The Lazy Artist

Explorations Into Shortcut Learning in Convolutional Neural Networks

Sai Kapil Bharadwaj

sai.yeddanapudi@research.iiit.ac.in

February 9, 2026

First Page Details

As mentioned in the introduction

As of Saturday the 7th of February:

- Task 0
- Task 1
- Task 2 - Only dabbled my toes in the water for this one
- Task 3
- Task 4 - One of two intervention methods have been implemented.

I skipped Task 2 and decided would come back to it later after task 5. I spent a lot of time on task 1, a bit too much. Task 2 was comparatively more open ended than tasks 3 and 4, meaning it would drown more of my time, moreover 3 and 4 felt like direct next steps after task 1 (fool it, prove it, fix it).

A note to the reader

Why I chose this task

- When I first read the tasks, I was most clueless about CV and Quant
- With Graphs and NLP, I felt a probable approach for the tasks
- I was equally clueless about Quant as CV, but between the two, I leaned towards CV
- Maybe this was because of the thousands of hours I spent playing video games in yesteryears, or that XAI is interesting to me
- **I have zero substantial experience in machine learning and computer vision.**
- What I want to showcase here is my thought process, and scientific approach to problems. What better environment is there to showcase my critical process than the one where I have no prior knowledge of, and need to critically think through every small thing (even those that will seem trivial to a learned mind).
- This presentation is an extensive, detailed walkthrough of my journey.

A note to the reader

Important Links

- I don't know where this started, but I picked up a habit of documenting every thought, every decision, every hurdle and every setback.
- You can find exactly what I did, day wise, what happened, the status of my journey, *my raw, unfiltered notes here* (viewer discretion is advised).
- As I will discuss later on, I had to tinker a bit with my CNN to get close to the accuracy numbers in Task 1. You can find the *google sheet with my experiments and results here*

Task 0 : Generating the Datasets

Foreground or Background - Not so trivial!?

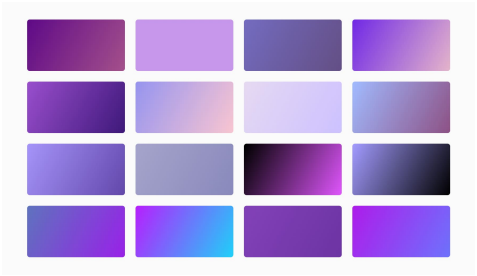
- The first bifurcation point in the decision tree was whether to color in the **background** or **foreground**.
- The twist that the background color cannot be a solid one, made it feel like coloring the foreground digit strokes would be easier to implement.
- But quickly, I realised we needed to use Grad-CAM to "prove" what the CNN was looking at
- If I color the foreground digits, the digit's edges and shape are spatially entangled with the color bias, are they not?
- So when I use Grad-CAM, and it shows heat blobs on the digit, how will I know if the model is focusing on the edges and shape of the digit or the color bias inside the digit
- Hence, I chose to color the background, spatially un-tangling the shape and edges from the color bias, so that my Grad-CAM analysis in Task 3 would be clearer.

Task 0 : Generating the Datasets

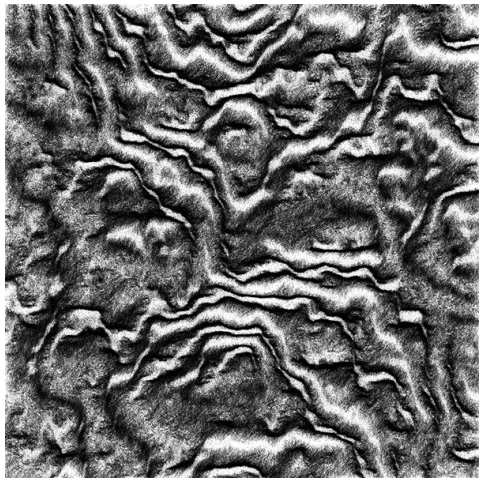
How do you color the background - Part 1

- This in itself, I feel, is an interesting thing to explore: *how will different techniques of biasing the background, like noise with color tints, low frequency color fields, textured patterns, effect how the model cheats?*

(a) High-frequency color-tinted noise



(b) Low-frequency smooth color fields



(c) Textured background patterns (perlin here)

Figure: Background biasing strategies used to induce spurious correlations in Colored MNIST.

Task 0 : Generating the Datasets

How do you color the background - Part 2

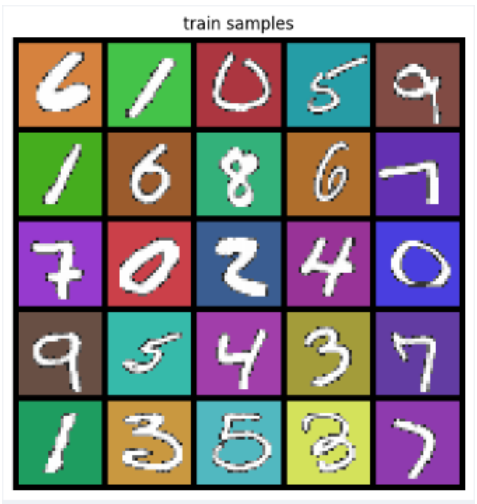
Background Coloring	Shortcut Feature Introduced	What the Model Learns if it Cheats	Expected Grad-CAM Behavior
High-frequency tinted noise	Global channel-wise color statistics (mean RGB bias across the image)	Associates the *overall chromatic distribution* of the background with the class label, relying on aggregated color statistics rather than spatial structure	Diffuse, low-contrast activation spread broadly across the background rather than digit contours
Low-frequency color fields (smooth gradients / blobs)	Large, spatially coherent color regions with class-dependent chromatic bias	Detects the presence and location of dominant color regions using localized background color cues as a strong predictive signal	Localized, high-intensity activation over smooth color blobs with minimal attention to digit strokes
Explicit textured backgrounds (patterns, stripes, Perlin noise)	Joint color–texture patterns correlated with class labels	Learns texture-sensitive features that entangle color and spatial pattern, treating background texture as a discriminative object	Sharp, high-contrast activation aligned with texture elements, often ignoring digit shape entirely

Task 0 : Generating the Datasets

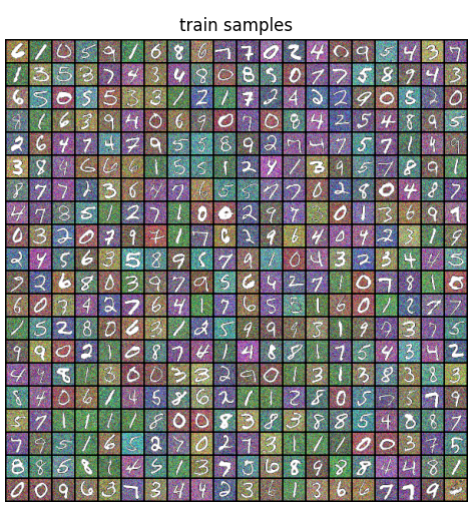
How do you color the background - Part 3

- Considering that this was my first time being exposed to all of this, I naturally believed everything the LLM told me.
- I chose **high frequency random gaussian-noise tined with color** to bias my MNIST data sets, because according to the LLM, this was the simplest and most straightforward way, given my situation.

(a) Biasing with 100% noise



(b) Biasing with 100% color



(c) Biasing with 60% noise and 40% color mixed

Task 0 : Generating the Datasets

How do you color the background - Part 4

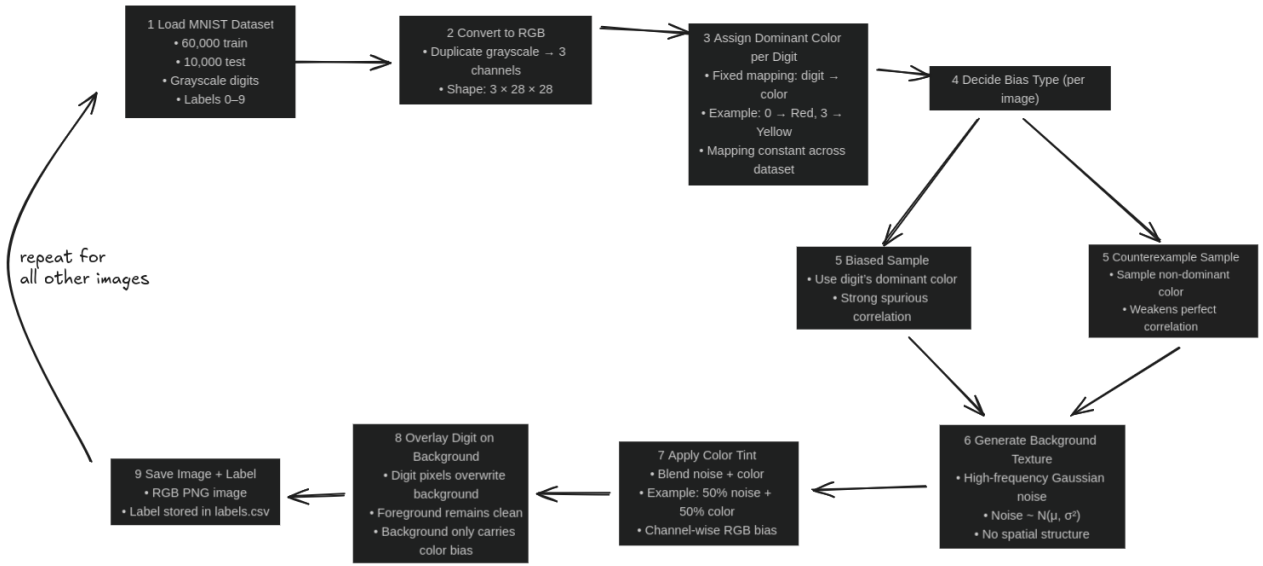


Figure: Pipeline illustrating the generation of biased Colored MNIST samples, from grayscale digit loading to background bias injection and dataset splitting

To Ponder

How will the proportion in which noise and color is put together affect the model's cheating behaviour?

Task 1 : Training and Fooling a CNN

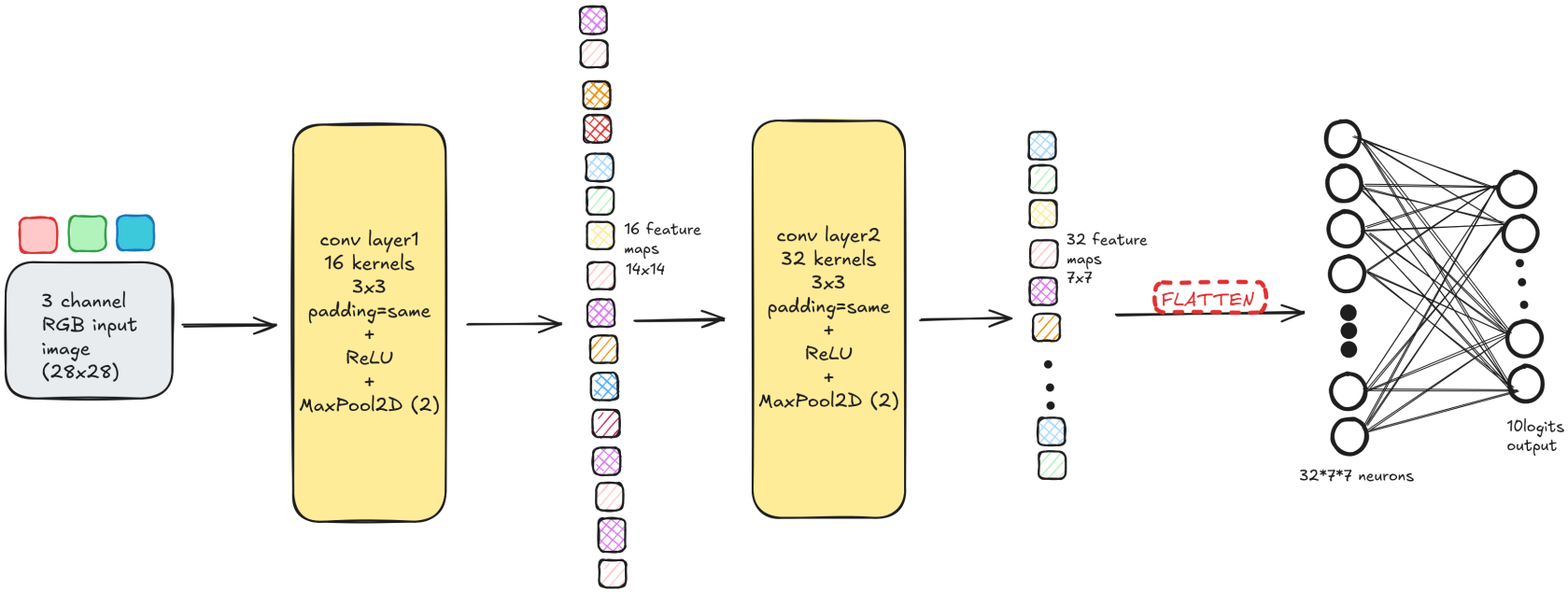
Gathering prerequisites

Since I knew nothing about CNNs, I had to spend some considerable time learning about them. The notes I made and some of the ideas I tried to capture from the below resources are present in the sections Day3 to Day6 of my notes, mentioned in the introduction. Resources I went through to learn about and build my CNN:

- 3Blue1Brown's Deep Learning playlist and the video on convolutions
- An excellent presentation by Brandon Rohrer who explained a seemingly complex topic in such an easy way.
- A helpful, practical guide to implementing simple CNNs I found on github: Bengal1's Simple CNN Guide

Task 1 : Training and Fooling a CNN

Initial Architecture



- This was trained for 7 epochs, and at this point, I did not yet realize the importance of hyperparameter tuning.

Task 1 : Training and Fooling a CNN

The Battle for Accuracies - 1

Some Stupidity from Me

- For task 0, I thoroughly read the LLM's code before running it
- It worked quite well; some trust began to build
- I ran the code for task 1 without reading it!
- There it was, 2 days gone
- It was training the model to predict the background color mapping directly and not the digit label itself - took me a while to catch this bug
- I even began to question my data generation process and thought something could have been wrong with that!

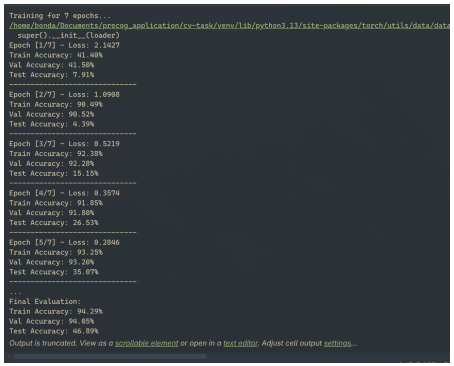
```
  super().__init__(loader)
Epoch [1/7] - Loss: 0.5168
Train Accuracy: 94.79%
Val Accuracy: 94.73%
Test Accuracy: 46.51%
------------------------------
Epoch [2/7] - Loss: 0.1707
Train Accuracy: 95.79%
Val Accuracy: 95.93%
Test Accuracy: 60.58%
------------------------------
Epoch [3/7] - Loss: 0.1184
Train Accuracy: 96.81%
Val Accuracy: 97.02%
Test Accuracy: 70.58%
------------------------------
Epoch [4/7] - Loss: 0.0878
Train Accuracy: 97.87%
Val Accuracy: 97.92%
Test Accuracy: 78.88%
------------------------------
Epoch [5/7] - Loss: 0.0672
Train Accuracy: 98.21%
Val Accuracy: 98.23%
Test Accuracy: 81.93%
------------------------------
Epoch [6/7] - Loss: 0.0567
Train Accuracy: 98.62%
Val Accuracy: 98.47%
Test Accuracy: 84.54%
```

Task 1 : Training and Fooling a CNN

The Battle for Accuracies - 2

Things I looked At

- Event after fixing the training logic error, the model was still getting 40%+ consistently on the hard set.
- I was doubting my architecture and data generation processes heavily



(a) Model achieving high accuracy on hard set too

(b) Dataset but with the digit edges blurred before overlaying

(c) Dataset with blurred digit edges and solid background colors

(d) Dataset with tinted noise, but digits are faint, blending into the background

Task 1 : Training and Fooling a CNN

The Battle for Accuracies - 3

I tried looking at papers that tried to illustrate shortcut learning of CNNs on CMNIST. Below were some I skimmed through to learn about their experimental setup, since my model was still too accurate on the hard set. Below are the key findings I felt were relevant to my case:

1. Arjovsky et al. Invariant Risk Minimization They used **MLPs** instead of CNNs (see section 5.2)
2. Ahmed et al. Systematic Generalizations with Group Invariant Predictions They used 3 convolutional layers and 1 fully connected layer. They provided the color palette used for their digits, which I incorporated into my method (see appendix)
3. Gowda et. al InBiaseD - Inductive Bias Distillation They too used **MLPs**, but they also had foreground and background biases in their dataset for CMNIST. This assured me that background bias was not the issue.
4. Vasudeva et al. They too used a **MLP** model to demonstrate shortcut learning on CMNIST. I noticed their citation in section 1.1 stating models trained using **Stochastic Gradient Descent** were more prone to learning "simple features".
5. Arpit et al. They too had background biases to CMNIST. Also they added **gaussian noise** to their CMNIST images and claimed that in some cases it, and **no proper regularization** improved shortcut learning.

Task 1 : Training and Fooling a CNN

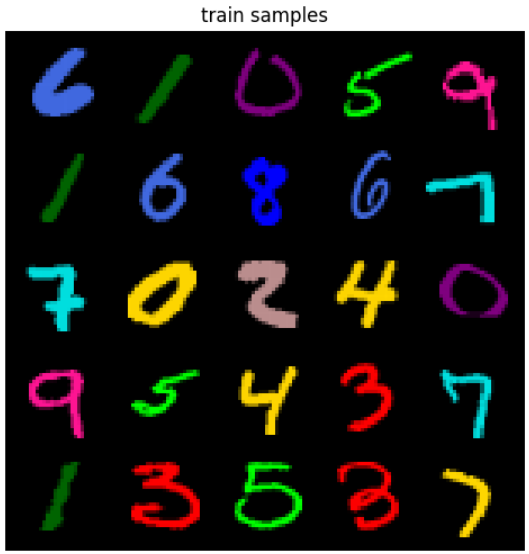
The Battle for Accuracies - 4

It was at this point where no matter what I tried, I could not get the accuracy on the hard set to fall below 20%. It felt like everyone else except me got past this stage. It really felt like I was doing something wrong and that this task should not have been this hard. It was at this point that I spoke to my colleagues: *Srinivas Kolla and Mukund Hebbar*. In an entry task like this, due to the shortage of resources (seats in Precog), I do understand the natural need for "gatekeeping" one might feel. I could sense this toxicity in the air. I don't think it's that deep. Luckily, those two also understood that research is about collaboration, and there is really no need for gatekeeping.
We discussed our strategies for tasks 0 and 1: data set generations, model architectures, reasoning etc. They tried to help me out. Still nothing still seemed to work.

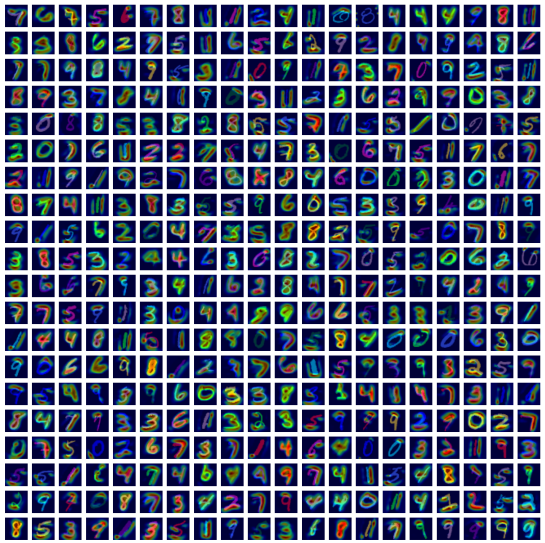
Task 1 : Training and Fooling a CNN

The Battle for Accuracies - 5

(a) Dataset generation using low frequency background textures (no noise)



(b) Dataset generation where foreground strokes are colored



(c) Gradcam heatmaps on model trained on dataset generated in (b)

Figure: Different dataset generation strategies I tried to use to get the model to cheat

I also spent a day trying to look at all the different hyperparameters involved and how their values will effect the cheating behaviour of my model (see day 8 in my notes).

Task 1 : Training and Fooling a CNN

The Battle for Accuracies - 6

- Till now, I never directly asked the AI to solve the problems I faced, without asking it to implement an idea I had to solve it.
- Time was running out, so I yielded. I gave Antigravity some restrictions for this task, and an objective to get the accuracy below 20% on the hard set.
- A 30-minute nap later, it was done.
- **The AI resorted to increasing the bias to 99% in the dataset**.
- This (I forgot to mention in my prompt) I felt was cheating and out of bounds of the rules of the task.
- However, it did use a **different model architecture** (added two fully connected layers before finaly output layer), which I incorporated into my design, along with other hyperparameter choices.
- This is when I really put my foot down on the gas, and decided to manually tweak all different hyperparameters and record how the model's accuracies change.
- *Google sheet with my experiments and results here*

Task 1 : Training and Fooling a CNN

The Battle for Accuracies - 7

In the next few slides, I present some of my experiments and the insights I drew from them.

- Trials 1-29 were performed using my initial CNN architecture, where the only fully connected layer is from the flattened feature maps of last conv layer to the output logits directly.
- In this architecture, the model consistently performed well, except on those where the bias was increased to 99% or the *alpha* attribute (determines how strongly to overlay the digit on the background) is decreased.

Task 1 : Training and Fooling a CNN

The Battle for Accuracies - 7

Insight 1

As the **percentage of bias** increased in training set, the model's accuracy generally dropped on the hard set. An increased bias gave the model fewer counter examples during its training.

Insight 2

Decreasing **batch size** allowed more gradient updates during training, this gave the model an opportunity to learn more features (shape too) resulting in greater accuracies on the hard test set as compared to using larger batch sizes.

Task 1 : Training and Fooling a CNN

The Battle for Accuracies - 7

Insight 3

Hypothesis: A larger **kernel size** would promote the model's ability to look at larger, global features, and hence learn the global average color bias better, leading to a decreased accuracy on the hard set; however this was false. Trials 20-24 show that the model had comparable performances between two kernel sizes.

Insight 4

Hypothesis: Fewer **number of filters in each conv layer** would mean that the model has lesser chance to learn shape (since few filters only), and hence, it would only learn color, and its accuracy would plumet on the hard set. Trials 25-29 show that this is not the case. I halved the number of filters in each layer by half, but the performance was still comparable.

Task 1 : Training and Fooling a CNN

The Battle for Accuracies - 8

- Trials 30-39 were performed using the model Antigravity generated: two conv layers, two fully connected layers and one more going to the logits.
- This is where things began to take a turn and the numbers started to move in my favour.
- I believe this model was "cheating better" because of the more fully connected layers it had. As mentioned previously, most papers demonstrating shortcut learning did use only MLPs for the CMNIST dataset, and the architecture having a greater number of fully connected layers made it more like an MLP.

Insight 5

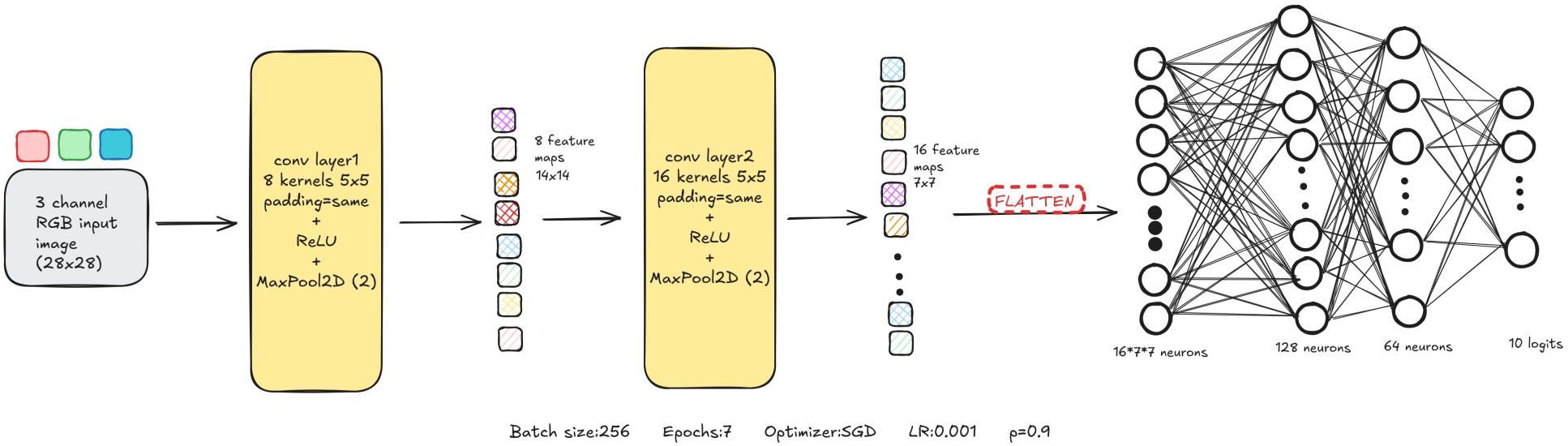
Training the model for a larger **number of epochs** drastically improved its performance on the hard set. This could be because it eventually began to learn the shape features as it passed through the dataset again and again (see trial 36)

Insight 6

Increasing the **learning rate** drastically improved the model's performance on the hard set. The loss landscape was quite complex, and a high learning rate could have caused it to overshoot narrow minima associated with the birttle shortcut features.

Task 1 : Training and Fooling a CNN

Final Architecture



- The final accuracy of this model on the data set with 40% noise and 60% color, generated in task 0, is **93.12%** on train and **8.13% on test**.

Task 1 : Training and Fooling a CNN

Results - Confusion Matrix

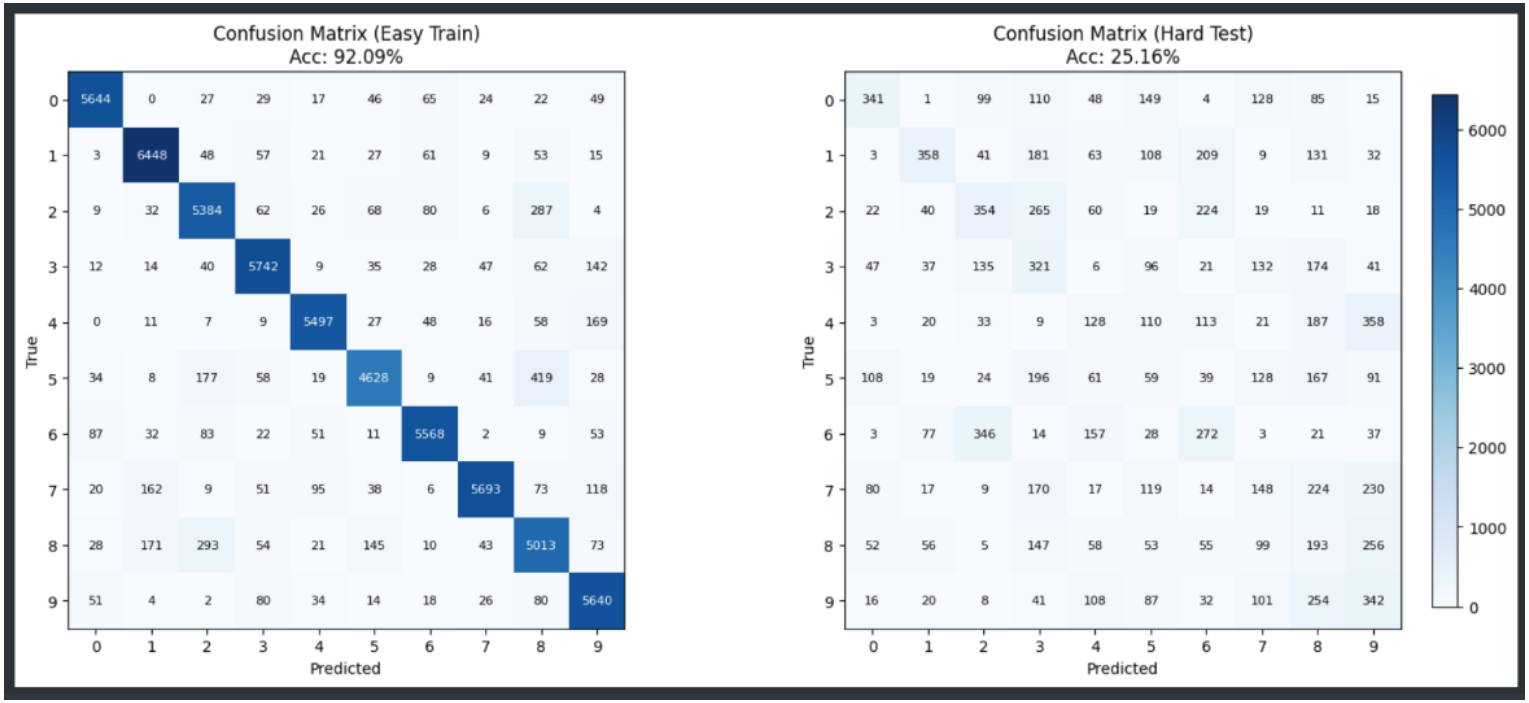


Figure: Confusion matrix when test set was generated randomly

- The test set was generated by setting the image to any random color out of the 9 colors it was allowed to take.
- To get a "better" confusion matrix and confusion analysis of the model and how it shortcut learned, it would be better if we inverted the correlation in the hard set instead of randomly assigning colors.
- I did this exactly. I re-generated my data set, but in the test set, all 0s take the color of 1s, all 1s take the color of 2s... all 9s take the color of 0s.

Task 1 : Training and Fooling a CNN

Results - Confusion Matrix

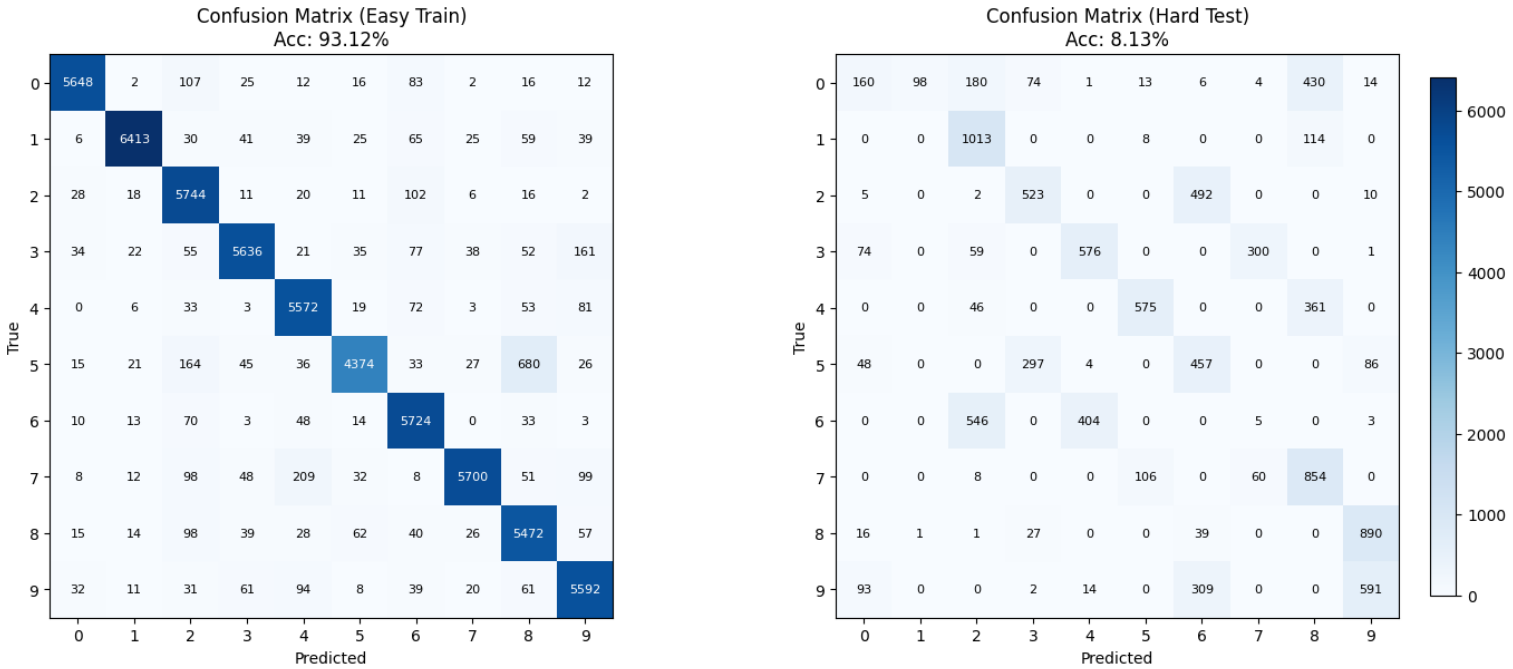


Figure: Confusion matrix when test set was generated by inverting the correlations

- Some consequences of shortcut learning are visible here.
- All 1s were colored like 2s, and the model prodicted 1s as 2s.
- Similar for 3,4,5,7,8.

Task 2 : Feature Visualization

- Task 1's hurdles consumed much of my time
- Task 2 was quite open ended and had no clear objective and one could go as far as they wanted. I did not have any "tangible" output to show yet.
- I wanted to do tasks 3 and 4 as they felt like the immediate next steps after fooling the model (proving it using gradcam and fixing it).
- I did come back to task 2 at the end, but as I want this presentation to represent my chronological journey, we will go through tasks 3 and 4 and circle back to 2 at the end.

Task 3 : Grad-CAM Heatmaps

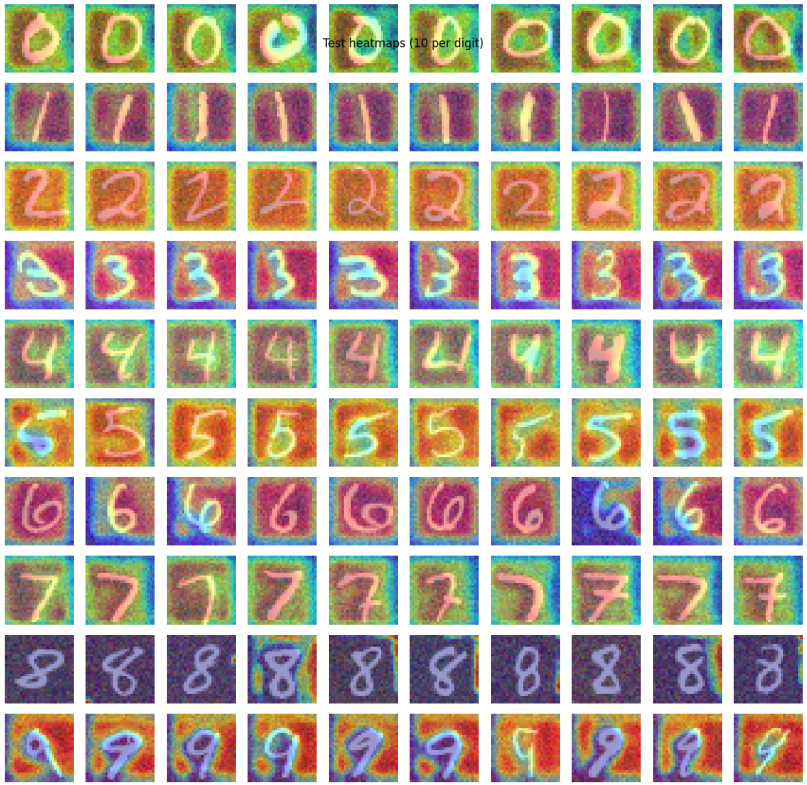
Gathering Prerequisites

- The first step was to learn about Grad-CAMs, what they were, how they worked and how to implement them from scratch
- To this end, I went through the following resources:
  - A video by A Data Odessy about Grad-CAMs
  - An excellent written article, an extension of the above video
- Luckily, this task was smooth sailing
- Everything from the theory, to the implementation just seemed to click nicely

Task 3 : Grad-CAM Heatmaps

Results

(a) Heatmaps on train set of model that learnt spurious features



(b) Heatmaps on test set of model that learnt spurious features

Figure: Generated Grad-CAM heatmaps

Task 3 : Grad-CAM Heatmaps

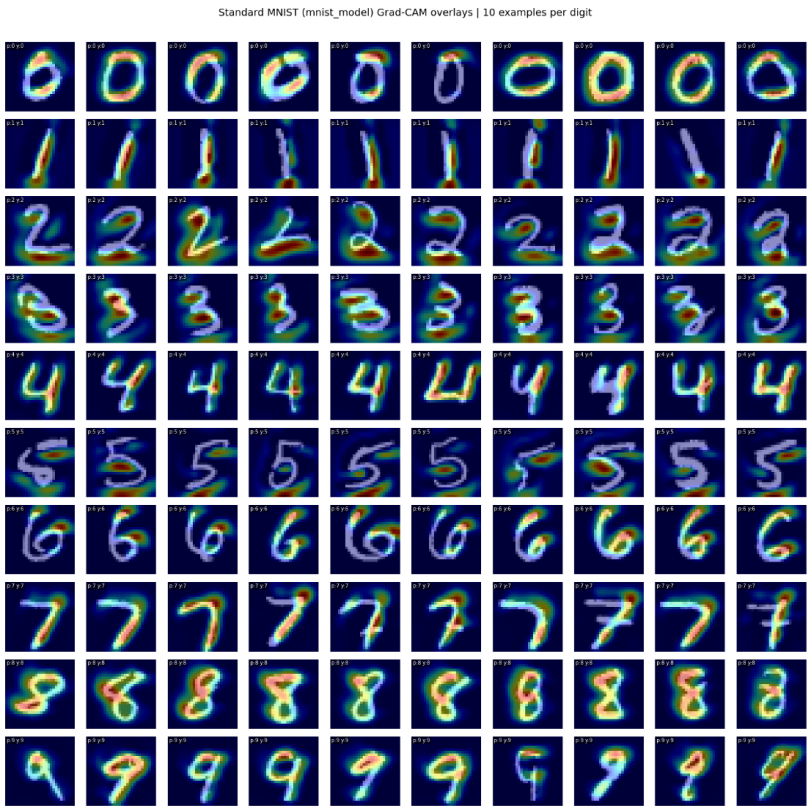
Results

Out of curiosity, I trained the same model architecture on the standard MNIST dataset. I then decided to evaluate it on the biased data set to see if it would apply the learnt knowledge about "edges and shapes" to the biased data set too and do well on it.

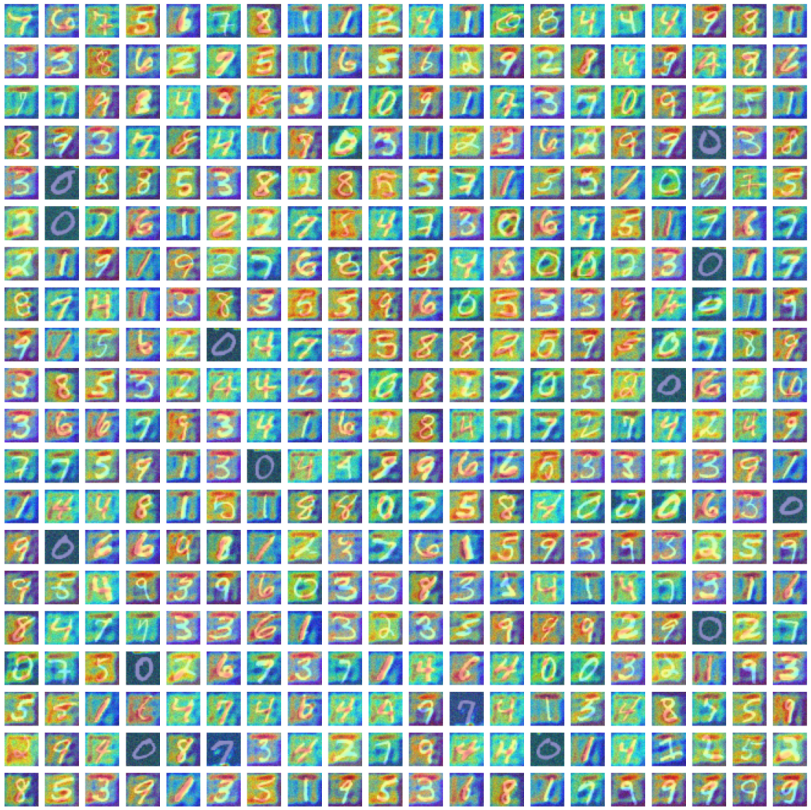
- The model did indeed learn shapes and edge features
- Suprisingly, eventhough it did learn the intended concepts, it evaluated to only about 30% on the colored data set

Task 3 : Grad-CAM Heatmaps

Results



(a) Heatmaps on MNIST of model trained on MNIST



(b) Heatmaps on CMNIST of model trained on MNIST

Task 4 : The Intervention

My Initial Thoughts

- By this time, I had only about 5 days left (before the extension)
- Although the task had asked me to be creative and implement some of my own creative ideas to get the model to beat the bias, I had already been swayed by the research papers I came across in Tasks 0 and 1
- Those papers were proposing different methods to prevent shortcut learning
- Of the ones that were achieving at least 70% on the debiased samples, the one that caught my eye the most was Bassi et al. Training with Explanations Alone
- The next few slides explain their approach and my implementation of it

Task 4 : The Intervention

Training with Explanations Alone (TEA) - Part 1

- TLDR; the proposed way of training to prevent shortcut learning is a **teacher-student** model approach
- Bassi et al : instead of training the student to match output label predictions with the robust teacher, they suggest to make the student **learn the explanation heat maps** of the teacher.
- Given that the teacher has indeed learnt the "intended features" and not shortcuts, making the student learn these output activation maps will essentially force it to learn the same features / look at the same areas in the image as the robust teacher
- As a natural consequence of learning the "reasoning" behind predictions, the student eventually learns to make correct predictions on the labels too

Task 4 : The Intervention

Training with Explanations Alone (TEA) - Part 2

method	Coloured MNIST				DogsWithTies			
	biased acc	unbiased acc	deceiv. bias acc	gen. gap ↓	biased acc	unbiased acc	deceiv. bias acc	gen. gap ↓
AG-Sononet [20]	$100_{\pm 0}$	$16_{\pm 0}$	$0_{\pm 0}$	$100_{\pm 0}$	$99_{\pm 2}$	$50_{\pm 1}$	$3_{\pm 2}$	$96_{\pm 3}$
ViT-B/16 [21]	$100_{\pm 0}$	$21_{\pm 0}$	$1_{\pm 0}$	$99_{\pm 0}$	$88_{\pm 3}$	$58_{\pm 4}$	$52_{\pm 5}$	$36_{\pm 6}$
standard classifier [22]	$100_{\pm 0}$	$18_{\pm 0}$	$0_{\pm 0}$	$100_{\pm 0}$	$98_{\pm 2}$	$69_{\pm 4}$	$41_{\pm 5}$	$57_{\pm 5}$
group robustness								
IRM [17]	$100_{\pm 0}$	$60_{\pm 1}$	$53_{\pm 1}$	$47_{\pm 1}$	$85_{\pm 4}$	$67_{\pm 4}$	$27_{\pm 4}$	$57_{\pm 5}$
GroupDRO [16]	$100_{\pm 0}$	$52_{\pm 2}$	$40_{\pm 2}$	$59_{\pm 2}$	$71_{\pm 4}$	$70_{\pm 4}$	$63_{\pm 5}$	$8_{\pm 6}$
PGI [15]	$100_{\pm 0}$	$64_{\pm 2}$	$58_{\pm 2}$	$42_{\pm 2}$	$88_{\pm 3}$	$71_{\pm 4}$	$56_{\pm 5}$	$32_{\pm 6}$
distillation								
TEA teacher†	$99_{\pm 0}$	$99_{\pm 0}$	$99_{\pm 0}$	$0_{\pm 0}$	$96_{\pm 2}$	$96_{\pm 2}$	$96_{\pm 2}$	$0_{\pm 3}$
st. G.-CAM & out. [14]	$100_{\pm 0}$	$91_{\pm 0}$	$85_{\pm 0}$	$15_{\pm 0}$	$75_{\pm 4}$	$72_{\pm 4}$	$70_{\pm 4}$	$5_{\pm 5}$
student rep. [12]	$100_{\pm 0}$	$92_{\pm 0}$	$87_{\pm 0}$	$13_{\pm 0}$	$53_{\pm 4}$	$53_{\pm 4}$	$55_{\pm 3}$	$2_{\pm 5}$
student outputs [11]	$100_{\pm 0}$	$93_{\pm 0}$	$87_{\pm 0}$	$13_{\pm 0}$	$87_{\pm 3}$	$81_{\pm 3}$	$76_{\pm 4}$	$11_{\pm 5}$
GALS [26]††	N/A	N/A	N/A	N/A	$76_{\pm 4}$	$74_{\pm 4}$	$72_{\pm 4}$	$4_{\pm 6}$
TEA student (ours)	$99_{\pm 0}$	$98_{\pm 0}$	$98_{\pm 0}$	$1_{\pm 0}$	$79_{\pm 4}$	$82_{\pm 4}$	$78_{\pm 4}$	$2_{\pm 5}$

Figure: Table 4 from Bassi et al. showing state of the art performance of TEA on CMNIST over other strategies to prevent shortcut learning

Task 4 : The Intervention

Training with Explanations Alone (TEA) - Part 3

- Before implementing this, I noticed something key in the paper
- The proposed method works best only when heatmaps are generated using Layer-wise Relevance Propogation (LRP)
- One of my main motives for adopting this method was that I already produced the Grad-CAM heatmaps for task 3, and I had hoped to use the same, however, Bassi et al. claim that TEA works best with LRP only
- Substituting LRP heatmaps with Grad-CAM heatmaps showed to significantly reduce TEA's resistance to bias
- The next few slides present what I understood about LRP and the TEA training loss function used to train the student model.

LRP Fundamentals

Bassi et al. present the below equations in Appendix A of their paper. Below are my understandings drawn from the same.

$$R_j^{L-1} = \sum_k \frac{w_{jk}^L \, a_j^L}{z_k^L + \text{sign}(z_k^L)\,\epsilon} \, R_k^L \tag{A1}$$

$$\text{where: } z_k^L = \sum_j w_{jk}^L \, a_j^L \tag{A2}$$

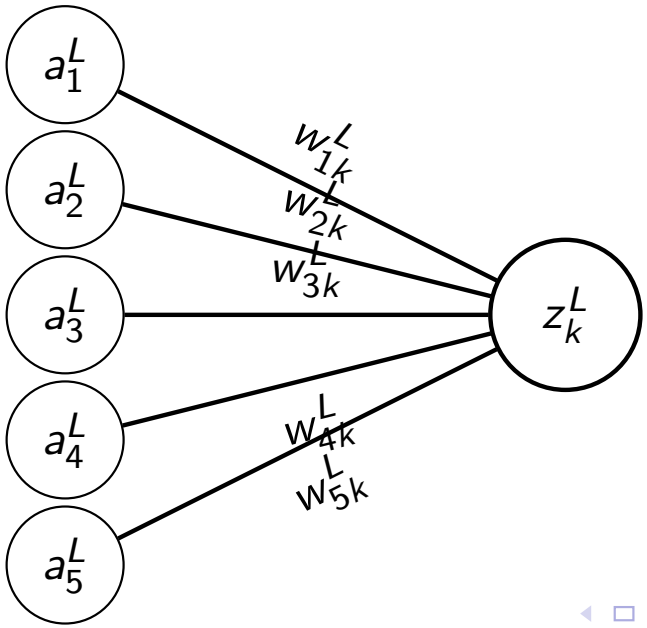
$$R_k^N = \begin{cases} z_k^N & \text{if } k = c \\ 0 & \text{otherwise} \end{cases} \tag{A3}$$

Task 4 : The Intervention

Training with Explanations Alone (TEA) - Part 5

$$z_k^L = \sum_j w_{jk}^L \, a_j^L \tag{A2}$$

- Z_k^L is the output of the k^{th} neuron in the L^{th} fully connected layer *before ReLU* activation
- It is function of the outputs of neurons j in the previous layer: a_j^L



Task 4 : The Intervention

Training with Explanations Alone (TEA) - Part 6

$$R_j^{L-1} = \sum_k \frac{w_{jk}^L \, a_j^L}{z_k^L + \text{sign}(z_k^L)\,\epsilon} \, R_k^L \tag{A1}$$

- This equation describes how *relevances* are propogated from neuron k in layer L, backwards to neuron j in layer $L-1$
- R_k^L denotes how relevant neuron k in layer L is to the output class (say c) of the neural network wrt which we are generating our heatmap to explain
- LRP backpropogation begins at one of the neurons in the final output layer (say neuron c in layer N) $R_c^N := Z_c^N$ (here we begin LRP backpropogation with the neuron associated with the class we want LRP heatmaps to explain - c)
- Relevances of other neurons in the last layer other than the interested class are set to 0: $R_{k!=c}^N := 0$

The Final Outputs of LRP Backpropogation

- LRP backpropogation ends at the input layer where LRP explanation heatmaps (ie relevance of each input pixel to the neural network's output) is formed: $\mathbf{R^0} = \left[R_k^0\right]$
- In the explanation heatmap $\mathbf{R^0} = \left[R_k^0\right]$:
  - Negative values indicate areas that *reduced* the classifier's confidence for the explained class c
  - Positive values indicate areas that increased the classifier's confidence for the explained class c
  - Low absolute relevance values near zero indicate areas the classifier paid little attention to

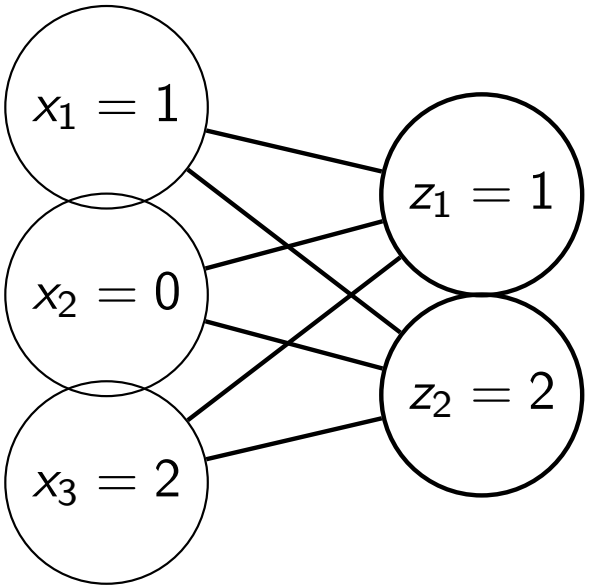
Deeper Look at Equation A1

$$R_j^{L-1} = \sum_k \frac{w_{jk}^L \, a_j^L}{z_k^L + \text{sign}(z_k^L)\,\epsilon} \, R_k^L \tag{A1}$$

- The amount of relevance backpropagated from neuron k to neuron j depends on the relative influence of neuron j on the output of neuron k (Z_k^L)
  - This is clearly represented in the fraction in A1
  - Numerator gives that component of Z_k^L coming from neuron j, and the denominator is the basically the total output of neuron k
  - The extra term in the denominator is to ensure the math does not break (if $Z_k^L = 0$), but also represents taylor series approximation losses (see relevant_papers/tea_deep_dive_annotated.pdf for more)

Task 4 : The Intervention

Training with Explanations Alone (TEA) - Part 9



$$W = \begin{bmatrix} 1 & 0.5 & 0 \\ 0 & 0.5 & 1 \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Procedure

1. Compute z_k via a forward linear pass.
2. If $z_k \leq 0$, set its relevance to zero (could be killed by ReLU).
3. Stabilize activations:

$$z_k \leftarrow z_k + \text{sign}(z_k)\,\epsilon$$

4. Normalize relevance:

$$s_k = \frac{R_k}{z_k}, \qquad \mathbf{s} = \begin{bmatrix} 4/z_1 \\ 2/z_2 \end{bmatrix} \approx \begin{bmatrix} 0.96 \\ 0.995 \end{bmatrix}$$

5. Distribute relevance back:

$$R_j = \sum_k w_{jk}\, a_j\, s_k$$

Vectorized form: $\mathbf{R} = W^\top \mathbf{s}$.

Task 4 : The Intervention

Training with Explanations Alone (TEA) - Part 10

Why TEA Works

- For bias to influence the classifier's output, its influence must flow from its position in the input image through all layers
- If the teacher classifier pays no attention to bias, no LRP relevance flows to the bias in its heatmaps, so similarly, no LRP will flow in the student's heatmap, making it pay no attention to bias

The issue with TEA and Grad-CAM

- Grad-CAM is based on intermediate representations (heatmaps are usually consturcted using activations of the last convolutional layer) and relevances don't propogate all the way back to the input image
- The main issue is that the student can match the intermediate representations by still learning the spurious features
- Grad-CAM suffers spurious-mapping: the last convolutional layers have large receptive fields. Features of the input image can be mapped to deep activations which are not aligned with these features.

The TEA Training Loss

$$d(\mathbf{H}_s, \mathbf{H}_t) = \frac{\|\mathbf{H}_s - \mathbf{H}_t\|_1}{\sqrt{\|\mathbf{H}_s\|_1 \, \|\mathbf{H}_t\|_1}}$$

- The above aims to measure the difference between the heatmaps between the teacher and the student
- L1 norm is just the sum of the absolute values of elements of the vector
- $\mathbf{H_s}$ *and* $\mathbf{H_t}$ are student and teacher heatmaps generated by LRP backpropogation taken to the input layer.
- Bassi et al. provide an explanation as to why they normalize this in this way (something to do with dealing with training instability in the initial stages of training).

Bassi et al. also explain the noisiness of heatmaps. To counteract this, they propose their novel **multi-resoultion heat loss**, which is what is used finally for training

$$\mathcal{L}_{\text{tea}}(\mathbf{H}_s, \mathbf{H}_t) = \frac{1}{M} \sum_{m=0}^{M-1} d(\text{AvgPool}(\mathbf{H}_s;\, K = 2^m), \text{AvgPool}(\mathbf{H}_t;\, K = 2^m))$$

Task 4 : The Intervention

Training with Explanations Alone (TEA) - Part 12

Implementation Details - 1

- For this entire process to remain scientific, the only thing we should change is the loss function during training.
- All other hyperparameters, including learning rate and batch size, must be kept the same as they were for Task 1.
- When I used the same hyperparameters as in Task 1 for Task 4, my model did not cross an accuracy of 10% on either the easy or the hard sets.
- This is definitely an indicator of something that is wrong with my architecture or experimental setup somewhere along the way from Task 0.

TEA training with the same parameters as in Task 1

Task 4 : The Intervention

Training with Explanations Alone (TEA) - Part 13

Implementation Details - 2

- Bassi et al. in Appendix F.3 discuss their exact hyperparameters. Their learning rate was 0.01 and batch size was 128.
- I retried TEA training, but changed learning rate from 0.001 to 0.005 and batch size from 256 to 128.
- The model then learnt the features and ignored the bias
  - Now, because I changed some hyperparameters, this increase in accuracy cannot solely be attributed to the training technique.
  - I could not do much. I have no explanation but a feeling that the loss landscape is very complex.

6) Run TEA training + save student weights

```
student_path = OUT_DIR / "student_weights.pt"

student = train_student_tea(
    epochs=7,
    batch_size=128,
    optimizer_name="sgd",
    lr=0.005,
    grad_clip_norm=5.0,
    background_threshold=0.9,
    student_heatmap_class="teacher",
    debug_first_batch=False,
)

torch.save(student.state_dict(), student_path)
print("Saved student weights:", student_path)
```

[8] ✓ 2m 49.9s

```
[Student TEA] Epoch 1/7 | tea_loss=2.584255
Easy train (biased) accuracy: 10.46%
Hard test (debiased) accuracy: 10.28%
[Student TEA] Epoch 2/7 | tea_loss=2.081607
Easy train (biased) accuracy: 10.45%
Hard test (debiased) accuracy: 10.28%
[Student TEA] Epoch 3/7 | tea_loss=1.999021
Easy train (biased) accuracy: 11.76%
Hard test (debiased) accuracy: 10.38%
[Student TEA] Epoch 4/7 | tea_loss=1.943929
Easy train (biased) accuracy: 20.51%
Hard test (debiased) accuracy: 19.23%
[Student TEA] Epoch 5/7 | tea_loss=1.862043
Easy train (biased) accuracy: 32.10%
Hard test (debiased) accuracy: 31.44%
[Student TEA] Epoch 6/7 | tea_loss=1.759918
Easy train (biased) accuracy: 62.26%
Hard test (debiased) accuracy: 61.44%
[Student TEA] Epoch 7/7 | tea_loss=1.654965
Easy train (biased) accuracy: 72.38%
Hard test (debiased) accuracy: 72.01%
Saved student weights: /home/bonda/Documents/precog_application/cv-task/task4/TEA/outputs/student_weights.pt
```

Results

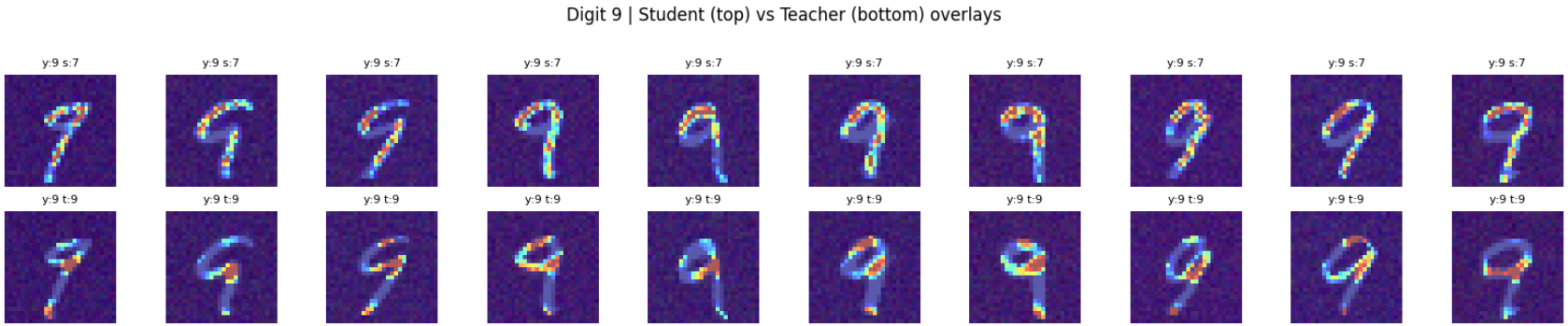


Figure: LRP Heatmaps of the student (after training) and teacher on the digit 9

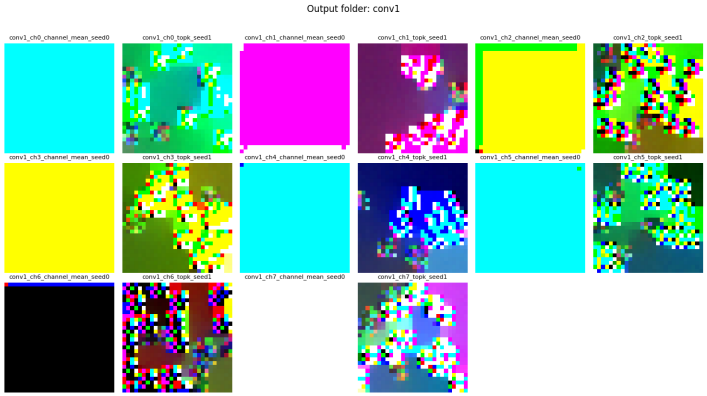
LRP heatmap overlays for other digits can be found at the end of task4/TEA/tea.ipynb

Tasks 2 and 5: Half-assed Attempts

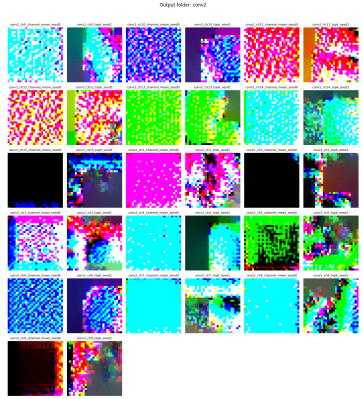
Yup you read that right. I was exhausted by this point, and it was Saturday 7th February by now. These two tasks, I did not spend much time on at all, although these are the most interesting parts in my opininon.
Regardless, I share whatever outputs I got

Task 2 : Optimized Image Tensors

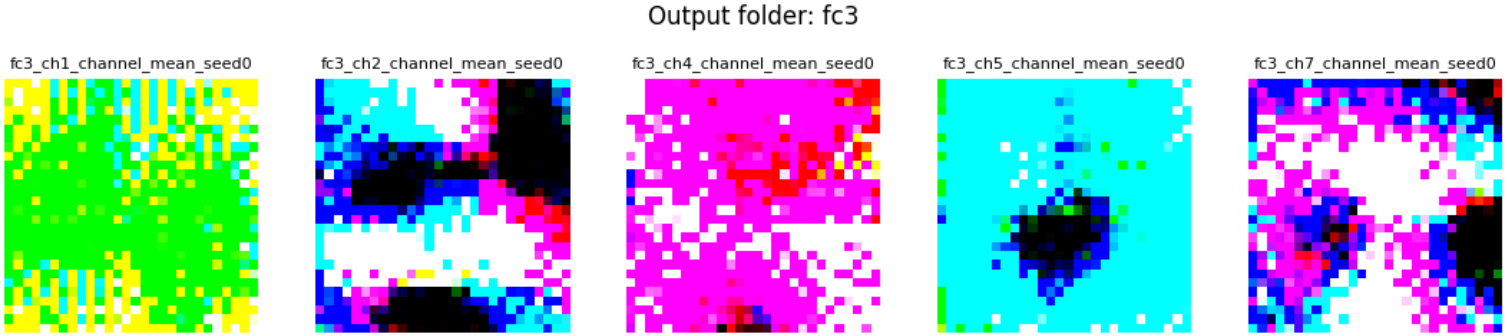
Some Output Images of the Biased Model



(a) Tensors optimized for different channels in conv1



(b) Tensors optimized for different channels in conv2

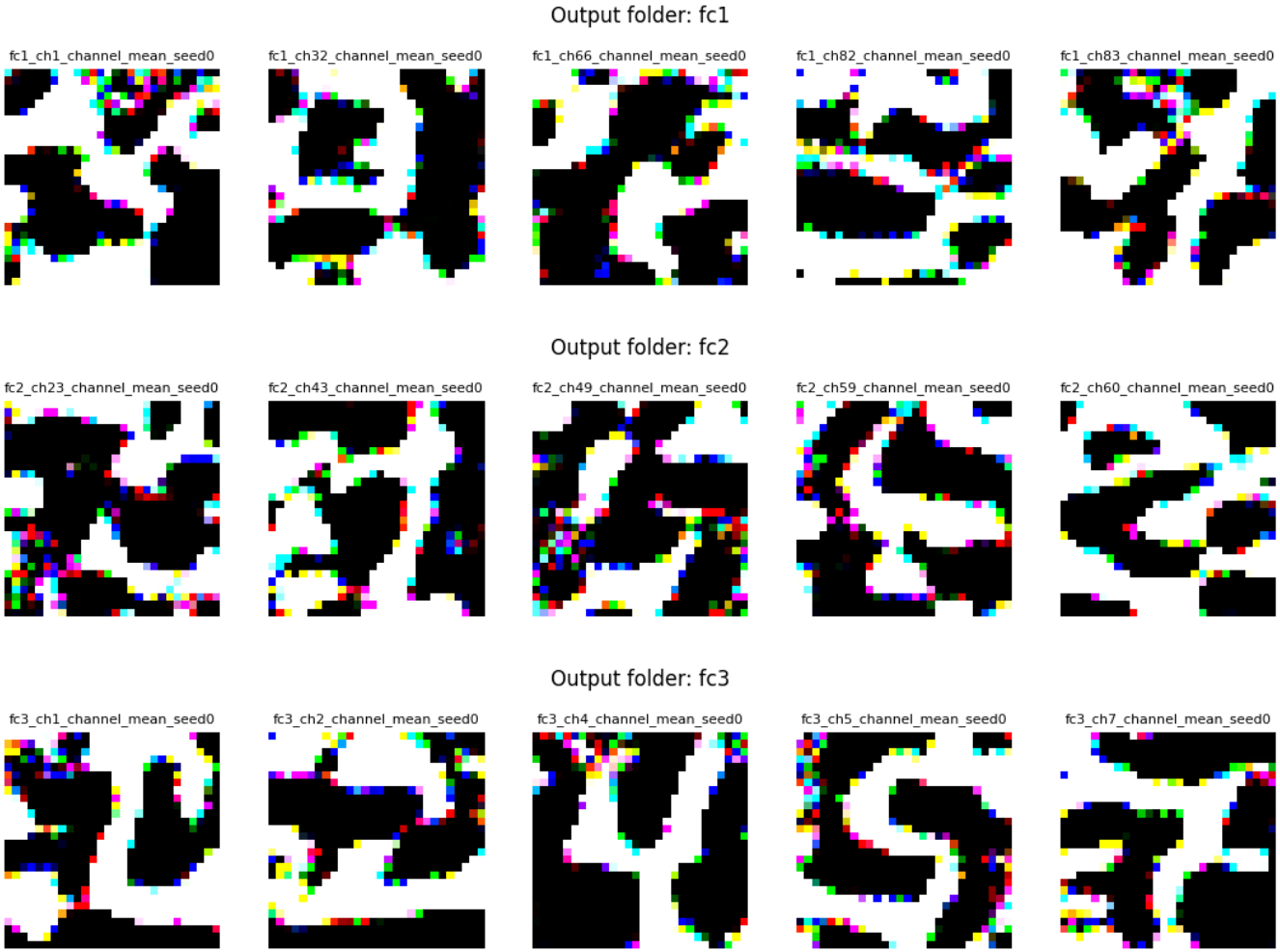


(c) Tensors optimized for different neurons in fc3

Figure: Early layers detect major colors, the rest, although

Task 2 : Optimized Image Tensors

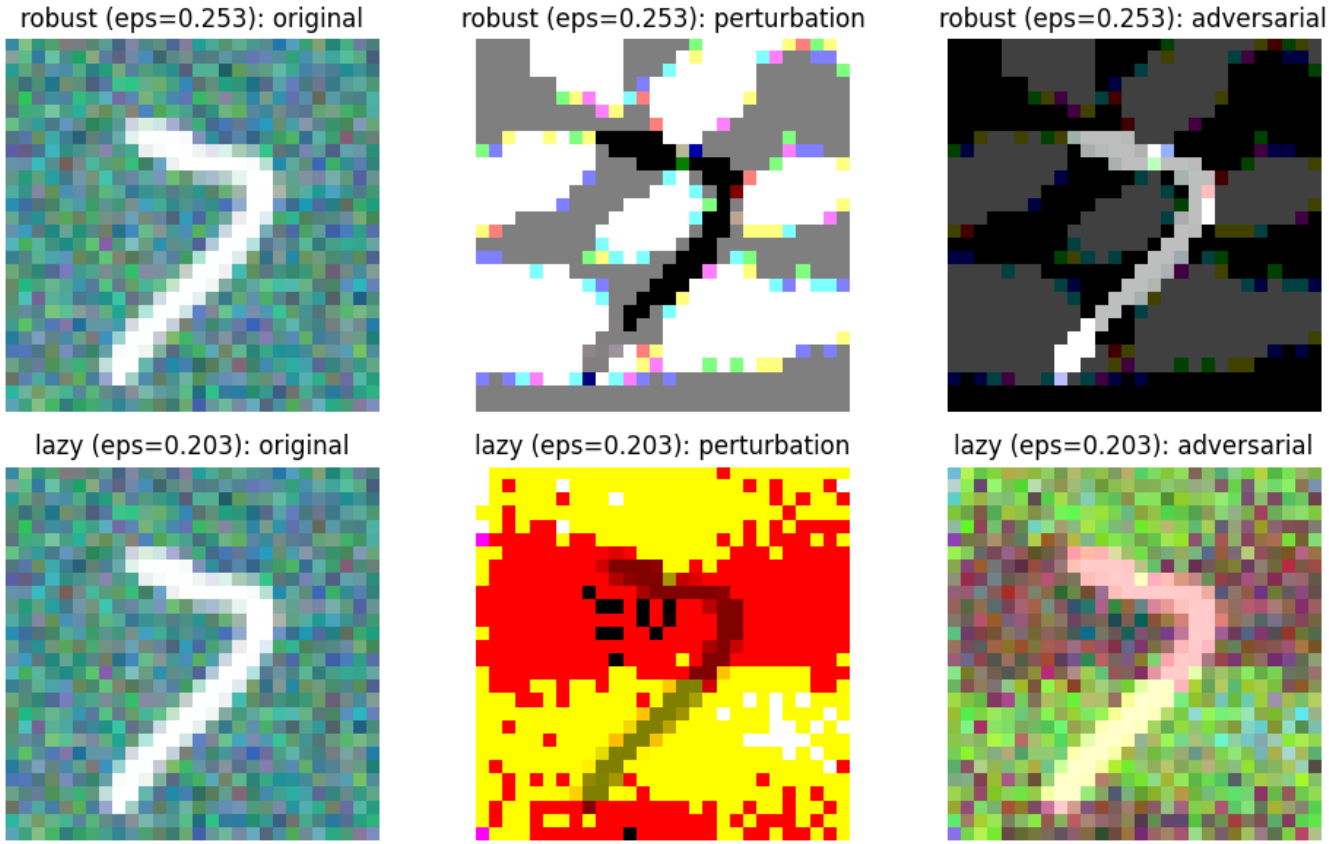
Some output images of model trained on MNIST



Although the convolutional layers were still uninterpretable (see experiments/task2/explore.ipynb), the tensors optimized for neurons in the fully connected layers were interpretable. Especially the ones in fc3, we can clearly see the shapes 4,5 and 7.

Task 5 : Adversarial Example Generation

On Monday morning, while on YouTube, I came across this video, and the way in which the creator explained these concepts gave me some energy to try task 5. This was done using one prompt and not really something I should be showing.



Conclusions

- In the beginning, I had a feeling that CNNs cheat *naturally* and quite easily
- However, as my journey progressed, I realised that this is not the case - the loss landscape was definitely very complex and not as trivial as I thought it to be
- I had to enfore very specific conditions, architectures, training setups for it to actually cheat
- I don't know if this is normal or if it was just a quirk of the MNIST dataset being relatively *"simple"* as compared to other datasets where shortcut learning is observed

Personal Takeaways

- I usually have very high standards for work and I'm quite harsh on myself
- From not knowing a single thing about CNNs to learning about CNNs, ML, Grad-CAMs, exploring the phenomenon of shortcut learning and more, I have to say that I have grown a bit
- I still believe the work is not the best quality and that there is a lot of scope for betterment of the work itself and my understandings, but I sure as hell am proud of the *effort* I put into all of this.
- **If given a chance to join Precog, I would love to continue this line of work and dive deeper into XAI**

Thank You

If you made it to the end and read all of my slides (which I spent HOURS making) thank you. I hope you can pardon any mistakes - after all it is my first time doing anything related to CV.