 2.2 CORRELATION REGULARIZATION

When a model predominantly relies on simpler features to make predictions, the feature maps in the lower layers tend to exhibit high correlation along the channel dimension. Jin et al. (2022) introduced a metric to penalize this excessive correlation, which has been shown to improve out-of-distribution (OOD) generalization. While some degree of correlation is necessary for aggregating information from lower to upper layers, excessive correlation in lower-layer activations indicates simplicity bias, hindering the network from capturing more diverse local features. To mitigate simplicity bias, we introduce a regularization term in the loss function that penalizes excessive correlation in the feature maps of lower layers along the channel dimension. The regularization term is computed by first extracting channel-wise values for a each spatial dimension, then constructing a covariance matrix between channels for each spatial dimension using values from multiple samples in the batch. The final regularization penalty is obtained by summing the absolute values of the off-diagonal elements in the covariance matrix across all spatial dimensions, ensuring reduced redundancy in learned features. Details on the mathematical formulation of Neuronal Correlation can be found in Appendix section A.3. An illustration of this concept can be found in 1a.

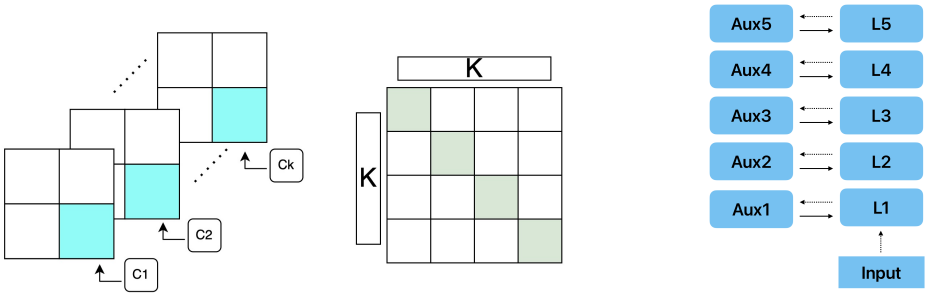
## 2.3 SELF SUPERVISED AUGMENTATION

In the above methods, we aim to motivate the model to learn higher-complexity features. Instead of directly forcing the model, we employ a self-supervised method to encode these features in a compressed representation. Specifically, we construct an autoencoder, where the encoder part mirrors the initial layers of the deep learning model. The autoencoder is trained to take an image as input and reconstruct the same image as output. This forces the model to capture all useful features in the intermediate representation, as it must learn these features to accurately reconstruct the image. After training the autoencoder, we extract the learned weights and integrate them into the deep learning model, freezing these layers. The model is then fine-tuned on the classification task. An illustration of this concept can be found in 2.

# 3 DATASET AND RESULTS

We build a custom *CIFAR10-MNIST* dataset that combines the *CIFAR-10* and *MNIST* datasets using a transparency factor $\alpha = 0.3$, creating a superimposed representation with both feature sets to evaluate our methods for this task. We experiment with a simple 5 layer CNN model. We refer to the plain model as Simple CNN.

The first task, known as the two-image task, involved training on image pairs: "plane" + "0" and "car" + "1". During testing, the model encountered novel pairs not found in training: "plane" + "1" and "car" + "0". The goal was to correctly classify the CIFAR image while preventing reliance on the MNIST features. The results for this task for all the methods are presented in Table 1. As visible, the Weighted Forgetting Sieve outperforms the Feature Sieve. Correlation Regularization on the Simple CNN also exhibits an improvement over traditional training. Self-supervised Augmentation achieves the best results among all the methods compared.




(a) Correlation Regularization (b) Weighted Forgetting Sieve

Figure 1: Visualization and explanation of key components: (a) Correlation Regularization, where for a fixed spatial coordinate, values along the channel and batch dimensions are collected to construct the covariance matrix between different channels. In the final loss term, all the non-shaded terms in the covariance matrix are summed up. (b) Weighted Forgetting Sieve, where each layer has an auxiliary layer attached to it. The dotted arrows denote the flow of feature maps during the forward pass. The solid arrows denote the flow of forgetting gradient during the backward pass. As is visible, each layer receives its own forgetting gradient.

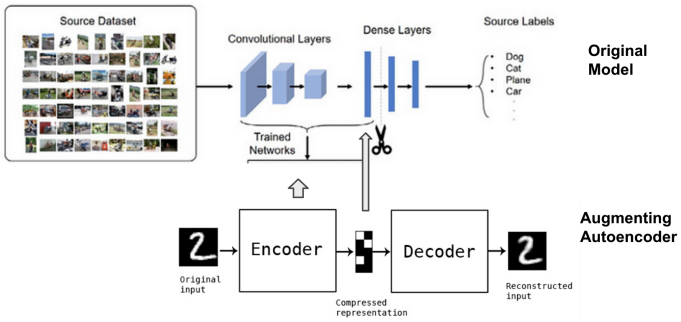



Figure 2: Architecture for the Self-Supervised Augmentation Method

We also tested our best performing method, Self-Supervised Augmentation, on the three-image test. Its training set included {“plane” + “0”}, {“car” + “1”}, and {“bird” + “2”}, while the test set included all possible combinations of these images. As evidenced by 3, the Self-supervised Augmented CNN exhibits higher cross-pair accuracy than the simple CNN, indicating that our method can successfully learn higher-complexity features. Additional experiments and their results can be found in Appendix section A.1.

| MNIST Class | CIFAR Class | | |
|---|---|---|---|
| | Plane | Car | Bird |
| 0 | 99.4 | 5.6 | 11.8 |
| 1 | 3.8 | 99.2 | 25.2 |
| 2 | 17.6 | 24.4 | 99.0 |

| MNIST Class | CIFAR Class | | |
|---|---|---|---|
| | Plane | Car | Bird |
| 0 | 88.0 | 20.0 | 46.0 |
| 1 | 15.8 | 93.2 | 38.4 |
| 2 | 41.2 | 28.4 | 77.8 |

Figure 3: Results for the three-image test. (Left) Simple CNN (Right) Self-supervised Augmented CNN.

## 4 Future Work and Conclusion

Neural networks have been shown to exhibit a bias for learning simple features in favor of more complex features. This affects their robustness and out-of-distribution generalization. The methods presented in this paper include the Weighted Forgetting Sieve, Correlation Regularization and Self-Supervised Augmentation. The results from our designed tasks demonstrate the effectiveness

Table 1: Results for the two-image pair task

| Model | Training Accuracy (%) | Test Accuracy (%) |
|---|---|---|
| Simple CNN | 97.3 | 8.6 |
| Feature Sieve | 97.0 | 25.0 |
| Weighted Forgetting Sieve | 94.0 | 36.0 |
| Correlation Regularization | 84.0 | 32.0 |
| Self-supervised Augmentation | 84.0 | **46.0** |

of these methods in reducing simplicity bias in neural networks. Future work includes applying these methods in conjunction with one another, and theoretically grounding these works to better understand their advantages and limitations.

## REFERENCES

Gaojie Jin, Xinping Yi, and Xiaowei Huang. Neuronal correlation: a central concept in neural network, 2022. URL https://arxiv.org/abs/2201.09069.

Depen Morwani, Jatin Batra, Prateek Jain, and Praneeth Netrapalli. Simplicity bias in 1-hidden layer neural networks. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Rishabh Tiwari and Pradeep Shenoy. Overcoming simplicity bias in deep networks using a feature sieve. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. volume 64, pp. 107–115, New York, NY, USA, February 2021. Association for Computing Machinery. doi: 10.1145/3446776. URL https://doi.org/10.1145/3446776.

Jianyu Zhang, David Lopez-Paz, and Leon Bottou. Rich feature construction for the optimization-generalization dilemma. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 26397–26411. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/zhang22u.html.

# A APPENDIX

## A.1 ADDITIONAL RESULTS ON THE CIFAR10-MNIST DATASET

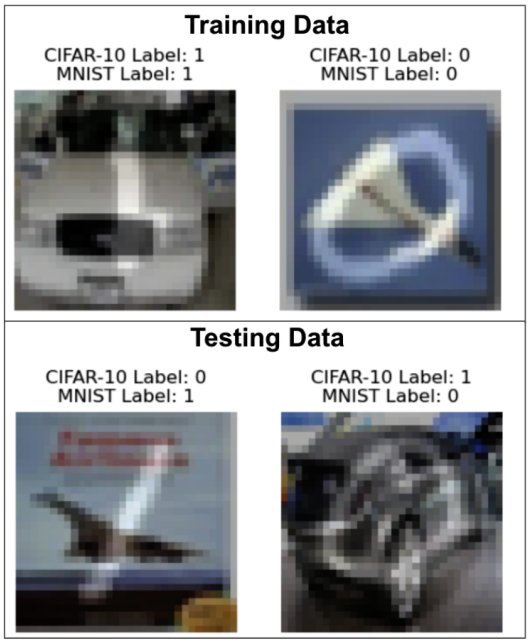



Figure 4: Sample images from the training and test datasets for the two-image task.

**Variable Alpha Test.** We tried varying the transparency value, $\alpha$, to investigate its effect on the results. The $\alpha$ values we tested with were $\{0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$. During the Simple CNN training stage, for $\alpha = 0.05$, the accuracy increased gradually with epochs. However, for $\alpha = 0.5$, the accuracy reached 85% in epoch 1 and 96% by epoch 5, indicating that the model was primarily learning simpler features. We also observed that for $\alpha$ values of 0.05 and 0.1, where the digits were barely visible, both the Simple CNN and the Self-Supervised Augmented CNN performed similarly. However, for $\alpha = 0.2$ and higher, the test loss on the cross terms dropped significantly for the Simple CNN, while it remained declined much less rapidly for the Self-Supervised Augmented CNN.

**Ten-image Task.** In the ten-image task, we scale up the tests to include all 10 labels of the CIFAR10 and MNIST datasets. As shown by the results in 5, the model on average improves the accuracy of the Simple CNN by almost 25%.

**Base Model Variations.** We also explored two base model variations to analyze their impact on results. First, we increased the number of model parameters to examine its effect on performance. Without augmentation, the results for the Simple CNN remained unchanged compared to the smaller model, but after applying Self-supervised Augmentation, they were better than even the smaller augmented model. Second, we tested training without freezing weights, allowing gradients to flow through all layers. As expected, this led to a slight decrease in accuracy compared to our approach, where weights are frozen after transfer.

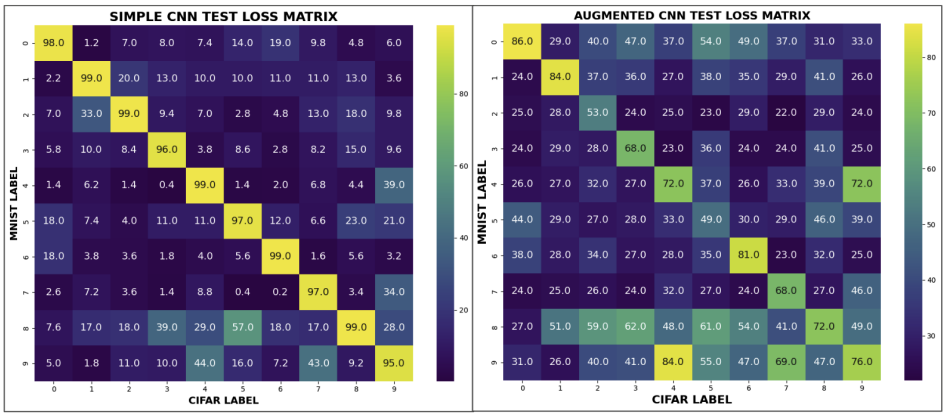



Figure 5: Ten-image task result. The Simple CNN has very low accuracy in the cross terms, while the Self-Supervised Augmented CNN has significantly better cross term accuracies, denoting its superior generalization capabilities.

### A.2 Pseudocode for Weighted Forgetting Sieve

**Algorithm 1** Weighted Forgetting Sieve

**Require:** Neural network model $\mathcal{M}$, importance weights $\mathbf{w}$, input image $\mathbf{x}$, optimizer $\mathcal{O}$, scheduler $\mathcal{S}$, number of classes $C$
**Ensure:** Updated model with selectively forgotten information
Create uniform distribution target $\mathbf{y} \leftarrow \frac{1}{C}\mathbf{1}_{B\times C}$ {$B$ is batch size}
Set $\mathcal{M}$ to evaluation mode
$\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, \mathbf{s}_4, \mathbf{s}_5, \leftarrow \mathcal{M}(\mathbf{x})$ {Forward pass through the model}
$\mathcal{O}$.zero_grad() {Reset gradients}
**for** $i \in \{1, 2, 3, 4, 5\}$ **do**
    Set $\mathcal{M}$ to evaluation mode
    Set sieve$_i$ to training mode
    Set corresponding core network components to training mode:
        - conv layer$_i$, pooling layer$_i$, batch norm$_i$
    $\ell_i \leftarrow w_i \cdot \text{BCE}(\mathbf{s}_i, \mathbf{y})$ {Weighted binary cross-entropy loss}
    $\ell_i$.backward {Compute gradients}
**end for**
$\mathcal{O}$.step() {Update model parameters}
**if** $\mathcal{S}$ is not None **then**
    $\mathcal{S}$.step() {Update learning rate schedule}
**end if**
**return** $\mathcal{M}$ {Return updated model}

### A.3 Neuronal Correlation Definition

Let $T_l \in \mathbb{R}^{n\times A\times B}$ represent the feature maps at the $l$-th convolution layer, and let $T_{liab}$, where $i \in \{1, \ldots, n\}$, represent the output of the $i$-th feature map in the $l$-th layer at the spatial dimension $(a, b)$.

We define the correlation regularization term $\rho(T_l)$ as:

$$\rho(T_l) = \sum_{b=1}^{h}\sum_{a=1}^{w}\sum_{\substack{i,j=1\\ i\neq j}}^{n}\left|\frac{\rho(T_{liab}, T_{ljab})}{2}\right|$$

where:

$$\rho(T_{liab}, T_{ljab}) = \frac{\text{cov}(T_{liab}, T_{ljab})}{\sigma_{T_{liab}} \sigma_{T_{ljab}}}, \quad \rho(T_{liab}, T_{ljab}) \in [-1, 1].$$

Intuitively, $\rho(T_l)$ represents the **total pairwise correlation between channels in the feature maps** of the $l$-th layer.