Bias Detection and Mitigation in Convolutional Neural Network Image Classification

Leanne Beltman, Seth Bonin, Justin Coffey, Connar Gibbon, and Libby Stephan

College of Arts and Science, Washington State University

DATA 424: Data Analytics Capstone

Dr. Sergey Lapin

April 20, 2025

Table of Contents

1. Abstract

Convolutional neural networks (CNNs) frequently learn spurious correlations in their training data, resulting in unequal performance across classes and demographic groups. In this work, we introduce three lightweight, model-agnostic metrics to detect bias: interclass standard-deviation checks, maximum-to-minimum accuracy ratios, and training-versus-validation performance gaps. We evaluate these metrics on the Biased MNIST dataset, which applies controlled correlations between digit labels and distractor features (color, position, texture) at correlation levels from 0.1 to 0.9. Our experiments show that the first two metrics effectively capture class-imbalance issues and variance, while the third metric consistently identifies overfitting to spurious cues when an unbiased validation set is available. To reduce bias, we propose a mixed-bias training protocol that combines equal parts low-correlation and high-correlation data, yielding up to 44 percent relative improvement in balanced test accuracy compared to training solely on biased data. These results provide practitioners with practical tools for both detecting and mitigating distribution-induced bias in CNN classifiers.

2. Introduction

Deep CNNs have revolutionized image classification, powering applications from medical diagnosis to autonomous driving. Unfortunately however, these models often learn and amplify biases present in their training data, resulting in uneven performance across different classes or demographic groups. In domains where safety is the primary concern—such as tumor detection or facial recognition—such biased behavior can perpetuate unfair outcomes or even endanger lives.

Bias in data arises in many forms: class imbalance, spurious correlations between target labels and background features, or the intentional exclusion or overrepresentation of certain subpopulations. Left unchecked, these biases enable CNNs to shortcut cues: using background color or texture, instead of learning the true distinguishing features of each class. Consequently, a model may achieve high overall accuracy yet perform catastrophically on underrepresented or out of distribution samples. In this capstone project, we systematically investigate how to detect and mitigate distribution - induced bias in CNN image classifiers. Using the Biased MNIST dataset—which injects controlled correlations between digit class and distractor attributes (color, position, texture)—we first quantify bias via three complementary metrics: (1) unusually large interclass variance, (2) extreme class accuracy gaps, and (3) training–validation accuracy discrepancies. We then explore a simple mitigation strategy, a mixed - bias training, that blends unbiased and biased subsets to guide the model toward more generalizable feature representations.

Our central research question is: When presented with datasets exhibiting low versus high correlation between digit labels and distractor features, how effectively can we detect bias in a CNN, and to what extent can mixed‑bias training recover balanced performance across all classes? By answering this question, we aim to provide practitioners with both practical detection tools and a lightweight mitigation recipe for combating bias in real world CNN deployments.

3. Literature Review

Before we began creating and implementing a model, a literature review surrounding bias and bias mitigation techniques was crucial. We first needed to understand the different types of bias to be able to more easily identify them as we began to analyze the initial outputs of our model. There are four types of bias we initially sought to define and understand:

- *Distribution bias* occurs when the dataset used to train a model does not represent the total true population. This can occur when certain subgroups of the population are underrepresented, such as the underperformance of a model with females when the training data is majority male. The model learns to perform better for the majority group of the population, which leads to worse performance for the minority groups in the data.
- *Hardness bias* happens when there are examples of difficult, or "hard," examples to classify. Due to this classification difficulty, the model may generalize too much, leading to too much attention being shifted to these examples and, therefore, an underperforming model.
- *Explicit bias* is a result of purposeful selection of data to intentionally discriminate or favor certain outcomes. This bias is essentially hardcoded into the data and, thus, into the model. The model produced may reinforce this explicit bias, which can lead to a cycle of intentional manipulation of the data.
- *Implicit bias* tends to be harder to detect. This is a bias that unknowingly influences either the data collection, sampling, or both by those performing these tasks. Similar to explicit bias, the bias can lead to model results that enforce certain stereotypes or prejudices.

Now that the types of bias are identified, there are several common causes for bias in data, which lead to biased results from models. Class imbalance usually leads to distribution bias since an overrepresentation of a particular group can cause a model to prioritize said group in classification. Common patterns resulting from the majority group may cause a model to ignore less prevalent patterns of minority groups to achieve a smaller classification error. Of course, prejudices or simply human error, whether in sampling or creating measurement criteria, can introduce bias in the data and the model.

In the context of the Biased MNIST dataset, we can expect certain biases in our model thanks to purposeful distractors that have been added. For instance, imbalanced frequencies of certain digits can lead to distribution bias. Certain patterns, such as a specific digit being majority a certain color, can have a profound impact on a model's performance. Other added distractors, like added letters and manipulated digits (color, position, scale, etc.), can introduce bias as the model trains to identify those distractors rather than the digits themselves. There are more than enough possibilities that can end up introducing unwanted bias into our model.

Given these potential biases, we can introduce different methods in an attempt to decrease model bias and increase performance. Class balancing can ensure distribution bias is less likely in the model by manipulating the amount of samples in each class (number of digits). Data augmentation is another method that ensures any transformations, such as rotations and scaling, is uniform across all observations. We can also focus on the accuracy of individual digit classification rather than overall accuracy - limiting the influence outside distractors have on model performance.

Class balancing techniques, like Synthetic Minority Oversampling Technique (SMOTE), can help avoid distribution bias in our model. SMOTE utilizes the process of oversampling to populate the minority class in the data to match the size of the majority class. However, instead of simply duplicating samples, SMOTE instead generates synthetic samples to accomplish this. "In short, the process to generate the synthetic samples are as follows.

1. Choose random data from the minority class.
2. Calculate the Euclidean distance between the random data and its k nearest neighbors.
3. Multiply the difference with a random number between 0 and 1, then add the result to the minority class as a synthetic sample.
4. Repeat the procedure until the desired proportion of minority class is met.

This method is effective because the synthetic data that are generated are relatively close with the feature space on the minority class, thus adding new "information" on the data, unlike the original oversampling method" (Viadinugroho, 2021). There are demonstrated examples of SMOTE increasing model accuracy, such as in *Enhancing Transfer Learning for Medical Image Classification with SMOTE: A Comparative Study* by Alam et al. (2024), where they state that "performance in diabetic retinopathy detection is hindered by class imbalance. To mitigate this, we integrate the Synthetic Minority Over-sampling Technique (SMOTE) with TL and traditional machine learning(ML) methods, which improves accuracy by 1.97%, recall (sensitivity) by 5.43%, and specificity by 0.72%" (Alam et al., 2024).

In addition to SMOTE, the Tomek Links method can be used as an undersampling technique. "This method can be used to find desired samples of data from the majority class that is having the lowest Euclidean distance with the minority class data (i.e. the data from the majority class that is closest with the minority class data, thus make it ambiguous to distinct), and then remove it" (Viadinugroho, 2021). These two methods working in tandem provide a powerful class balancing technique to combat potential bias.

Lastly, spurious correlations, or relationships between two variables that appear to be connected but are not actually causally related, can lead to implicit bias or even perhaps hardness bias in our model. This can happen as a result of our model recognizing patterns when identifying digits; colored backgrounds and other distractors can lead to patterns that add bias to the model. According to Taghanaki et al. (2022), identifying these spurious correlations and applying mitigation measures, like MaskTune, can increase model performance and accuracy, since test data may not share the same patterns as a set of training data with spurious correlations.

MaskTune identifies the features the model is relying heavily on for classification, distractors for example, and applies a "mask" to hide these features to force the model to take a more generalized approach to the samples. "MaskTune generates a diverse set of partially masked training examples, forcing the model to investigate a wider area of the input features landscape e.g., new pixels" (Taghanaki et al., 2022, pg. 2). For example, take a dataset that has different landscapes and animals in images - "if cows only appear on grass in the training set then it is unclear which of the grass or the cow refers to the "cow" label" (Taghanaki et al., 2022, pg. 2). If another animal appears in grass in the test set or a cow appears on another surface, the model will perform poorly. With these bias definitions and mitigation techniques in mind, we move forward with developing our model.

4. Methodology

4.1 Biased MNIST Dataset

The dataset we will use derives from the well-known MNIST dataset which contains images of handwritten digits in black and white with the goal to classify each digit. The Biased MNIST dataset alters these images to include various distractors in the image to purposely introduce noise. These additional features include digit position, digit color, digit scale, letter (which may appear elsewhere in the image), letter color, texture (an overlayed pattern), and texture color.

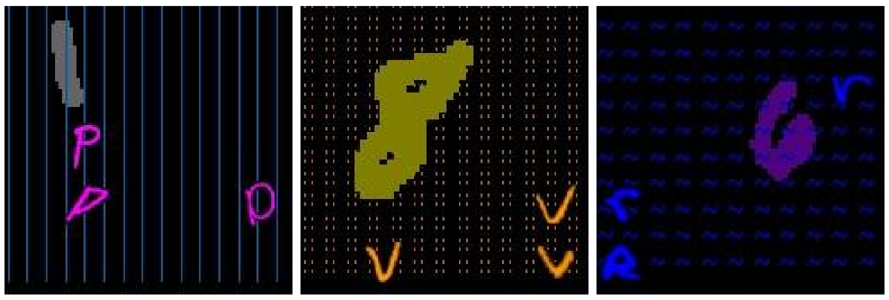
Figure 1: example images from Biased MNIST

The frequency of these features in each digit class depends on the level of bias/correlation for each folder Biased MNIST provides. The correlation levels of this dataset spans 0.1, 0.5, 0.75, 0.9, 0.95, and 0.99, where 0.1 has a more even distribution of the additional features per class and 0.99 has the most imbalanced distribution per class. Predefined test subsets are given in the "full" folder and the "test" folder.

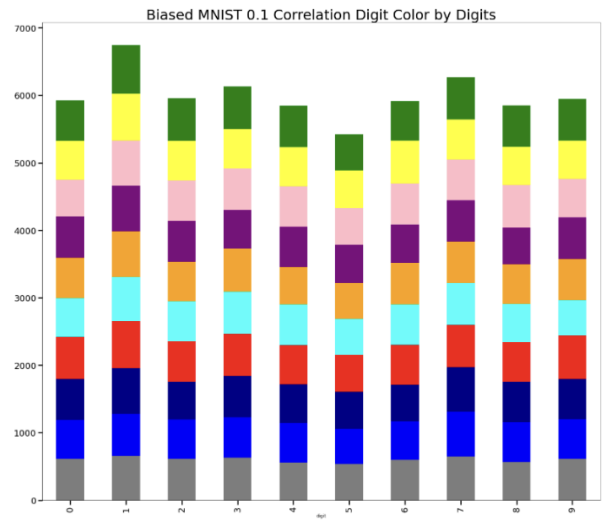

Figure 2

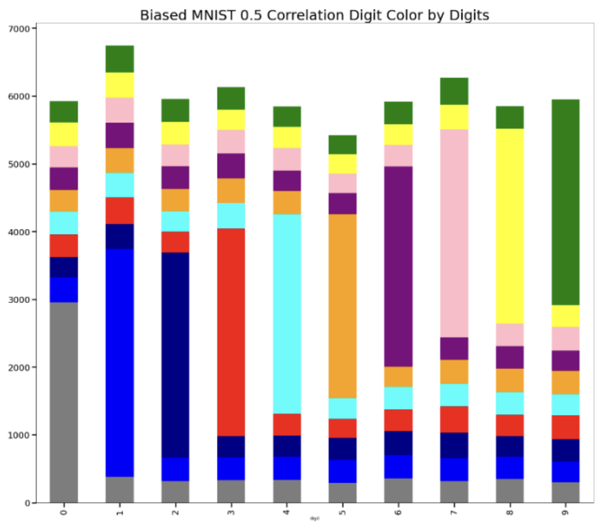

Figure 3

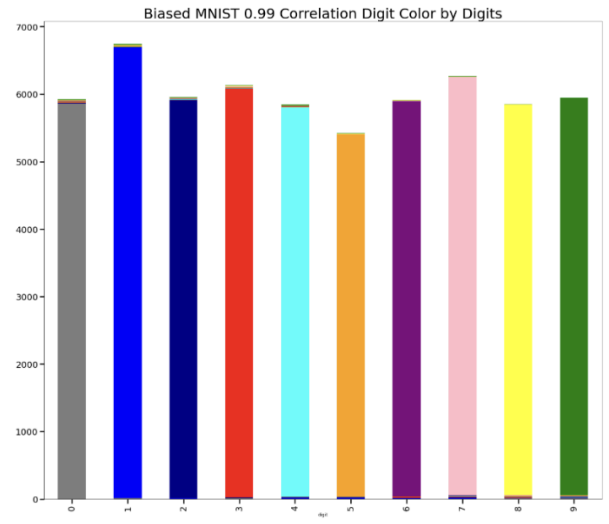

Figure 4

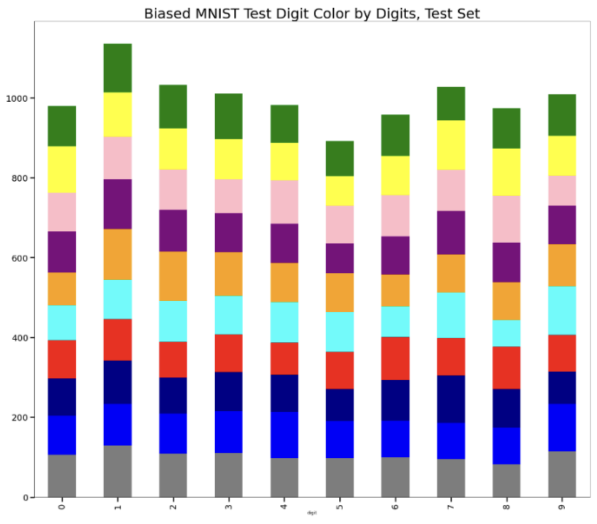

Figure 5

Figures 2-5: distribution of digit color by correlation levels and test set

4.2 CNN Version 1

To evaluate the impact of dataset bias on classification performance, we implemented a lightweight convolutional neural network (SimpleCNN) using PyTorch. The goal of this model was not to achieve high accuracy, but to provide a consistent baseline for evaluating bias detection and mitigation strategies across datasets with different levels of correlation.

The SimpleCNN model is composed of two convolutional layers and a single fully connected output layer. The first convolutional layer accepts RGB images of size 160x160 and transforms them from 3 input channels to 16 channels using a 3x3 kernel, stride of 2, and padding of 1. The second convolutional layer increases the channel count from 16 to 32 using the same kernel and stride settings. Each convolutional layer is followed by a ReLU activation and 2x2 max pooling, reducing the spatial dimensions. The feature maps are flattened into a 3200-dimensional vector (32 x 10 x 10), which is passed into a fully connected layer that outputs logits for each of the 10 digit classes.

The model trained using a cross-entropy loss function, which is appropriate for multi-class classification tasks, and optimized with the Adam optimizer at a learning rate of 0.001. We used a batch size of 32 and trained for 3 epochs during initial subset testing. Training and test images were normalized using per-channel means and standard deviations computed from random 1,000 image samples. Each run used a 10% training subset (6,000 images), with evaluation on a 20% test subset.

Bias Level	Training Accuracy	Test Accuracy	Train-Test Gap	Epochs
0.1	43.53%	37.55%	5.98%	3
0.5	67.63%	18.00%	49.63%	3
0.75	90.67%	15.20%	75.47%	3
0.9	98.00%	12.45%	85.55%	3

Table 1: CNN version 1 results

These initial results align with expectations for models trained on increasingly biased datasets. As the correlation levels increase, the model learns to rely on shortcut features introduced by the bias, such as color or texture, rather than the true underlying digit structures. This leads to rapid increases in training accuracy, but poor generalization to the balanced test set.

4.3 CNN Version 2

While the SimpleCNN model was able to show the increased reliance on shortcut features to predict digit class, we want to achieve a higher test accuracy to provide a point of comparison for our bias detection metrics. Therefore, a second version of the CNN was built upon the foundation of SimpleCNN with the goal to increase test accuracy.

In this newer model, several additions were made. Firstly, a third convolutional layer was added in hopes that more complex features could be learned beyond the first two layers. This is followed by an adaptive average pooling layer which standardizes any input to be 5x5. Then, a dropout layer is added to randomly set 50% of features to 0 during training; this is to reduce overfitting. Lastly, a learning rate scheduler is added to decrease learning rate by 0.5 every 10 epochs to speed up learning early in the beginning, and the number of epochs was increased to 50 to allow enough time for the model to learn.

4.4 Model Performance Per Group

There are four groups of models based on the training and testing data we would like to use to evaluate and tune our bias metics. To ensure model results are not due to chance, four instances of each type of model using CNN version 2 are trained and tested to gather a small sample of accuracies for a total of 16 model instantiations. The four groups of models are as follows:

1. Training sampled from the first half of 0.1; test sampled from the second half of 0.1
    a. (9900 train - 990 test)
    b. Simulates training and testing on unbiased data from same distribution
2. Training sampled from the first half of 0.9; test sampled from the second half of 0.9
    a. (9900 train - 990 test)
    b. Simulates training and testing on biased data from same distribution
3. Training sampled from all of 0.1; test sampled from "full/test"
    a. (9960 train - 1000 test)
    b. Simulates training on unbiased data and testing on unbiased data not necessarily from same distribution
4. Training sampled from all of 0.9; test sampled from "full/test"
    a. (9960 train - 1000 test)
    b. Simulates training on biased data and testing on unbiased data not necessarily from same distribution

Group 1	Mean Train Accuracies	Mean Test Accuracies
Overall	99.63%	70.70%
Class 0	99.66%	75.96%
Class 1	99.89%	82.12%
Class 2	99.70%	63.18%

Group 2	Mean Train Accuracies	Mean Test Accuracies
Overall	100%	97.58%
Class 0	100%	95.26%
Class 1	100%	96.48%
Class 2	100%	97.42%

Class 3	99.73%	69.22%
Class 4	99.48%	73.12%
Class 5	99.50%	65.32%
Class 6	99.78%	80.67%
Class 7	99.80%	72.43%
Class 8	99.27%	55.19%
Class 9	99.80%	67.91%

Table 2: group 1 average performance

Class 3	99.97%	96.41%
Class 4	100%	98.70%
Class 5	100%	96.81%
Class 6	100%	97.75%
Class 7	100%	98.65%
Class 8	100%	98.89%
Class 9	100%	99.21%

Table 3: group 2 average performance

Group 3	Mean Train Accuracies	Mean Test Accuracies
Overall	99.59%	74.58%
Class 0	99.55%	75.03%
Class 1	99.84%	87.07%
Class 2	99.78%	74.24%
Class 3	99.46%	75.12%
Class 4	99.48%	75.8%
Class 5	99.25%	68.73%
Class 6	99.90%	81.63%
Class 7	99.76%	75.27%
Class 8	99.31%	63.10%
Class 9	99.52%	67.51%

Table 4: group 3 average performance

Group 4	Mean Train Accuracies	Mean Test Accuracies
Overall	100%	15.95%
Class 0	100%	17.21%
Class 1	100%	13.41%
Class 2	100%	11.74%
Class 3	100%	14.39%
Class 4	100%	15.66%
Class 5	100%	13.68%
Class 6	100%	21.74%
Class 7	100%	21.85%
Class 8	100%	17.33%
Class 9	100%	13.63%

Table 5: group 4 average performance

Group 1, which trains and tests on unbiased data from the same distribution, has a mediocre performance of 70.70% overall. The highest class wise accuracy falls upon digit 1 (Class 1) with 82.12% while the lowest class wise accuracy is accredited to digit 8 with 55.19%. Group 2 was trained on highly biased data and tested on highly biased data. The overall test accuracy and the class wise accuracies are all extremely high being over 95%. These high accuracies are not unexpected as the high bias in the training and testing data gives the model something easy to learn to predict upon. Group 3 was trained and tested on unbiased data, but not necessarily from the same distribution. Group 3's performance is similar to group 1's performance with an overall test accuracy of 74.58%; similarly, the best and worst class wise accuracies are digit 1 and digit 8 with accuracies 87.07% and 63.10% respectively. Group 4 has the worst performance with an overall test accuracy of 15.95%, which is to be expected since this model likely over relies on biased attributes to make its predictions.

4.5 Bias Detection Metrics

We developed three metrics to identify potential bias indicators from our model results.

The first metric takes an array of class accuracies from the validation data (in our case, this is an array that contains 10 class accuracies - one for each digit from 0-9). First, the mean validation accuracy and standard deviation are calculated from the array of class accuracies. Then, the value of each class accuracy is checked using a for-loop to determine if a given class accuracy is three or more standard deviations away from the mean. When the underlying distribution of validation accuracies is approximately normal and the distribution parameters are true, there is a 98% chance that a given class accuracy will be within three standard deviations of the mean in either direction. If a class accuracy falls at or outside of three standard deviations from the mean, this is an unlikely result and the metric would then return true, indicating that the model appears to be biased. The likelihood of having a class accuracy so far from the mean would be very low in a model that is theoretically able to predict outcomes of each class with equal accuracy. This would likely indicate the presence of bias in some form that is impairing or dramatically improving the model's ability to accurately predict instances of a certain class compared to the remaining classes.

The standard deviation approach works well when the variation between classes (and thus the standard deviation) is relatively small. When the variation between the classes is much larger, it may become difficult to observe points that are three standard deviations away from the mean because the standard deviation becomes so large. We created a second bias metric to address this limitation and identify potential bias where the first metric may fail. The second metric also takes in an array of validation class accuracies like the standard deviation metric. However, rather than looking for accuracies that are more than three standard deviations away from the mean, this metric subtracts the minimum class accuracy from the maximum class accuracy and indicates bias if the difference is greater than 30%. In a situation where the difference between the best and worst class is 30% or greater, the standard deviation metric would likely struggle because the large standard deviation makes it unlikely that points will reside three or more standard deviations away from the mean. This metric works well in cases of high variance between classes, which is also an indicator of potential bias. If a model has large fluctuations across its class accuracies, this is cause for concern and likely indicates the presence of bias or hardness in the data that has not been accounted for.

The third and final metric aims to identify gaps between training and validation results, which targets overfitting and poor generalization. Models that are trained on highly correlated data may be unable to generalize when introduced to unbiased data that does not contain the original correlations from the training set. For example, if a model was trained exclusively on images of people with brown hair also having brown eyes, it may falsely associate having brown hair means that the person must also have brown eyes or vice versa. If this model was introduced to an image of a person with brown eyes but *blonde* hair, it may falsely classify the person as having brown hair because the model has incorrectly taught itself that all people with brown eyes must also have brown hair. In these cases, the model would perform very well on the training data because it has learned the correlation rather than the actual target variables. However, this model would generalize poorly to an unbiased validation set, which manifests as significantly lower validation accuracies when the correlation is not present.

The third metric we developed takes an array of training accuracies for each class and an array of validation accuracies for each class. Using a loop, it compares the gap between training and validation accuracy for each class. If any of these gaps between training and validation accuracy for a class is greater than 45%, the model is deemed to be biased due to poor generalization and potential overfitting.

For this research, we used a subset of data with less bias (the 0.1 correlation biased MNIST dataset) and a subset of data with heavy bias (the 0.9 correlation biased MNIST dataset). We used multiple models trained on both of these datasets to determine the thresholds values for our bias detection metrics. The models that were trained on the low correlation datasets would be considered unbiased by our detection metrics, but the 0.9 dataset would be considered an example of a biased dataset because of the presence of intentional heavy correlation between variables in it. We made adjustments to the thresholds so that the metrics would perform as intended.

4.6 Bias Mitigation Techniques - Mixed Bias Training

A model's performance relies on the type of training data it is fed. Using this knowledge, we would like to determine whether mixing unbiased and biased training data would yield higher test accuracies than training fully on biased data. If this mixture approach has a higher average test accuracy than group 4, then we can infer that time to curate biased data from unbiased data or unbiased data from potentially biased data is worth the resource-expense to reduce the bias of a model.

5. Results

5.1 Bias Detection Metrics

Below is a table containing the results of applying our tuned bias metrics to four different models from each group. Metric 1 refers to the standard deviation check for variance, metric 2 refers to the gap between the best and worst class accuracy, and metric 3 refers to the class gaps between training and validation accuracies.

	Group 1		Group 2		Group 3		Group 4	
Result	True	False	True	False	True	False	True	False
Metric 1	0	4	0	4	0	4	0	4
Metric 2	2	2	0	4	0	4	0	4
Metric 3	2	2	0	4	0	4	4	0

Table 6: Results of bias metrics on each group

When applying these metrics to multiple models in each group after tuning, we observed that the performance of our metrics relies strongly on the availability of an unbiased validation set for model testing. When observing the group 2 results (training and validation data are both from the high correlation dataset), we can see that none of our bias metrics were able to detect bias from the model despite it being trained on data that is heavily unbalanced. However, when the models that were trained on high bias data were tested on data without that correlation present, our bias metric 3 was very successful and identified bias in each of those models. This performance on models without an unbiased validation set was also observed when comparing group 1 (both the training and validation data were from the low correlation dataset) and group 3 (the training data from the low correlation dataset and the validation dataset from the unbiased validation set). There was very high variance among the classes for group 1 as the model struggled to classify certain digits, which led to metric 2 (the gap between maximum and minimum class accuracy) being true in 2/4 of our trials.

When applied to models that were tested on the unbiased validation set, we were able to determine that each of the group 3 models were unbiased by all metrics. Each group 4 model was biased using metric 3 to evaluate them. The gap between training and validation performance for group 4 models was extremely large, dropping from an average training accuracy of 100% to 16%. This caused metric 3 to deem each group 4 model biased. However, group 4's models all performed approximately equally poorly on the validation data, with the average gap between best and worst class accuracy being just over 11%. Thus, the check of the gap between best and worst accuracy and the standard deviation check were not fruitful due to this low variance between classes.

5.2 Bias Mitigation Techniques - Mixed Bias Training

Mixed Bias	Mean Train Accuracies	Mean Test Accuracies
Overall	99.94%	60.45%
Class 0	100%	61.37%
Class 1	100%	74.54%
Class 2	99.93%	55.99%
Class 3	99.88%	64.8%
Class 4	99.92%	61.82%
Class 5	99.97%	52.55%
Class 6	99.95%	68.62%
Class 7	100%	57.99%
Class 8	99.79%	51.59%
Class 9	99.97%	51.44%

Table 7: Mixed bias training results

When half of the training data is unbiased and half is biased, the average accuracies for both overall and class wise are much higher than only training on biased. However, it is still lower than training on unbiased data. Nonetheless, this higher test accuracy compared to group 4 implies that time spent to parse out biased features from potentially biased training data can be a useful preparation step in the model building process. By identifying potentially biased features in training data, a more unbiased fraction of the training dataset can be derived and incorporated into training data.

6. Discussion

When observing our metric results, we saw that our bias detection metrics rely on the availability of unbiased data for model validation and do not perform well when this condition is not met. Because of this condition, it is important that substantial time is devoted to identifying potential sources of bias in the dataset of interest before using these metrics. Our focus was on identifying the symptoms of distribution bias since we focused on high and low correlation datasets, and we tuned our models so they treated the low correlation models as unbiased and the high correlation models as biased. If a specific bias of interest and its potential causes are not identified prior to using these metrics, they can be difficult to tune in a meaningful way when you don't have a clear goal. Furthermore, if one is unaware of potential bias sources in their data, they cannot be sure if they have an unbiased validation set, which this model requires for optimal performance.

We also observed that the thresholds required for our metrics to perform as intended (particularly metrics 1 and 2) were much larger than we initially expected. This was largely due to the fact that our group 3 models had a lot of variance between classes, with an average gap between best and worst class accuracy at 24%. The models trained on the unbiased 0.1 dataset generally struggled to classify digits 5, 8, and 9 while being able to classify digit 1 with a substantially higher accuracy than most other classes. This could be an example of hardness in our dataset, as some digits may be naturally harder than others for the model to distinguish despite relatively equal training exposure. The group 4 models had much less variation between class accuracies, as they all performed similarly poorly. The average gap between the best and worst class accuracies for group 4 was approximately 11%, which is less than half of the gap observed in the group 3 models. The group 4 models also struggled to classify 5s and 9s, but the class accuracies had much less variance than the group 3 models. Because of this, metrics 1 and 2 had to be tuned to relatively large values to achieve our desired outcome. We placed much more emphasis on metric 3, which was the best at detecting overfitting and poor generalization as symptoms of our targeted distribution bias.

During testing, we found that metrics 1 and 2 were often not in agreement with one another. This aligned with our expectations, since metric 2 was developed to detect bias when the variance between classes was large and metric 1 may then fail. When they did agree while our models were being tuned, there was an observation that was 3 or more standard deviations away from the mean while also meeting or exceeding the required gap between best and worst accuracy. Metric 3 appears to be mostly independent of the other two. It frequently detects bias on its own when there is a large gap between training and validation accuracy but the variance between classes is large but below 30%. Metric 3 was our most effective option out of the metrics we developed for detecting distribution bias in our models.

In terms of bias mitigation, our mixed training approach where half of the training data contained highly biased data and half contains unbiased data proved to show higher average test accuracies across the board. This is unsurprising as the model was given the chance to learn relationships that were not tied to the purposefully added distractions. These positive effects show that, like our metrics, the availability of unbiased data is essential for detecting and mitigating bias from models. Though it may be time-consuming and financially expensive to seek out potential biases from data, our results imply that these additional costs are necessary for improving the robustness of models.

7. Conclusion

Image classification serves as a potent and effective tool across a splay of domains, yet its ability to accurately predict and its robustness to spurious correlations rests upon the identification of biased features within the available data. Given that considerable attention has been given to seed out any potential biases, our metrics, specifically metric 3, have the capacity to testify whether a model is biased or not. To lessen the overreliance a model may have on these spurious features, a mixture of training data from biased and unbiased sets have shown to improve performance from training that is entirely biased. Our metrics and mitigation methods suggest that the basis for identifying and improving model biases lies within an individual's ability to parse out any biases that permeate the available data.

References

Viadinugroho, R. A. A. (2025, January 21). *Imbalanced classification in Python: Smote-tomek links method*. Towards Data Science. https://towardsdatascience.com/imbalanced-classification-in-python-smote-tomek-links-method-6e48dfe69bbc/

Alam, Z., Roy, T., Kawsar, H. M. N., & Rimi, I. (2024). Enhancing Transfer Learning for Medical Image Classification with SMOTE: A Comparative Study [Review of *Enhancing Transfer Learning for Medical Image Classification with SMOTE: A Comparative Study*]. *Arxiv*. https://arxiv.org/abs/2412.20235

Shrestha, R., Kafle, K., & Kanan., C. (2024). Occam-nets-v1. Github. https://github.com/erobic/occam-nets-v1

Taghanaki, S., Khani, A., Khani, F., Gholami, A., Tran, L., Mahdavi-Amiri, A., & Hamarneh, G. (2022). *MaskTune: Mitigating Spurious Correlations by Forcing to Explore*. https://proceedings.neurips.cc/paper_files/paper/2022/file/93be245fce00a9bb2333c17ceae4b732-Paper-Conference.pdf