PRECOG Report : Lazy Learner

Navraj Singh

February 12, 2026

Contents

1 Introduction

Neural Networks are Lazy Learners, they exhibit the phenomenon of simplicity bias which means that they rely on the simplest features to classify the data correctly [1]. This has been extensively studied in recent years, and it has also been proven that for certain architectures the model relies completely on the Linear Feature. [1,2]. In this report we will try to establish certain claims regarding the training dynamics of a 3 layer CNN model as well as a Modified ResNET18 model on biased dataset.

A short description of the progress made on each task:

- Task 1: It was established with extremely rleiable evidence that training a lazy CNN or ResNET18 moddel using a 95 % biased dataset is not possible, the 5% unbiased samples act as Robust Examples and steer the model to capture the shape dynamics of the digit. A considerable amount of effort was spent to diagnose and even eradicate this robustness as detailed in the Task 1 section, but it would be established with certainity in section 2 that we required a completely different approach to get rid of this robustness.
- Task 2: We analysed a robust and a lazy model to probe what exactly the model relies on to predict its output.
- Task 3: A GradCam class was implemented from scratch and results supporting our previous claims were observed.
- Task 4 : We used two different techniques to make the lazy model robust, and it was estabilised that they were not able to make the model prefer shape of the digits rather than color, in the end we used a different approach to achieve high test set accuracy.
- Task 5 : We used a PGD attack on both the robust and lazy model to quanitfy how easy it is fool each one of them.
- Task 6 : An SAE was trained and used to steer a model's prediction in the wrong direction after 'killing off' its most dominant feature.

2 Task 0

We used the MNIST dataset as a base to make our modified dataset, there were different kinds of datasets that were created for reasons that will be stated in the Task 1 section.

- Dataset 1: We used the whole MNIST dataset (0-9) and added a specific color to specific digits based on a pre-defined color mapping. The test/hard set was then made by using the pre-defined color mapping and assigning the color of the next digit to its previous one (in a cyclic fashion). This dataset has an inherent noise in the background of the same color as the digit.

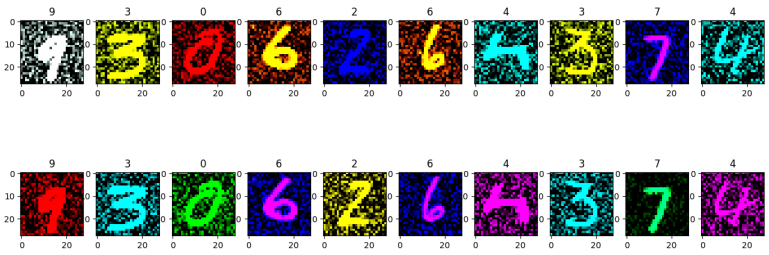


Figure 1: Top Row: Train Dataset, Bottom Row: Test Dataset. As it is evident that in the test dataset '3' was assigned 'Cyan' color (which was assigned to '4' in the train dataset.)

- Dataset 2: Another dataset was created but with a flat background and the collor applied on to the stroke of the digit. The color map of the test dataset is formed in the same fashion.

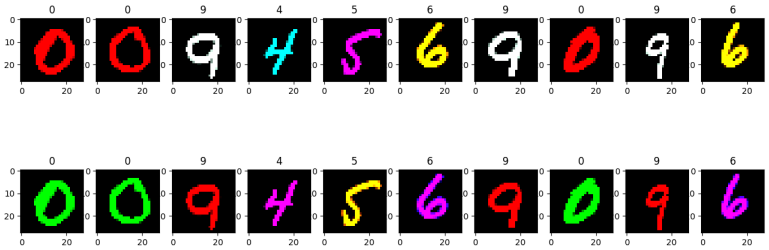


Figure 2: Top Row: Train Dataset, Bottom Row: Test Dataset. The test and train color maps are formed in the same fashion, only distinction is the removal of background noise.

- Dataset 3, 4: We used only the first 6 classes (0-5) to create a smaller dataset, for reasons which again will be clarified in the next section. The only distinction in dataset 3 and 4 is that of the background noise and flat background of their images.

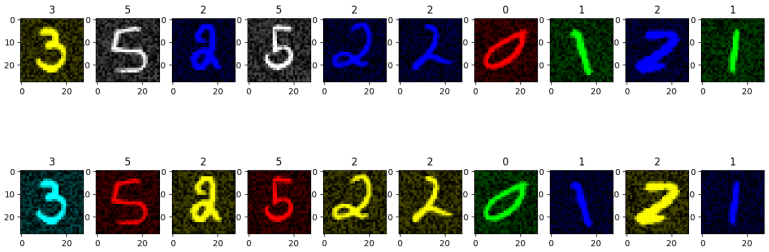


Figure 3: Top Row: Train Dataset, Bottom Row: Test Dataset. The test and train color maps are formed in the same fashion,we have used just the images with labels form 0-5.

3 Task 1

During the initial training of a CNN (3 Convolutional layers) and a ResNET18(modified to handle 28 by 28 images of MNIST) it was observed that both the models were performing extremely well on the Hard/Test Set, this was an unexpected observation, which led to a diagnostic search for the same, we used a variety of parameters and techniques to exactly pin-point the details of why this was happening.

3.1 Diagnosing Accidental Robustness

There can be a lot of culprits which caused the model to be robust:

- Training on a flat background makes it easier for the model to see the image thus relying on shape more than color
- The colors of the Digits are not visually separable enough so the model relies on shapes of digits instead.
- The model is too smart to just the color of the image.

Note : All the accuracy measures can be verified from the python notebook, where all the experiments are labeled.

All of these reasons were tested for their plausibility by training the model on various types of datasets and various different architectures.

- For the claim of flat background being an easy task , we curated a background with noise on it (even to a 0.5 intensity value), this meant that the model wouldnt be able to see the image and thus would rely on the color. The results observed were that model's Hard Set accuracy was still 80%-90%. (The data claims are all available in the python Notebook).
- We curated another dataset (Dataset 3 and 4) with images only with 0 to 5 class labels so that the color mapping would be distinct enough, to verify this claim, the model performed extremely well on the both Background Noise and Flat Background Hard Dataset.
- This claim was the easiest to verify and reject by training models of varying sizes, a simple 2 Layer CNN with just 8 filters in both the layers achieves satisfactory accuracy on the Hard Dataset.

Now the last plausible reason left was the one discussed earlier, the non biased images act as "Vaccinations against The Bias", this claim was verified by increasing the amount of biased examples in the Training Dataset.

It was observed that as the Bias Increased the Training Loss of the model decreased faster, indicating that the model started relying more and more in the color of the images to classify them, but in the Hard Dataset the accuracy which was 0% in the first epoch (for 99%) bias started to increase substantially with each epoch reaching 80% by epoch 7 or 8.

During the training of the model on 100% biased dataset it was observed that the train loss plummets instantly implying that the model relies heavily on the color metri but test accuracy goes to 0 and never improves.

3.2 Diagnosis and Result

This suggests that the Non-Biased examples do help the model to be robust against the Simpler Features like color and force it to learn Shapes of Digits for classification.

So we train our lazy model with 100% bias, as even 99.95% bias leads to an observed increase in accuracy after 15 epochs.

The confusion matrix of the model on both the train and Hard dataset was as expected, there was a clear diagonal in the train dataset confusion matrix and an off diagonal in the Hard dataset as we moved the maps cyclically by 1 index.

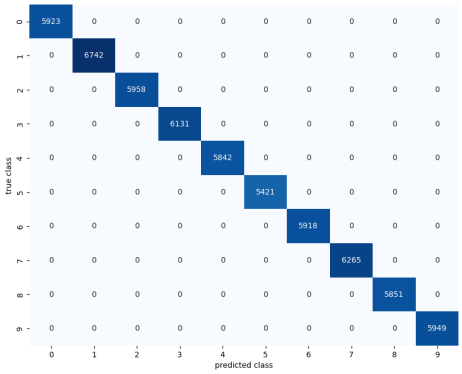


Figure 4: Confusion Matrix of Lazy Model on Train Dataset

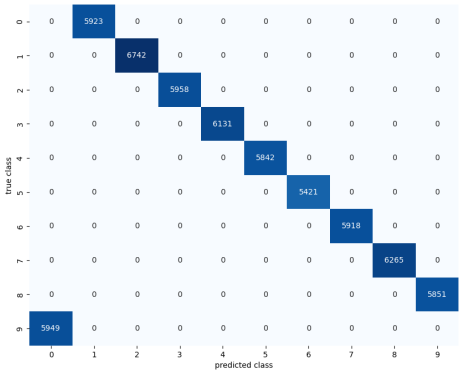


Figure 5: Confusion Matrix of Lazy Model on Hard Dataset, Shifted by a diagonal.

4 Task 2

In this task we were required to use an optimisable image tensor and optimise it to increase the Channel Wise mean activations of various filters, using this we would be able to probe what exactly the Model "sees".

4.1 Results

We analysed two models, one was trained with 100% (Lazy Model) bias and the other one with 95% bias (robust model)

4.1.1 Lazy Model

The first layers' 16 filters are just color detector filters which detect various colors, we expected redundancy in these filters as there were 10 colors 3 of which were primary and 16 filters.

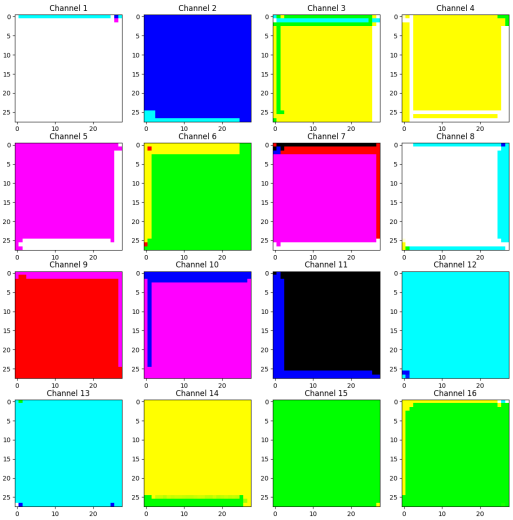


Figure 6: Filter for Layer 1 of Lazy Model

The last convolutional layer's 32 filter of the lazy model are not easily interpretable, they are random noise thus implying that model doesnot rely on the shape of the image, something which is different in the robust model

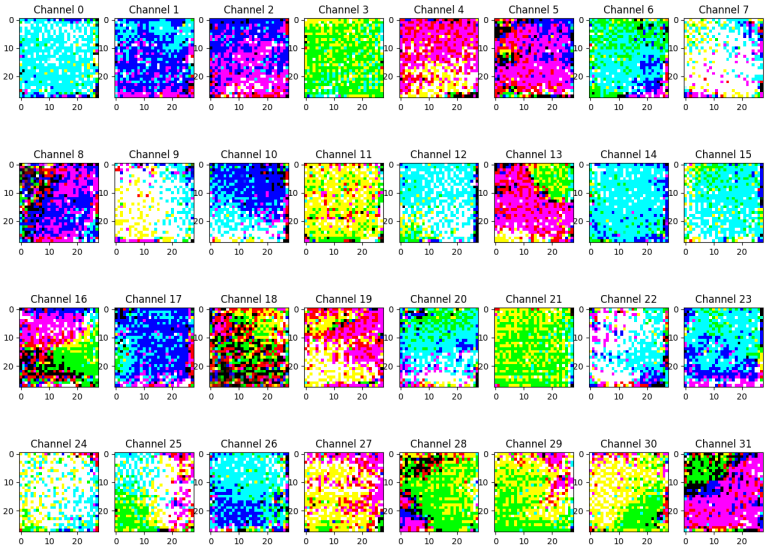


Figure 7: Layer 3 Filters, they are random noise not meant to extract any kind of spatial info or edge detection.

We can perform a simple test on our lazy model, this will allow us to better understand what kind of images leads to our model's prediction being what they are, lets say we want to know what kind of image would make our model predict it to be 1 with the most confidence.

We observe (Figure 8) that an image with just blue noise makes our model's prediction to be maximum for class label 2. This is due to the fact that our model was trained on the images of class 2 with blue color.

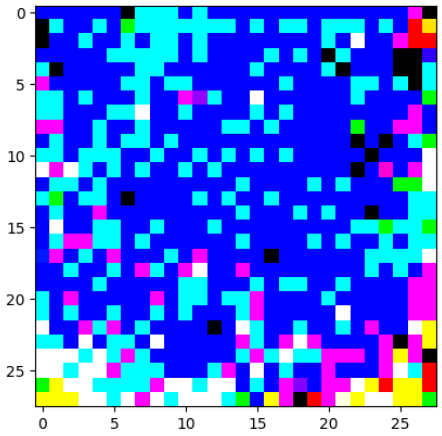


Figure 8: The optimised image for prediction to be 2 is random blue noise.

4.1.2 Robust Model

As established above a model trained with even 95% image bias is a robust model, so we should expect to see some kind of edge detectors like "Gabor Filters" in the inner layers. The first layer of the robust model is still a color detection layer similar to the lazy model but the inner layer's feature maps do act like edge detectors as evident from figures 9, 10.

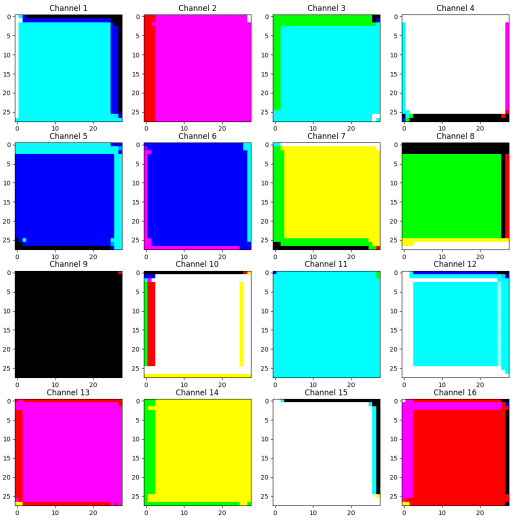


Figure 9: Robust Model Layer 1.

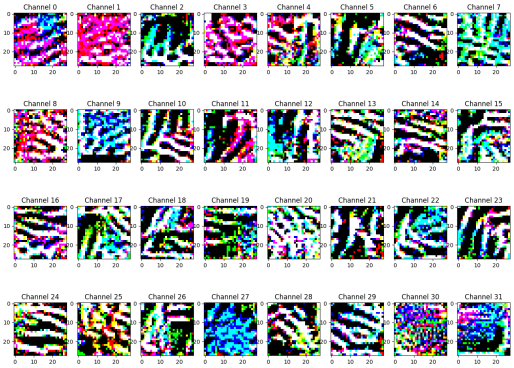


Figure 10: Robust Model Layer 3.We can see the evidence of Gabor Filters and Edge Detectors.

Now if we do the same analysis of prediction maximizing for the model to predict 2 with the highest confidence , we get an image with curves similar to the curves of digit 2 as seen in figure 11.

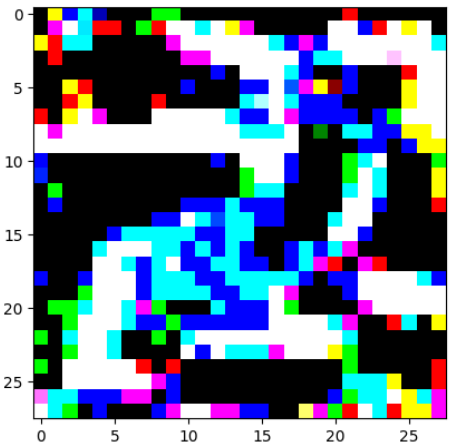
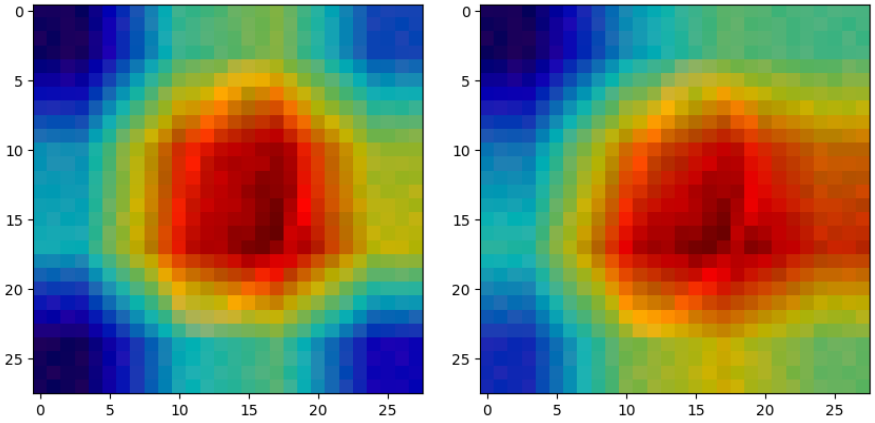
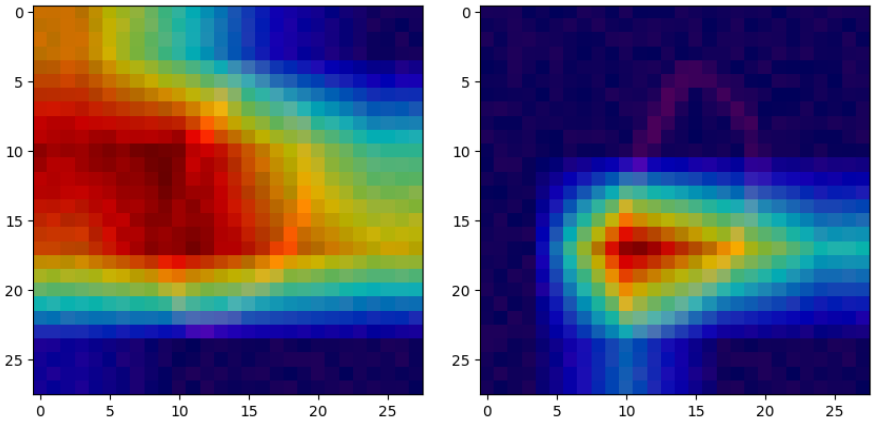


Figure 11: Robust Model Optimised Image for 2.We can see the evidence of Curves with shape like the digit 2.



(a) Heatmap for Prediction : 0 (b) Heat Map for prediction : 6

Figure 12: GradCAM Heatmaps for Lazy Model.



(a) Heatmap for Prediction : 0 (b) Heat Map for prediction : 6

Figure 13: GradCAM Heatmaps for Robust Model.

5 Task 3

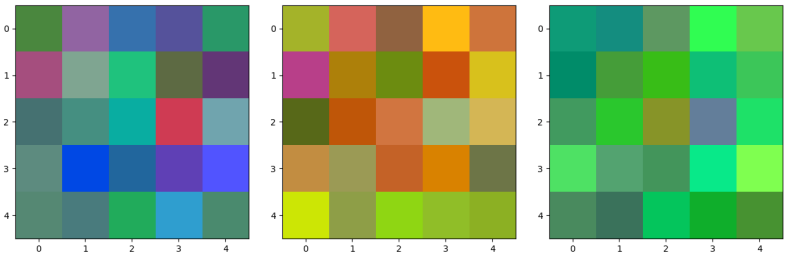
Here a GradCAM was implemented from scratch, this allowed us to know exactly where the model sees when it predicts a certain class. We expect the lazy and robust model to look at different regions when they predict the class. A lazy model will just look at a circular region centered around the digit as it looks for color but a robust model will look for the exact curves and shapes in the image.

5.1 Lazy Model

The lazy model's GradCAM heatmaps show that the model while predicting the True Label for the image just looks at the circular region near the image , disregarding any spatial information and shape information. For an image of 0, to predict it just looks at the region of the image where 0 is present and to predict 6 on the same image it looks again at the similar region (because 6 has a color mapping of orange and 0 of red so the red channel activates the 6 prediction as it finds one of the constituent colors of orange in that region).

5.2 Robust Model

In the Heat Maps of Robust model we observe that the model actually looks for shape and spatial information, for example while predicting 0 it looks at the global curvature of the image and to predict 6 on the same image of 0 it looks at the curvature of the lower left corner of 0 which is similar to the lower loop of digit 6.



(a) 1st RSV Kernel (b) 2nd RSV Kernel (c) 3rd RSV Kernel

Figure 14: Visualization of the top 3 Right Singular Vector Kernels.

6 Task 4

In this task we were asked to create a Robust Model from the pretrained model, We used two different approaches both of which yielded no substantial result so we switched to a different approach.

6.1 Ortho P

This approach pioneered by Morwani [2] suggests that Lazy Models learn just a Low-Dimensional projection of the dataset thus they miss out on the features which are much more rich and information dense, this leads to them being brittle. They propose a novel approach of training a model on a biased dataset, the model after learning the Lower Dimensional Projections of the inputs will become a Lazy Model, now we train another model which uses the same inputs as the previous model but we remove the Singular Directions Learned by the initial model from the inputs so that the Lower Dimensional Features will not be learned twice, this is done by Projcting the inputs into the Orthogonal Subspace of the Projection matrix learned by initial model, hence the name Ortho-P. The ensemble of these models creates a robust model which is immune to the lazy learning behavior.

6.1.1 Analysis

If we analyse the Weight Matrix of our Lazy Models using Singular Value Decomposition, we learn that the singular values of the weight matrix dont fall off as described in [2] thus the effective rank of the Matrix is not low in comparison to 16 (the number of kernels in first layer) so we cant find a Projection Matrix which projects the input vectors into a Lower Dimensional Subspce. This is also evident from the Right Singular Directions represented by the Kernels of the model, they are not a single color as we expected them to be but a mixture of various shades.

So this approach was discarded as it wouldnt be feasible to train a robust model using this approach.

6.2 Color Penalty and Region Penalty

In this approach we tried to create a new loss function where if the model used the pixels outside of the Digit Pixels then it would be penalised moreover if the model used the Kernels which prioritise color features to make predictions, it was penalised heavilt.

The color penalty was achieved by adding a term of the Standard Deviation of the Gradients of 3 different channels of the input image, and the Region Penalty was calculated by masking the background pixels and calculating the L2 Norm of the gradients of the same. Bothe these terms were added to the vanilla Cross Entropy Loss along with their respective scaling factors (Details are available in the notebook)

6.2.1 Analysis

The loss terms were quite unstable thus the task of training the model to ignore the background pixels as well as the color of the digit's pixels in the image could not be achieved, the test accuracy (as

evident in the python notebook) struggled to reach even 30% for a model already trained of a 100% biased dataset.

Thus We used the 95% biased dataset to train a Robust Model and Compare its Accuracy with a Lazy Model. The Robust Model achieved 95% accuracy on the hard dataset.

7 Task 5

We used a Projected Gradient Descent Attack, whose implementation was obtained from a previous personal project. The PGD attack basically finds out small perturbations in an input image which would decrease the model's loss w.r.t the target label specified in the PGD attack. This means trying to fool a Neural Network to classify an image of 7 as an image of 3.

7.1 Results

The results of PGD Attack were surprising to say the least, both the Robust and the Lazy Model were hard to fool. A peculiar observation was noted where a lazy model was harder to fool because it relied simply on the color of the image, thus slightly changing the pixel values by adding targeted noise meant nothing but if we gave the PGD Attacker freedom to change the values significantly then it simply changed the color of the Input Image to the pre-defined color of the Target Label.

A robust model was harder to fool on an epsilon value of 0.05 but on increasing the epsilon value slightly we observed the Robust Model to deviate from its predictions.

8 Task 6

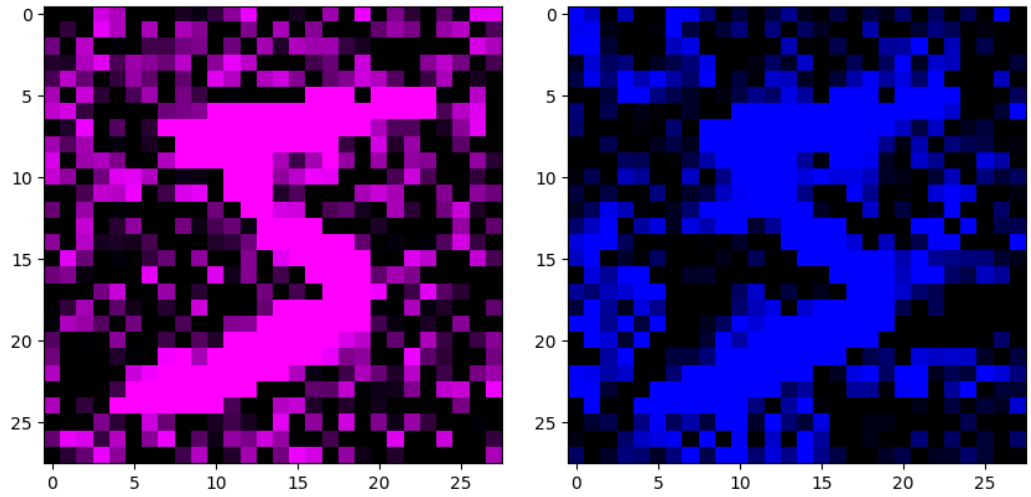
In this task we trained a Sparse Auto Encoder on the penultimate layer's activations (size 128) of the model, the expansion factor was 10 thus we observed 1280 hidden dimensions.

8.1 Training Results

- The model converged successfully over 15 epochs, reducing its training loss from 24.86 to 1.06.
- Reconstruction Loss dropped significantly from 24.15 to 0.58. This proves that SAE learned a high-fidelity representation of the CNN's internal state implying that it can perfectly reconstruct the model's "thoughts."
- The average number of active neurons (L0 Norm) was 536.1 (42%) sparse, this level of sparsity was enough for the SAE to disentangle the dominant color bias from spatial features.

8.2 Steering / Intervention

We selected a biased Image where the model's confidence was high (85%) and after passing it through the SAE we identified feature 472 as the most dominant feature.After manually subtracting this feature from model's hidden state and the model changed its prediction from 5 to 3along with a confidence drop to 52%.



(a) Original Input Image (b) Adversarial Image Using PGD Attack

Figure 15: Lazy Model's Adversarial Image, Note that the PGD Attacker converted the Image from Magenta to Blue. This is because the target label for PGD Attack was 2 and Color Map for 2 was Blue.

References

[1] H. Shah et al., *The Pitfalls of Simplicity Bias in Neural Networks*, NeurIPS, 2020.

[2] D. Morwani et al., *Simplicity Bias in 1-Hidden Layer Neural Networks, 2023*

[3] T. Bricken et al., *Towards Monosemanticity: Decomposing Language Models*, Anthropic, 2023.

[4] A. Madry et al., *Towards Deep Learning Models Resistant to Adversarial Attacks*, ICLR, 2018.