Advances in Artificial Intelligence and Machine Learning; Research 5 (1) 3605-3626 Received 01-01-2025; Accepted 24-03-2025; Published 31-03-2025

Improving CNN Robustness to Color Shifts via Color Balancing and Spatial Dropout

Pradyumna Elavarthi elavarpa@mail.uc.edu
Department of Computer Science
University of Cincinnati
Cincinnati, OH 45221, USA

Anca Ralescu ralescal@ucmail.uc.edu
Department of Computer Science
University of Cincinnati
Cincinnati, OH 45221, USA

Corresponding Author: Pradyumna Elavarthi

Abstract

Convolutional neural networks (CNNs) have demonstrated remarkable success in vision-related tasks. However, their susceptibility to failing when inputs deviate from the training distribution is well-documented. Recent studies suggest that CNNs exhibit a bias toward texture instead of object shape in image classification tasks, and that background information may affect predictions. This paper investigates the ability of CNNs to adapt to different color distributions of an image while maintaining context and background. The results of our experiments on modified MNIST, CIFAR10 and CIFAR 100 data demonstrate that changes in color can substantially affect classification accuracy. The paper explores the effects of various regularization techniques on generalization error across datasets and proposes an architectural modification using in a novel way color balancing and spatial dropout regularization. This enhances the model reliance on color-invariant intensity-based features for improved classification accuracy. Overall, this work contributes to ongoing efforts to understand the limitations and challenges of CNNs in image classification tasks and offers a potential solution to improve their performance.

Keywords: Color shifts, Invariant learning, Out-of-distribution generalization

Citation: Pradyumna Elavarthi and Anca Ralescu. Predicting COVID-19 Outcomes Among Albertans With Diabetes and COVID-19: A Machine Learning Approach. Advances in Artificial Intelligence and Machine Learning. 2025;5(1):205.

1. INTRODUCTION

Deep learning models have made significant advancements in various domains, particularly in vision and natural language processing [1, 2]. Among the popular architectures, convolutional neural networks (CNNs) have emerged as powerful tools for visual tasks, finding applications in diverse fields from medicine to construction. The success of CNNs can be attributed not only to the increased computational capacity but also to their remarkable generalization capabilities [3]. However, recent studies have uncovered potential challenges associated with the generalization of CNNs, revealing their susceptibility to learning non-causal features and their limited performance on distributions different from the training data [4–7].

One fundamental objective of machine learning is to develop models that demonstrate high modeling and generalization performance, that is, on the training and unseen test data respectively. Overfitting occurs when a model becomes overly specialized to the training data, capturing spurious patterns that do not exist in the broader population, leading to poor generalization despite low training error. To address this issue, various regularization techniques have been devised to improve the model's generalization by mitigating the impact of overfitting. These techniques include weight penalization to discourage excessive importance on specific features and dropout regularization, which introduces randomness by probabilistically excluding neurons, thereby encouraging the model to rely on other informative features.

Pooling operations in CNNs contribute to their translation invariance, while data augmentation plays a crucial role in enhancing their overall robustness. However, studies have shown that certain distributional shifts, such as maintaining the shape while altering the texture of objects, can bias CNNs towards prioritizing texture over shape information. It has been proposed [8], that improving the shape bias of models can enhance their robustness against such perturbations. Furthermore, investigations into the shape bias property of CNNs [5], revealed that CNNs do not inherently exhibit a strong shape bias, leading to reduced accuracies in detecting negative images. The authors argued that negative images, which are semantically similar to normal images, can pose challenges for CNNs and demonstrated their limitations in this regard.

We investigate the generalization ability of CNNs in classifying images sampled from different color distributions while maintaining pixel intensities identical to the training data. Our study focuses on multiple representations of the datasets: custom-colored versions of MNIST [9], and a more complex dataset, CIFAR10 [10], which is employed to validate the obtained results on real-world images. Specifically, we modify the MNIST data to create three distinct datasets: the first containing only green color, the second with a single-color channel, and the third incorporating all three-color channels. Also, colored jitter is added to these datasets to see if the introduction of colored noise influences generalization. Equal numbers of examples are assigned to each class label within each color variant, ensuring no inherent correlation between color and class label. As a result, these datasets are semantically similar, as color conveys no additional information for classification purposes.

Through systematic experiments, we empirically show that CNNs trained using traditional layers like convolutions, Batch normalization and Max Pooling cannot generalize well to the distributional changes in color. So, we use a color balancing and spatial dropout [11], layer before using convolutions on the input image as a form of regularization to improve the model's reliance on color invariant features for classification. We show experimentally that this method improves the generalization ability of CNNs to distributional shifts in color.

Overall, this research aims to explore the behavior and capabilities of CNNs in the context of color-invariant feature extraction and classification. The findings have implications for understanding the underlying mechanisms of CNNs and can contribute to the development of more robust and adaptable models in various image classification tasks.

2. RELATED WORK

Lately, there have been numerous studies assessing the robustness of deep learning models and their ability to generalize to out-of-distribution (OOD) testing data. Zhang, et al. (2016) [12], demonstrated that Stochastic Gradient Descent (SGD) trained CNNs can memorize the entire training dataset even when the labels are randomized, challenging the assumption that low generalization error corresponds to better understanding rather than memorization. In a theoretical study [13], Kabir et al. showed that the generalization of CNNs can be improved if an optimal background class is used and proposed a methodology for developing background classes.

Geirhos, et al. (2018) [8], Geirhos, et al. (2020) [6], showed that ImageNet-trained CNNs exhibit a bias towards the texture of objects rather than their shapes. They hypothesized that enhancing shape bias could improve CNN performance on out-of-distribution images. This hypothesis was further investigated by Islam, et al. (2021) [14], who studied latent representations to determine the extent to which CNNs learn shape features. They concluded that shape cues are learned in the early epochs and stored in a CNN's deeper layers. In a study [15], it was shown that shape-biased CNNs may perform well on out-of-distribution tasks that align closely with the shape features learned during training but may falter when the out-of-distribution data involves unfamiliar shape configurations or when texture plays a crucial discriminating role.

Mummadi, et al. (2021) [16], explored whether shape bias contributes to robustness against distributional shifts. They found that using different stylized augmentations can improve corruption robustness, although shape bias may be a byproduct rather than a primary factor. Hosseini, et al. (2018) [17], Hosseini, et al (2017) [5], Hosseini, et al (2018) [18], conducted several experiments on MNIST and CIFAR-10 datasets to assess the shape bias property of CNNs. By creating a complemented dataset of MNIST images with negative versions that preserve shape cues but alter background and foreground, they demonstrated that CNNs trained on normal images struggle to detect negative images. Mixing a few negative images with normal ones helped the neural network learn to detect negative images. In a recent study by Li, et al.(2023) [19], it was shown empirically

that enforcing the sparse coding constraint using a non-differential Top-K operation can lead to the emergence of structural encoding in neurons in CNNs, resulting in shape-biased models.

Arjovsky, et al.(2020) [4], introduced Invariant Risk Minimization (IRM), a framework aimed at improving the robustness of models by encouraging them to learn invariant predictors across different environments. This approach addresses the susceptibility of CNNs to spurious correlations present in training data. This issue is further explored by D'Amour, et al. (2020) [20], who revealed CNNs' tendency to learn superficial patterns rather than understanding underlying concepts. Sauer and Geiger (2021) [21], in their work on Counterfactual Generative Networks emphasized the importance of counterfactual reasoning in mitigating biased data.

Data augmentation techniques are generally used to enhance the robustness of deep learning models to account for image variations that arise in practical scenarios.Unlike geometric methods, photometric methods modify the pixel content of the images while preserving their spatial structure according to specified functions resulting in the compositional change in RGB channels. The most commonly used photometric data augmentation approaches include color jittering, color space conversion, and distortion techniques [22]. In [23], Karargyris proposes a color space transformation network, a simple CNN that incorporates a color space matrix to learn useful color space parameters from data and then apply these parameters to the input samples during the training process. Another approach is to use multi-branch networks [24], where each branch performs a specific, pre-defined image transformation operation. The effect of color variability was studied by De, et al. (2021) [25], and found that CNNs are sensitive to color changes to the level of individual blocks. Dataset augmentation methods such as PSNR and Inverse PSNR [26], have been used to improve the robustness of vision models with smaller datasets. Still, this approach requires longer training times and can be effective only in the presence of known transformations. Although some approaches may use data augmentation in connection with this problem, data augmentation was not used here because the intent is to show the effect of the proposed architecture, regardless of the data on which it is applied.

Recent research by Lengyel, et al. (2021) [27], on zero-shot day-night domain adaptation leverages a physics prior to bridge the gap between day and night images, demonstrating a significant advancement in handling domain shifts without requiring labeled data from both domains. This work builds on foundational principles established by Geusebroek, et al. (2001) [28], in their exploration of color invariance, where they introduced techniques to achieve consistent object recognition regardless of varying illumination conditions. Finlayson, et al. (2006) [29], further underscore the importance of preprocessing steps to enhance image quality and reduce artifacts that could impede model performance. Collectively, these studies underscore the need for innovative approaches to enhance the robustness and interpretability of CNNs, moving towards models that are not only intelligent in terms of performance but also resilient to shifts in data distribution.

3. METHODOLOGY

3.1 Dataset Description

To evaluate the generalization ability of CNNs, different custom versions of MNIST and CIFAR10 datasets were used in this paper. MNIST is a large collection of handwritten digits commonly used for testing image classification models. CIFAR10 is a large collection of common real-world objects like automobiles, ships, deer, etc. Both these datasets contain 10 different classes. Each dataset was modified to include variations in color and background to test how well CNNs can adapt to changes in input data that differ from the training scenarios. The customization process for the datasets is described below:

3.1.1 MNIST customization

The MNIST dataset, typically comprised of grayscale images of handwritten digits, has been modified to produce six enhanced versions (illustrated in FIGURE 1. These variations are designed to test the adaptability of CNNs to different color distributions and added visual complexities:

1. *Green Images Dataset:* All images were converted to a green color.
2. *Green Images with Jitter Dataset:* Random background jitter was added to the green images to introduce colored noise.
3. *Multi-Color Images Dataset:* Digits were randomly colored either red, blue, or green.
4. *Multi-Color Images with Jitter Dataset:* Colored jitter was incorporated into the background of the multi-colored images.
5. *RGB Images Dataset:* Images were converted to RGB, allowing digits to appear in combinations of red, green, and blue.
6. *RGB Images with Jitter Dataset:* Colored jitter was added to the RGB images, increasing the complexity.

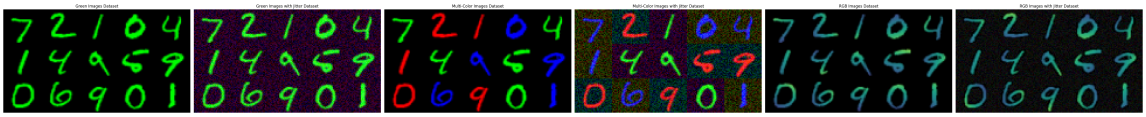

Figure 1: Sample from each dataset variant.

3.1.2 CIFAR-10 dataset customization

The CIFAR-10 dataset, which consists of 32×32 color images of objects from ten different classes such as animals and vehicles, is widely used to assess the robustness and adaptability of CNNs in real-world scenarios. These images closely capture the complexities of real-world visual data. To evaluate the effect of varying lighting conditions that are common in real-world settings, the hue of the CIFAR-10 images was systematically adjusted as described below and shown in FIGURE 2.

1. *Original Images Dataset:* Utilized the unmodified original CIFAR-10 images.
2. *Slight Hue Adjusted Dataset:* The hue of the images was adjusted by a slight increment ($\Delta_{hue} = 0.1$), equivalent to a 36-degree shift on the color wheel.
3. *Moderate Hue Adjusted Dataset:* The hue of the images was adjusted by a moderate increment ($\Delta_{hue} = 0.2$), equivalent to a 72-degree shift on the color wheel.
4. *High Hue Adjusted Dataset:* The hue of the images was adjusted by a larger increment ($\Delta_{hue} = 0.3$), equivalent to a 108-degree shift.

Figure 2: Sample from each variant of CIFAR-10 datasets

3.2 Color Balancing Layer

We introduce a novel color balance layer designed to improve model generalization by adjusting the influence of color channels within input images. The layer redistributes the information across channels and facilitates a more balanced and informative representation of the input data. This layer reduces the redundancy by redistributing information from the dominant channels to the less represented channels, thus enhancing the overall information content of the image.

3.2.1 Mathematical description

The color balance layer implements an information exchange mechanism between the color channels through a linear weighted transformation:

Let c_r, c_g, and c_b denote the red, green, and blue channels of an input image I.

Let w_p be the primary weight and $w_s = \frac{1.0 - w_p}{2}$ be the secondary weight.

The transformation applied to each channel is defined as follows:

- For the red channel c'_r:
$$c'_r = c_r \cdot w_p + (c_g + c_b) \cdot w_s \tag{1}$$
- For the green channel c'_g:
$$c'_g = c_g \cdot w_p + (c_r + c_b) \cdot w_s \tag{2}$$
- For the blue channel c'_b:
$$c'_b = c_b \cdot w_p + (c_r + c_g) \cdot w_s \tag{3}$$

This transformation ensures that each channel contains original and redistributed components from other channels, enhancing the entropy of the image.

The primary weight (w_p) in a color balancing layer plays a critical role in determining the influence of each color channel's original content on the output image as shown in FIGURE 3. Specifically, this weight dictates the proportion of the original channel data that is retained in the transformed output. A higher w_p signifies a stronger retention of the original channel color, thus maintaining the dominance of each channel's inherent characteristics. For example, setting w_p close to 1.0 means that an image's red, green, and blue channels predominantly retain their original intensities, with only a minimal contribution from the other channels. This scenario is beneficial when the intrinsic color distribution is crucial for image interpretation, such as in scenarios requiring high color fidelity to the original scene. Conversely, a lower w_p reduces the dominance of the original channel intensities, thereby increasing the proportion of information obtained from the other channels and lead to a grayscale image. This increased cross-channel interaction can enhance the overall balance and uniformity of color representation in the image, potentially improving model's visual perception where original color biases need to be mitigated. By adjusting w_p, the color balance layer can thus finely control the balance between preserving original channel characteristics and promoting color uniformity, directly impacting the image's visual qualities and its suitability for further processing in tasks like image classification, where color cues might be crucial.

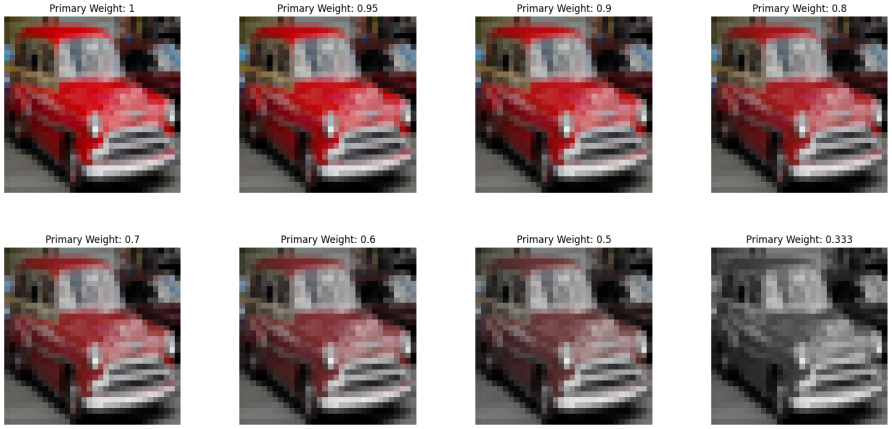

Figure 3: Effect of varying primary weight on the color balance of an image. From top left to bottom right: $w_p = 1, 0.95, 0.9, 0.8, 0.7, 0.6, 0.5, 0.333$.

3.3 Spatial Dropout Layer

The Spatial Dropout Layer is an adaptation of the traditional dropout regularization technique, which is commonly used in training neural networks. Unlike standard dropout, which randomly deactivates individual neurons, Spatial Dropout targets entire feature maps. This approach promotes the development of more robust and spatially invariant features within the network, as it prevents the model from overly depending on specific spatial patterns or locations within the feature maps.

Mathematically, the Spatial Dropout Layer is described by the transformation:

$$x_{\text{dropped}} = x \odot m \tag{4}$$

where $\odot$ denotes element-wise multiplication. The mask m is a binary mask that has the same spatial dimensions as the feature map x. This mask is randomly generated during each training iteration, with each element of the mask being set to zero with a probability p, which controls the rate at which feature maps are dropped, thereby adjusting the layer's regularization effect.

The introduction of Spatial Dropout forces the network to learn redundant representations across various spatial locations, enhancing the model's generalization capabilities and improving its robustness to spatial variations in the input data.

3.4 Network Architecture

3.4.1 For MNIST datasets

The CNN architecture employed in this study is specifically tailored to process MNIST images, drawing inspiration from the VGG network [30] but simplified to suit the relatively straightforward nature of the MNIST dataset. This adaptation involves reducing the number of layers while still adhering to core architectural principles that have proven effective for image recognition tasks.

Convolutional Layers: The network starts with two initial convolutional layers, each equipped with 32 3×3 filters, followed by additional layers that contain 64 filters. These layers are designed to extract low to mid-level features from the images.

Pooling Layers: Max pooling layers are incorporated to reduce the spatial dimensionality of the feature maps and to abstract the features further, reducing the computational load and to prevent overfitting.

Fully Connected Layers: A series of fully connected layers were used followed by a SoftMax activation to get the class probabilities of the input images.

This simplified VGG-inspired architecture effectively balances performance and computational efficiency, making it ideal for tasks involving less complex image datasets like MNIST. The architecture of the network is shown in FIGURE 4.

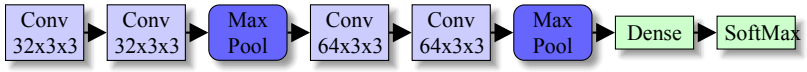

Figure 4: The architecture of simplified VGG16 network for MNIST images

3.4.2 For CIFAR dataset

For the CIFAR dataset, which contains simple real-world image data, the VGG16 architecture was adapted to suit the specific needs of this dataset. The VGG16 architecture, known for its depth and robustness, is particularly well-suited for capturing detailed features necessary for classifying real-world images. The adaptation includes:

Convolutional Blocks: The network consists of multiple blocks of convolutional layers, where each block progressively increases the number of 3×3 convolutional filters from 64 to 512. This setup is designed to enhance detailed feature extraction across the network.

Pooling Layers: Following each convolutional block, a max pooling layer is employed. These layers serve to reduce the spatial dimensions of the feature maps, thereby helping to prevent overfitting and reducing computational complexity.

Fully Connected Layers: The architecture culminates in three dense layers, leading to a final *softmax* activation function. The softmax layer is crucial for classifying the images into one of ten categories, reflecting the class labels of the CIFAR dataset.

This adapted version of VGG16 is optimized for performance with the CIFAR dataset, providing a robust framework for handling its diverse and complex image data. The architecture is shown in FIGURE 5.

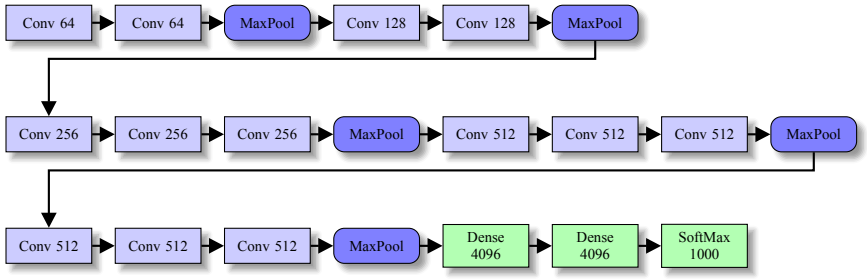

Figure 5: The architecture of VGG16 network used for CIFAR-10 images

3.5 Training and Evaluation

For both MNIST and CIFAR datasets, we train the respective CNN models on the original dataset and evaluate their performance on the color-shifted versions using cross-entropy loss and classification accuracy as the evaluation metric.

We used standard data augmentation techniques including random cropping, horizontal flipping, and translation, during training to improve generalization. The models are trained using Adam optimizer with an initial learning rate of 0.001 and batch size of 128 for 150 epochs.

3.6 Baseline Models

We first establish baseline performance by training the CNN architectures without color balancing and spatial dropout layer on the original MNIST and CIFAR-10 datasets. These baseline models serve as a reference for evaluating the impact of our proposed layers.

3.7 Models With Modified Layers

We then train our models using the proposed Color Balancing Layer and Spatial Dropout Layer after the input layer, on Green MNIST and original CIFAR-10 datasets. We conducted a comparative analysis against baseline models, models incorporating traditional normalization layers, such as Batch Normalization [31] and Layer Normalization [32], and models with depth-wise convolutions instead of normal convolutions. The models are tested on standard and color-shifted versions of the datasets to measure their robustness and generalization capabilities across varied color conditions, aiming to establish the effectiveness of the proposed layers.

4. RESULTS

4.1 Results of Standard Architectures on Modified MNIST and CIFAR10 Datasets

To establish a baseline for model performance, we tested different standard CNN architectures like VGG16, Resnet50 [33], Mobilenetv2 [34]and EfficientNet [35] on modified MNIST and CIFAR10 datasets. The results are shown in TABLE 1 and TABLE 2. The accuracy of the models drops significantly when tested on datasets different from the training distribution.

Table 1: Standard CNNs' performance on various modified MNIST datasets

Model	**Green Images**	**Green Images + Jitter**	**Multi-Color Images**	**Multi-Color Images + Jitter**	**RGB Images**	**RGB Images + Jitter**
VGG16	0.9939	0.9915	0.7881	0.7876	0.9696	0.9666
Resnet50	0.9858	0.8763	0.4917	0.4914	0.6773	0.5645
Mobilenetv2	0.9888	0.4398	0.4525	0.4497	0.4760	0.4542
Efficientnet	0.9856	0.8447	0.4437	0.4368	0.6121	0.5326

In addition, all models, including Vision Transformers (ViT) [36] and ConvNeXt [37], are pre-trained on ImageNet prior to fine-tuning on CIFAR-10 and CIFAR-100. Their performance metrics, reported in TABLE 3 and TABLE 4, provide further insights into the comparative effectiveness of both transformer-based and modern CNN-based architectures on modified data.

Table 2: Standard CNNs' performance on various modified CIFAR10 datasets

Model	Original Images	Slight Hue Adjusted Images	Moderate Hue Adjusted Images	High Hue Adjusted Images
VGG16	0.9258	0.8518	0.7578	0.6879
Resnet50	0.9346	0.8298	0.7572	0.6864
Mobilenetv2	0.9249	0.7945	0.6959	0.5901
Efficientnet	0.9314	0.7996	0.6432	0.6191

Table 3: Performance of Imagenet pre-trained models on CIFAR10

Model	Original Images	Slight Hue Adjusted Images	Moderate Hue Adjusted Images	High Hue Adjusted Images
VGG16	0.942	0.8871	0.7963	0.6531
ResNet50	0.940	0.8577	0.7544	0.7212
MobileNetV2	0.943	0.8336	0.7481	0.6947
EfficientNet	0.9452	0.8394	0.7278	0.6702
ConvNeXtTiny	0.9504	0.7499	0.6573	0.6214
ViT	0.9492	0.6604	0.5342	0.4515

Table 4: Performance of ImageNet pre-trained models fine-tuned on CIFAR-100.

Model	Original Images	Slight Hue Adjusted Images	Moderate Hue Adjusted Images	High Hue Adjusted Images
VGG16	0.7423	0.5420	0.4512	0.4202
ResNet50	0.8209	0.7104	0.6501	0.6197
MobileNetV2	0.7954	0.7233	0.6506	0.6055
EfficientNet	0.8542	0.7666	0.6921	0.6310
ConvNeXtTiny	0.8850	0.6244	0.5205	0.4777
ViT	0.8652	0.5561	0.4570	0.3642

4.2 Results for the MNIST Dataset

TABLE 5 shows the performance of each model on the MNIST dataset. The baselines models and models with traditional normalization layers exhibit a significant drop in performance when evaluated on the color shifted versions of MNIST. By contrast, the model with our proposed layers performed consistently across the color shifted datasets.

Table 5: Model performance on various modified MNIST datasets

Model	**Green Images**	**Green Images + Jitter**	**Multi-Color Images**	**Multi-Color Images + Jitter**	**RGB Images**	**RGB Images + Jitter**
Baseline	0.9972	0.7518	0.4378	0.3079	0.8777	0.8264
Baseline + colored jitter	0.9952	0.9941	0.4077	0.4111	0.9150	0.9378
Layer Normalization	0.9883	0.9784	0.4004	0.3997	0.9178	0.9129
Depthwise Convolution	0.9821	0.9821	0.4037	0.3904	0.6952	0.4439
Color Balancing	0.9821	0.9916	0.4004	0.3990	0.9434	0.8421
Spatial Dropout	0.9947	0.9942	0.4315	0.4311	0.9864	0.9876
Color balancing & Spatial Dropout	**0.9980**	**0.9975**	**0.9745**	**0.9757**	**0.9666**	**0.9652**

4.3 CIFAR Datasets

Similar to the MNIST results, the baseline model shows a significant drop in accuracy when tested on color shifted images of CIFAR-10 and CIFAR-100, with accuracy decreasing by as much as 15% as shown in TABLE 6 and TABLE 7. Whereas, the models using color-balancing and spatial dropout layer performed significantly better.

4.4 Effect of Primary Weight

We investigate the effect of primary weight on the model's performance and generalization. As the primary weight is decreased, the exchange of information between the color channels is increased. This strategy is designed to enhance image representation by integrating more diverse features from each channel. However, it also leads to a noticeable compromise: a reduction in the distinctiveness of the original color information as shown in the FIGURE 3. As the color is an important visual discriminator, a decrease in the model performance is expected. The results are shown in the tables below:

Table 6: Model Performance Across Hue Adjusted Image Sets of CIFAR-10

Model	**Original Images**	**Slight Hue Adjusted Images**	**Moderate Hue Adjusted Images**	**High Hue Adjusted Images**
Baseline	0.9258	0.8518	0.7578	0.6879
Color Balancing	0.9021	0.8256	0.7248	0.6818
Spatial Dropout	0.8130	0.7875	0.7638	0.7595
Color Balancing & Spatial Dropout	**0.9287**	**0.9181**	**0.9077**	**0.8992**

Table 7: Model Performance Across Hue Adjusted Image Sets of CIFAR-100

Model	**Original Images**	**Slight Hue Adjusted Images**	**Moderate Hue Adjusted Images**	**High Hue Adjusted Images**
Baseline	0.7184	0.6588	0.6062	0.5879
Color Balancing	0.7191	0.6462	0.6248	0.5918
Spatial Dropout	0.6830	0.6675	0.6438	0.6295
Color Balancing & Spatial Dropout	**0.7192**	**0.7084**	**0.6967**	**0.6793**

Table 8: Model performance on modified MNIST datasets for different primary weights

Primary Weight	**Green Images**	**Green Images + Jitter**	**Multi-Color Images**	**Multi-Color Images + Jitter**	**RGB Images**	**RGB Images + Jitter**
0.95	0.9907	0.9918	0.9693	0.9543	0.9769	0.9777
0.9	0.9980	0.9975	0.9702	0.9709	0.9666	0.9652
0.8	0.9981	0.9980	0.9802	0.9809	0.9886	0.9884
0.7	0.9973	0.9982	0.9890	0.9872	0.9893	0.9898
0.6	0.9976	0.9985	0.9970	0.9914	0.9924	0.9915
0.5	**0.9979**	**0.9918**	**0.9851**	**0.9913**	**0.9955**	**0.9935**

Table 9: Model performance on Hue adjusted images from CIFAR 10 for different primary weights

Primary Weight	Original Images	Slight Hue Adjusted Images	Moderate Hue Adjusted Images	High Hue Adjusted Images
0.95	0.9161	0.9031	0.8859	0.8892
0.9	**0.9258**	**0.9181**	**0.9077**	**0.8989**
0.8	0.9086	0.8924	0.8859	0.8892
0.7	0.9073	0.8899	0.8800	0.8761
0.6	0.9033	0.8863	0.8712	0.8861
0.5	0.8872	0.8793	0.8703	0.8724

From the tables above, it can observed that, decreasing the primary weight improved performance on MNIST datasets, even though the network became invariant to color, as shown in TABLE 8. However, model performance increased only until the primary weight was reduced to 0.9, it then deteriorated as the primary weight was further decreased on hue-adjusted CIFAR-10 images as shown in TABLE 9. Therefore, a lower primary weight is suitable for conditions where color does not play a critical role, whereas a higher primary weight should be maintained in situations where color is important for image classification.

4.5 DISCUSSION

Our experiments on both the MNIST and CIFAR-10 datasets demonstrate that CNNs trained on a specific color distribution struggle to generalize to color-shifted images, even when traditional normalization layers like Batch Normalization and Layer Normalization are employed. The proposed Color Balancing Layer and Spatial Dropout Layer effectively address this issue by enhancing the network's robustness to color variations and spatial dependencies, respectively. The Color Balancing Layer balances the color channels, reducing the impact of color shifts, while the Spatial Dropout Layer encourages the network to learn more robust and spatially invariant representations. The combination of these two layers yields the best performance, achieving significant accuracy improvements on the color-shifted versions of both datasets. These results highlight the importance of incorporating specialized architectural components to improve the generalization ability of CNNs on color shifted images.

While our proposed layers have shown promising results, there are potential limitations and future research directions. The effectiveness of these layers may depend on the specific dataset and color-shift characteristics. Further evaluation on a wider range of datasets and color-shift scenarios is necessary to assess their robustness.

Finally, exploring the integration of our proposed layers with other generalization techniques, such as data augmentation or adversarial training, could potentially lead to further performance improvements.

5. CONCLUSION

We addressed the challenge of improving the generalization ability of CNNs color shifted images. We proposed a novel approach by using Color Balancing layer and Spatial Dropout layer. Spatial Dropout effectively increases the robustness of the network by encouraging the model to learn more diverse and representative features that are not overly dependent on specific channels in the input image. The Color Balancing and Spatial Dropout layer balances the discriminative information in the color channels of the input image, reducing the impact of color shifts on the network's performance. We evaluated the effectiveness of our proposed layers on the modified MNIST, CIFAR-10 and CIFAR 100 datasets. Our experiments demonstrated that incorporating the Color Balancing and Spatial Dropout layer significantly improves the generalization ability of CNNs on color shifted images, outperforming baseline models.

References

[1] Krizhevsky A, Sutskever I, Hinton GE. Imagenet Classification With Deep Convolutional Neural Networks. Neural Inf Process Syst. 2012;25.

[2] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, et al. Attention Is All You Need. Adv Neural Inf Process Syst. 2017;13.

[3] Belkin M, Hsu D, Ma S, Mandal S. Reconciling Modern Machine-Learning and the Bias-Variance Trade-off. Proc Natl Acad Sci U S A. 2019;116:15849-15854.

[4] Arjovsky M, Bottou L, Gulrajani I, Lopez-Paz D. Invariant Risk Minimization. 2020. Arxiv preprint. https://arxiv.org/pdf/1907.02893

[5] Hosseini H, Xiao B, Jaiswal M, Poovendran R. On the Limitation of Convolutional Neural Networks in Recognizing Negative Images. 2017. Arxiv preprint: https://arxiv.org/pdf/1703.06857

[6] Geirhos R, Jacobsen JH, Michaelis C, Zemel R, Brendel W, et al. Shortcut Learning in Deep Neural Networks. Nat Mach Intell. 2020;2:665-673.

[7] Ritter S, Barrett DG, Santoro A, Botvinick MM. Cognitive Psychology for Deep Neural Networks: A Shape Bias Case Study. In: International conference on machine learning. 2017:2940-2949.

[8] Geirhos R, Rubisch P, Michaelis C, Bethge M, Wichmann FA, et.al. Image Net Trained Cnns Are Biased Towards Texture, Increasing Shape Bias Improves Accuracy and Robustness. 2018. Arxiv preprint: https://arxiv.org/pdf/1811.12231

[9] https://www.lri.fr/ marc/Master2/MNIST_doc.pdf

[10] https://www.cs.toronto.edu/ kriz/cifar.html.

[11] Tompson J, Goroshin R, Jain A, LeCun Y, Bregler C. Efficient Object Localization Using Convolutional Networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition CVPR. New York. IEEE. 2015:648-656.

[12] Zhang C, Bengio S, Hardt M, Recht B, Vinyals O. Understanding Deep Learning Requires Rethinking Generalization. Commun ACM. 2016. Arxiv Preprint: https://arxiv.org/pdf/1611.03530

[13] Kabir HM. Reduction of Class Activation Uncertainty With Background Information. 2023. Arxiv preprint: https://arxiv.org/pdf/2305.03238

[14] Islam MA, Kowal M, Esser P, Jia S, Ommer B, et al. Shape or Texture: Understanding Discriminative Features in CNNs. 2021. Arxiv preprint. https://arxiv.org/pdf/2101.11604

[15] Qiu X, Kan M, Zhou Y, Bi Y, Shan S. Shape-Biased Cnns Are Not Always Superior in Out-Of-Distribution Robustness. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision. New York: IEEE. 2024:2315-2324.

[16] Mummadi CK, Subramaniam R, Hutmacher R, Vitay J, Fischer V et al. Does Enhanced Shape Bias Improve Neural Network Robustness to Common Corruptions? 2021. Arxiv preprint: https://arxiv.org/pdf/2104.09789

[17] Hosseini H, Xiao B, Jaiswal M, Poovendran R. Assessing Shape Bias Property of Convolutional Neural Networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops. 2018:1923-1931. Arxiv Preprint: https://arxiv.org/abs/1803.07739.

[18] Hosseini H, Poovendran R. Semantic Adversarial Examples. 2018. Arxiv Preprint: https://arxiv.org/pdf/1804.00499

[19] Li T, Wen Z, Li Y, Lee TS. Emergence of Shape Bias in Convolutional Neural Networks Through Activation Sparsity. Neuro IPS. 2023;36:71755-71766.

[20] D'Amour A, Heller KA, Moldovan DI, Adlam B, Alipanahi B, et al. Under Specification Presents Challenges for Credibility in Modern Machine Learning. 2020. Arxiv preprint. https://arxiv.org/pdf/2011.03395

[21] Sauer A, Geiger A. Counterfactual Generative Networks. 2021. Arxiv preprint: https://arxiv.org/pdf/2101.06046

[22] Mumuni A, Mumuni F. Data Augmentation: A Comprehensive Survey of Modern Approaches. Array. 2022;16:100258.

[23] Karargyris A. Color Space Transformation Network. 2015. Arxiv preprint: https://arxiv.org/pdf/1511.01064

[24] Mumuni A, Mumuni F. CNN Architectures for Geometric Transformation-Invariant Feature Representation in Computer Vision a Review. SN COMPUT. SCI. 2021;2:340.

[25] De K, Pedersen M. Impact of Colour on Robustness of Deep Neural Networks. IEEE/CVF International Conference on Computer Vision Workshops (ICCVW). 2021:21-30.

[26] Kim EK, Lee H, Kim JY, Kim S. Data Augmentation Method by Applying Color Perturbation of Inverse PSNR and Geometric Transformations for Object Recognition Based on Deep Learning. Appl Sci. 2020;10:3755.

[27] Lengyel A, Garg S, Milford M, Van Gemert JC. Zero-Shot Day-Night Domain Adaptation With a Physics Prior. In: Proceedings of the IEEE/CVF international conference on computer vision. 2021:4399-4409.

[28] Geusebroek JM, Van den Boomgaard R, Smeulders AW, Geerts H. Color Invariance. IEEE Trans Pattern Anal Mach Intell. 2001;23:1338-1350.

[29] Finlayson GD, Hordley SD, Lu C, Drew MS. On the Removal of Shadows From Images. IEEE Trans Pattern Anal Mach Intell. 2006/2005;28:59-68.

[30] Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. ICLR. 2015. Arxiv Preprint: https://arxiv.org/pdf/1409.1556

[31] Ioffe S, Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. 2015. Arxiv preprint: https://arxiv.org/pdf/1502.03167

[32] Ba JL, Kiros JR, Hinton GE. Layer Normalization. 2016. Arxiv preprint: https://arxiv.org/pdf/1607.06450

[33] He K, Zhang X, Ren S, Sun J. Deep Residual Learning for Image Recognition. Rt-PA of the IEEE conference on computer vision and pattern recognition. 2016:770-778.

[34] Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC. MOBILENETV2: Inverted Residuals and Linear Bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018:4510-4520.

[35] Tan M, Le Q. Efficientnet: Rethinking Model Scaling for Convolutional Neural Networks. In: International conference on machine learning. 2019:6105-6114.

[36] Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, et al. An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale. In: International Conference on Learning Representations. 2021.Arxiv Preprint: https://arxiv.org/pdf/2010.11929

[37] Liu Z, Mao H, Wu CY, Feichtenhofer C, Darrell T, et al. A Convnet for the 2020s. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition(CVPR). New York: IEEE. 2022:11976-11986.

6. APPENDIX

In the appendix, we describe the algorithms used to create the color-shifted versions of MNIST data. These algorithms apply color transformations to the original grayscale images to simulate various color processing scenarios, such as changing image hues or adding color jitters. The methods are implemented to test the robustness of models against color variations and to enhance their ability to generalize across different color distributions.

6.1 Convert Images to Green-Channel Only

Algorithm 1 Convert Images to Green-Channel Only

procedure RvgvbConv($image\_array, n\_green\_only$)
    $choice\_list \leftarrow [0, 1, 2]$ ▷ Color channel indices: 0 = Red, 1 = Green, 2 = Blue
    $l \leftarrow []$ ▷ Initialize list to store images
    for $i \in$ range(len($image\_array$)) **do**
        $template \leftarrow$ zeros($(32, 32, 3)$, dtype = trainX.dtype)
        $image \leftarrow image\_array[i, :, :]$
        if n_green_only **then**
            $n \leftarrow$ random.choice($choice\_list$)
        else
            $n \leftarrow 1$ ▷ Always use green channel
        end if
        $template[:, :, n] \leftarrow image$ ▷ Assign image to chosen channel
        l.append($template$)
    end for
    return np.array(l)
end procedure

6.2 Algorithm for Random Color Jitter

Algorithm 2 Apply Random Color Jitter to Images

procedure ApplyJitter($image\_array$, $strength$)
  for $img \in image\_array$ **do**
    $jitter \leftarrow$ random values within $[-strength, strength]$
    $img \leftarrow img + jitter$ ▷ Add jitter to each pixel
    $img \leftarrow \text{clip}(img, 0, 255)$ ▷ Ensure pixel values are valid
  end for
  return $image\_array$
end procedure

6.3 Algorithm for Multi-Color Images with Color Jitter

Algorithm 3 Multi-Color Images with Color Jitter

procedure rvgvb_with_jitter(*image_array, n_green_only)*
  $choice\_list \leftarrow [0, 1, 2]$ ▷ *List of color channel indices*
  $l \leftarrow []$ ▷ *Initialize list to hold processed images*
  for $i \in range(len(image\_array))$ ***do***
    $jitter\_mask \leftarrow np.random.rand(32, 32, 3) \times 75$
    $template \leftarrow jitter\_mask.astype(np.uint8)$
    $image \leftarrow image\_array[i, :, :]$
    if n_green_only ***then***
      $n \leftarrow np.random.choice(choice\_list)$ ▷ *Choose a channel index randomly*
    else
      $n \leftarrow 1$ ▷ *Always use green channel if not random*
    end if
    $template[:, :, n] \leftarrow image$ ▷ *Assign image to chosen channel*
    $l.append(template)$
  end for
  return $np.array(l)$ ▷ *Return the processed image array*
end procedure

6.4 Algorithm to Apply Dynamic Color Palette to MNIST Images

The function *apply_dynamic_palette_to_batch* enhances a batch of MNIST images by applying a dynamic color gradient to each image, which is normally in grayscale. This is achieved by first normalizing the pixel values to the range [0,1]. The function then calculates a gradient direction based on a randomly chosen angle for each image, using this direction to determine the intensity of

the color at each pixel based on its distance from the center of the image. If enabled via a boolean flag, the function can also introduce a subtle random noise (jitter) to the background, thus adding visual complexity. The final output is a batch of RGB images, where the originally visible parts of the MNIST digits are now represented with a colorful gradient overlay, and the background remains either black or slightly jittered.

Algorithm 4 Apply Dynamic Color Palette to MNIST Images

procedure ApplyDynamicPalette($images$, $jitter$)
    rgb_images $\leftarrow$ zeros(shape = (len($images$), 32, 32, 3), dtype = $np.float32$)
    for idx, image $\in$ enumerate($images$) **do**
        norm_image $\leftarrow$ image.astype($np.float32$)$/255.0$
        angle $\leftarrow$ np.random.uniform$(0, 2 \times \pi)$
        direction $\leftarrow$ np.array([np.cos($angle$), np.sin($angle$)])
        center $\leftarrow$ np.array$([16, 16])$
        for $i \in$ range(32) **do**
            for $j \in$ range(32) **do**
                if norm_image$[i, j] > 0$ **then**
                    pos $\leftarrow$ np.array$([i, j])$
                    distance $\leftarrow$ np.dot(pos – center, direction)
                    max_dist $\leftarrow \sqrt{2} \times 16$
                    normalized_distance $\leftarrow$ (distance + max_dist)$/(2 \times$ max_dist)
                    color $\leftarrow$ plt.cm.viridis(normalized_distance)$[:3]$
                    rgb_images[idx, i, j, :] $\leftarrow$ color $\times$ norm_image$[i, j]$
                else if $jitter$ **then**
                    rgb_images[idx, i, j, :] $\leftarrow$ np.random.rand$(3) \times 0.1$
                end if
            end for
        end for
    end for
    return rgb_images
end procedure

7. Algorithm for Custom Color Balance Layer

Algorithm 5 Custom Color Balance Layer

Input: primary_weight
Output: balanced color channels
procedure CustomColorBalance($primary\_weight$)
    Initialize *primary_weight*
    Compute *secondary_weight* $\leftarrow (1.0 - primary\_weight)/2.0$
    Split input tensor into three color channels: *red*, *green*, *blue*
    new_red $\leftarrow$ *red* $\cdot$ *primary_weight* + *green* $\cdot$ *secondary_weight* + *blue* $\cdot$ *secondary_weight*
    new_green $\leftarrow$ *red* $\cdot$ *secondary_weight* + *green* $\cdot$ *primary_weight* + *blue* $\cdot$ *secondary_weight*
    new_blue $\leftarrow$ *red* $\cdot$ *secondary_weight* + *green* $\cdot$ *secondary_weight* + *blue* $\cdot$ *primary_weight*
    Stack *new_red*, *new_green*, *new_blue* back together
    return *output*
end procedure

procedure get_config
    config $\leftarrow$ super.get_config()
    config.update({primary_weight: primary_weight})
    return *config*
end procedure