The Lazy Artist

PRECOG Recruitment Task - Computer Vision
2025121007, Anushka Pratap Singh

Task Completion Status

Task	Status
Task 0: Biased Canvas	Complete
Task 1: The Cheater	Complete
Task 2: The Prober	Complete
Task 3: Grad-CAM	Complete
Task 4: Intervention	Complete
Task 5: Adversarial	Complete
Task 6: SAE	Complete

1. Introduction

Neural networks are sophisticated pattern-matching systems that optimize input-output mappings to minimize training loss, without inherent understanding of causal relationships. This fundamental limitation makes them susceptible to "shortcut learning", the tendency to exploit non-causal, spurious correlations present in training data.

This project investigates shortcut learning through controlled experimentation, diagnostic interpretability techniques, and mitigation strategies. We deliberately inject spurious correlations into MNIST data, diagnose the resulting model pathologies, and explore interventions to force learning of causal features.

2. Task 0: The Biased Canvas - Dataset Construction

2.1 Objective

Create a synthetic dataset with controlled spurious correlation between digit identity (signal) and foreground color (noise) to study shortcut learning behavior.

2.2 Methodology

We constructed a Colored-MNIST dataset by modifying standard MNIST digits with the following specifications:

Color Mapping: 10 distinct colors assigned to digits 0-9:
• Digit 0 → Red (255, 0, 0)
• Digit 1 → Green (0, 255, 0)
• Digit 2 → Blue (0, 0, 255)
• Digit 3 → Yellow (255, 255, 0)
• Digit 4 → Magenta (255, 0, 255)
• Digit 5 → Cyan (0, 255, 255)
• Digit 6 → Orange (255, 128, 0)
• Digit 7 → Purple (128, 0, 255)
• Digit 8 → Sky Blue (0, 128, 255)
• Digit 9 → Gray (128, 128, 128)

Easy Set (Train/Val): Strong spurious correlation
• 95% of each digit rendered in its dominant color
• 5% counter-examples with random colors
• Maintains consistent bias pattern across train and validation

Hard Test Set: Correlation destruction
• Each digit receives a color randomly sampled from the 9 non-dominant colors
• No consistent color-digit mapping exists
• Forces reliance on shape features for correct classification

2.3 Design Rationale

We tried multiple methods:

Failed Approach 1 - Initially, the hard test set used a deterministic inversion where digit d received color (d+1) mod 10. Result: 89.68% hard test accuracy. Why? The colors still had a perfect one-to-one relationship with digits, just shifted. The model's final layer simply adjusted to the new mapping.

Failed Approach 2 - Reducing model capacity did not force shape learning. The color signal wasn’t cheap enough for the model to cheat.

Failed Approach 3 - Spreading the color across the background (80% of the image) instead of the foreground didn't work. The statistical relationship stayed strong enough for the model to exploit.

Successful Approach - We randomly assigned colors in the hard test set, each digit could be any of 9 colors (everything except its training color). This completely broke the color-digit

correlation.

Key Insight: Shortcuts dominate only when they are both (1) cheaper than the true signal and (2) statistically reliable. Random color assignment in the hard set breaks condition (2), forcing the model to confront its shape-learning deficit.

2.4 Results and Validation

Distribution Analysis:

EASY SET (Train) - Dominant Color Distribution:
Digit 0: 95.31% red, Digit 1: 94.91% green, Digit 2: 94.78% blue,
Digit 3: 95.12% yellow, Digit 4: 95.75% magenta, Digit 5: 95.46% cyan,
Digit 6: 95.03% orange, Digit 7: 94.84% purple, Digit 8: 95.06% sky,
Digit 9: 94.76% gray

HARD TEST SET - Dominant Color Distribution:
All digits: 0.00% in dominant color (perfect exclusion verified)

This establishes a controlled experimental environment where:
• The Easy set provides a strong, exploitable shortcut
• The Hard set renders color information statistically useless
• Performance gap directly measures shortcut reliance

3. Task 1: The Cheater - Baseline Model Training

3.1 Objective

Train a standard CNN on the biased dataset to demonstrate shortcut learning and quantify the generalization gap.

3.2 Model Architecture

A simple 3-layer CNN was employed as the baseline "lazy model":
• 3 convolutional layers with ReLU activation
• Max pooling after each convolution
• Fully connected classification head
• Standard cross-entropy loss with SGD optimizer

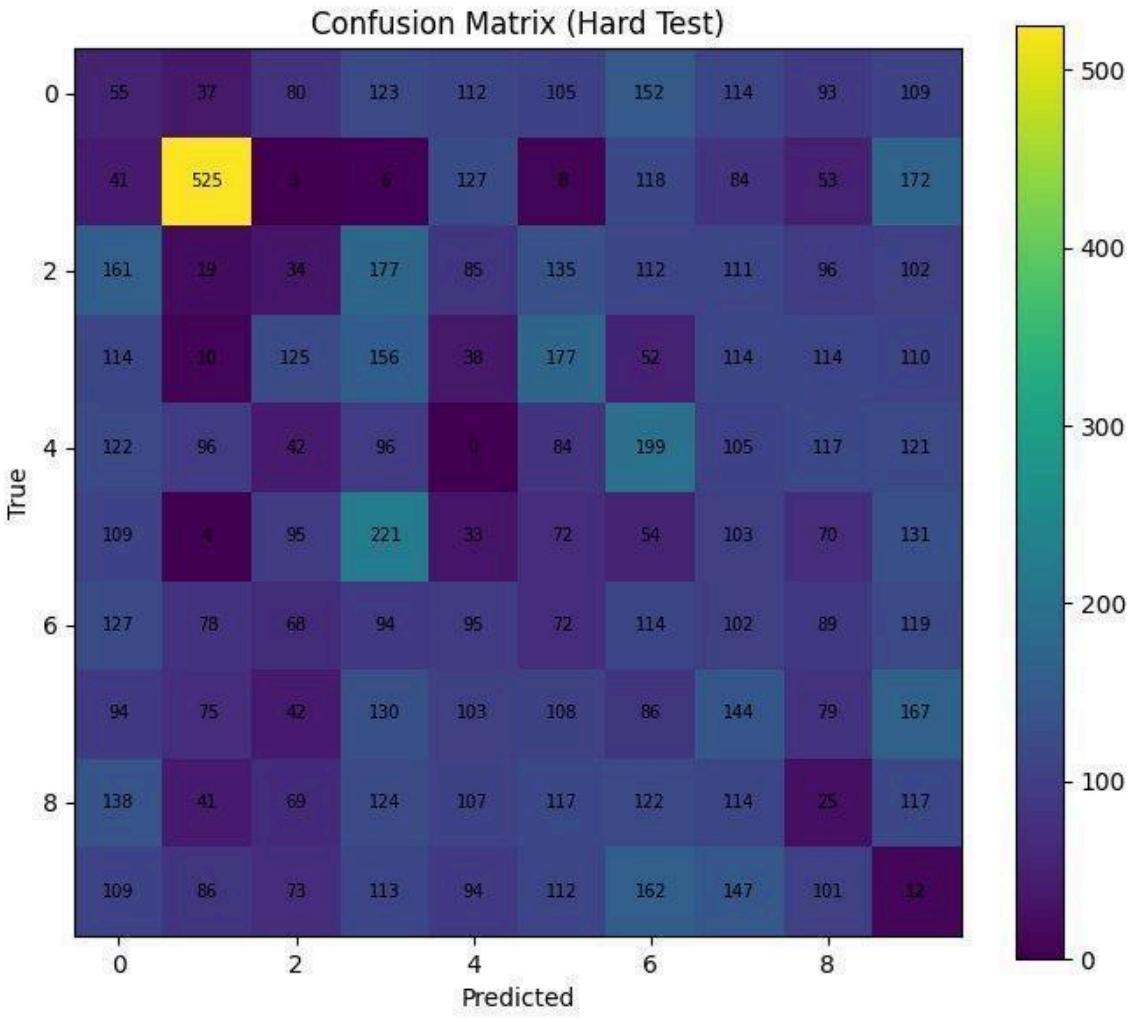
3.3 Results

Training Progression:
Epoch 1: Train 91.84%, Easy Val 95.16%, Hard 0.15%
Epoch 2: Train 95.07%, Easy Val 95.30%, Hard 3.34%
Epoch 3: Train 95.17%, Easy Val 95.21%, Hard 3.65%
Epoch 4: Train 95.33%, Easy Val 95.70%, Hard 11.52%

The 84-percentage-point gap between Easy Val and Hard Test provides strong evidence of shortcut learning. The model achieves near-perfect performance when the spurious correlation holds but degrades to near-random (10% baseline) when it breaks.

3.4 Confusion Matrix Analysis

The confusion matrix on the hard test set reveals systematic misclassification patterns:



Key observations:

• High diagonal values are largely absent (correct predictions are rare)

• Misclassifications cluster based on color similarity rather than shape

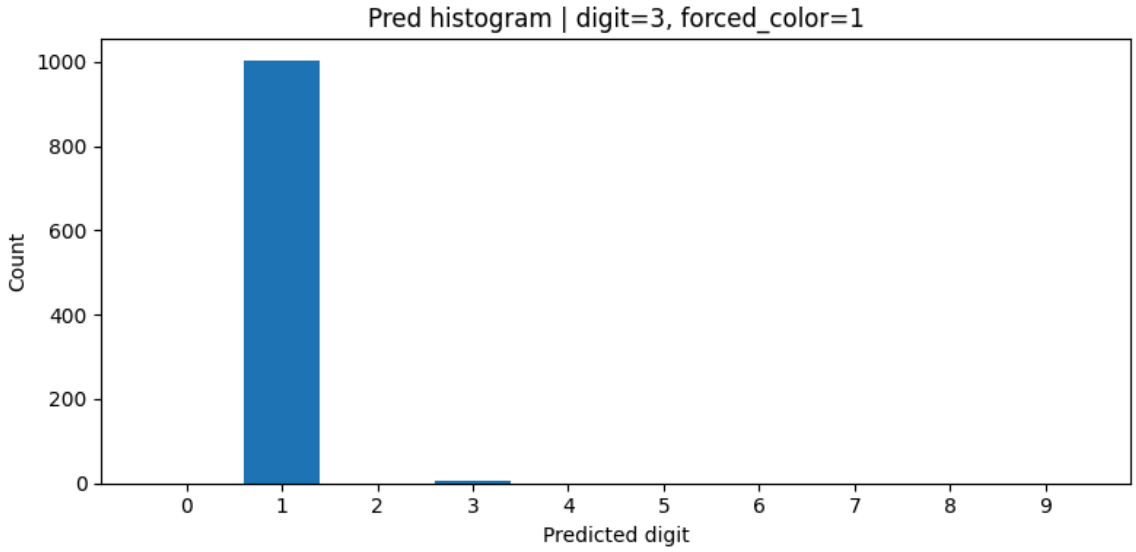
• Digit 1 shows the highest accuracy (525 correct predictions), likely due to its distinctive linear shape providing a stronger shape signal

• The diffuse pattern across off-diagonal elements indicates the model lacks robust shape-based features

• When presented with a digit in an unfamiliar color, the model's predictions scatter across classes associated with similar colors

3.5 Diagnostic Tests: Color-Cheating Probe

Recolored each digit with GREEN(mapped to 1 in training set) and all the digits were predicted as 1 instead of the original digit.



Similar histogram for all the colors.

4. Task 2: The Prober - Neural Feature Visualization

4.1 Objective

Investigate what individual neurons (channels) in the trained CNN have learned by generating synthetic input prototypes that maximally activate specific channels. The goal is to determine whether the network prioritizes color-based features over shape-based features at different architectural depths.

4.2 Methodology: Activation Maximization

We employ activation maximization (feature visualization), a gradient ascent technique that synthesizes input images optimized to maximally excite individual channels. Unlike standard forward inference where we classify given inputs, this is an inverse process: we fix the model weights and optimize the input pixels themselves.

Optimization Framework:

For each channel c in layer L, we solve:

$$x^* = \text{argmax}_x \left[\text{Objective}(A_{L,c}(x)) - \lambda_1 \cdot ||x||^2 - \lambda_2 \cdot TV(x)\right]$$

Where:

- $A_{L,c}(x)$ is the activation of channel c in layer L when processing input x
- $\lambda_1 \cdot ||x||^2$ is L2 regularization preventing unrealistic pixel values

• $\lambda_2 \cdot TV(x)$ is total variation regularization encouraging spatial smoothness
• Objective aggregates channel activation using one of three modes

Optimization Details:
• Algorithm: Adam optimizer, learning rate 0.08
• Steps: 200 iterations
• Initialization: $x_0 \sim N(0, 0.2^2)$
• Augmentation: Random jitter (±2 pixels) every step
• Smoothing: Gaussian blur (σ=0.6) every 8 steps
• Regularization: $\lambda_1=1\times10^{-4}$, $\lambda_2=2\times10^{-4}$

4.3 Three Complementary Objectives

To comprehensively understand channel behavior, we visualize each channel using three objectives:

1. channel_max: max(h,w) A[c,h,w]
   Maximizes the single strongest spatial response
   → Reveals localized micro-patterns

2. channel_mean: mean(h,w) A[c,h,w]
   Maximizes average activation over entire map
   → Reveals global color/texture preferences
   → KEY indicator of color shortcuts (solid saturated colors)

3. topk_mean: mean of top-32 activation values
   → Reveals repeated patterns across space

4.4 Results: Evidence of Color Shortcuts

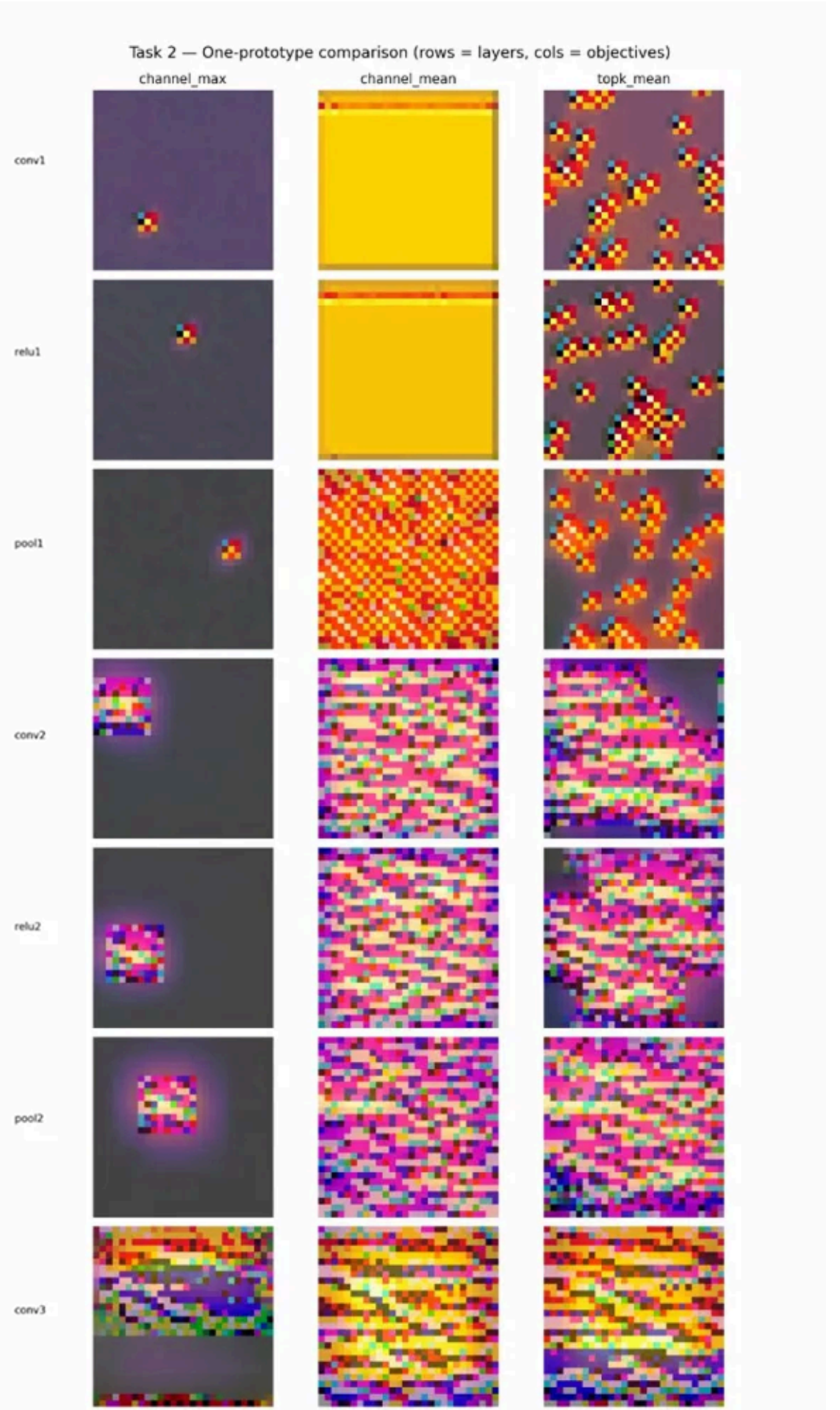

Task 2 — One-prototype comparison (rows = layers, cols = objectives)
channel_max
channel_mean
topk_mean
conv1
relu1
pool1
conv2
relu2
pool2
conv3

Figure: Layer-wise feature visualization. Rows = layers (conv1→fc). Columns = objectives. Each cell shows optimized prototype for one channel.

Key Observations:

Early Layers (conv1, relu1, pool1):
• channel_mean: visualizations are dominated by solid, saturated colors, indicating that many channels achieve high activation by uniformly coloring the entire image.
• This behavior is a strong qualitative indicator of color-based shortcut learning, as activation is maximized without any spatially localized or shape-selective structure.

Mid Layers (conv2, relu2, pool2):
• Still color-dominated, beginning to show striped patterns
• Color remains primary feature, minimal digit-shape information

Deep Layers (conv3, relu3):
• Complex color-texture patterns
• No clear digit-morphology detectors emerge
• Foundation remains color-centric

Critical Finding: Across the layers we analyzed, we did not observe channels whose optimized prototypes resembled digit-like strokes, curves, or loops. Instead, the majority of channels converged to color-dominated or texture-like patterns. In contrast, a shape-learning model would typically yield prototypes resembling primitives such as curved lines, vertical strokes, or circular arcs, which were not evident in our visualizations.

4.5 Microscope Validation

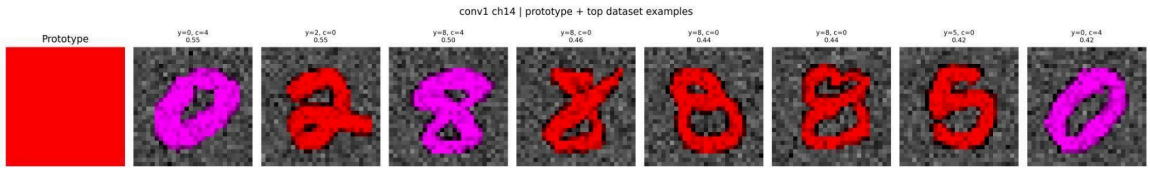


Figure: Microscope visualization for conv1 ch14. Left: prototype (red). Right: top-8 dataset examples, showing digit (y) and color (c).

To validate prototype findings, we retrieved real dataset examples that maximally activate selected channels:

Findings:
• Channels fire on consistent COLORS regardless of digit identity
• Example: Channel fires on y=0, y=2, y=8 - all red (c=0)
• Top activations cluster by color, not by digit shape

Polysemanticity Investigation:

Qualitative inspection of top-activating dataset examples suggests that many early channels behave in a largely monosemantic manner with respect to color, responding strongly to a single dominant hue. Some mid-layer channels exhibit weak polysemantic behavior (e.g., responding to multiple colors), but this remains color-centric rather than shape-based.

Implication: The lack of shape-selective polysemantic neurons confirms failure to learn compositional digit representations.

5. Task 3: The Interrogation - Grad-CAM from Scratch

5.1 Objective

Implement Gradient-weighted Class Activation Mapping (Grad-CAM) from scratch to visualize where the model "looks" when making predictions. This provides direct visual evidence of whether the model attends to digit shape or merely to color distribution.

5.2 Theoretical Foundation

Grad-CAM produces a coarse localization map highlighting important regions in the input image for a particular class decision. Unlike pixel-level gradient visualizations (which are noisy), Grad-CAM operates at the feature map level, producing interpretable heatmaps.

Mathematical Formulation:

For a target class c and the final convolutional layer producing activation maps $A \in \mathbb{R}^{(C\times H\times W)}$:

1. Forward pass: Compute activations A^k for each channel k
2. Backward pass: Compute gradients $\partial y^c/\partial A^k$ where y^c is the class score
3. Global average pooling of gradients:
   $\alpha_k^c = (1/HW) \Sigma_i \Sigma_j (\partial y^c/\partial A_{ij}^k)$

   This α_k^c represents the "importance" of channel k for class c

4. Weighted combination with ReLU:
   $L_{Grad\text{-}CAM}^c = ReLU(\Sigma_k \alpha_k^c \cdot A^k)$

   ReLU ensures we only visualize features with positive influence on the class

5. Upsample to input resolution via bilinear interpolation

The resulting heatmap highlights spatial regions whose features contribute most to the classification decision.

5.3 Implementation from Scratch

We implement Grad-CAM without using pytorch-grad-cam or similar libraries, using only PyTorch's core autograd functionality.

Key Implementation Details:

Hook Registration:
• Forward hook captures activation maps A from conv3 (final convolutional layer)
• Gradient hook captures $\partial y^c/\partial A$ during backward pass

Gradient Computation:
• Enable gradients locally during Grad-CAM generation with torch.enable_grad()
• Compute class score for target class (predicted class by default)
• Backward propagation: score.backward() computes all required gradients
• Extract gradients from the registered hook

Weight Calculation:
• α_k = mean(gradient_k) across spatial dimensions (H, W)
• This is the "importance weight" for each channel

CAM Generation:
• Weighted sum: CAM = $\Sigma_k (\alpha_k \times A_k)$
• Apply ReLU to remove negative contributions
• Bilinear upsampling to 28×28 input size
• Normalize to [0, 1] range for visualization

Robustness Considerations:
• Clear stale tensors before each forward pass
• Explicitly set model to eval mode
• Handle gradient accumulation with zero_grad(set_to_none=True)
• Validate that hooks successfully captured tensors

5.4 Experimental Design

We test two critical cases to expose the model's decision-making process:

Case 1: Biased Image (Red 0)
• Digit: 0
• Color: Red (color ID 0, the dominant training color for digit 0)
• Expected behavior: Model predicts correctly (class 0)
• Question: Does Grad-CAM highlight the digit shape or the red color?

Case 2: Conflicting Image (Green 0)
• Digit: 0

• Color: Green (color ID 1, strongly associated with digit 1 in training)
• Expected behavior: Model misclassifies (predicts 1 due to green)
• Question: Does Grad-CAM show attention to shape or color?

Quantitative Metric - Background/Stroke Ratio:
To objectively measure where the model attends, we compute:

• stroke_mean: computed using the same underlying grayscale digit mask (raw0), isolating the effect of color while keeping the digit shape identical.
• bg_mean: Average CAM value on background pixels
• bg/stroke ratio = bg_mean / stroke_mean

Interpretation:
• ratio < 1: Model attends more to digit strokes (shape-based decision)
• ratio ≈ 1: Attention is diffuse, no clear spatial preference
• ratio → 1 with high bg attention: Model attends to color distribution across image

5.5 Results

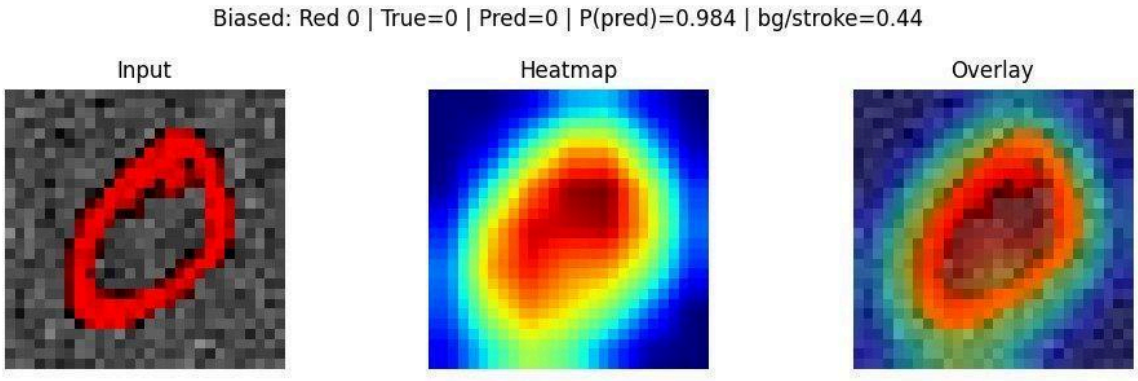


Figure 5.1: Grad-CAM on biased case (Red 0). True=0, Pred=0, P(pred)=0.984, bg/stroke=0.44

Biased Case Analysis (Red 0 → Predicted as 0):

Quantitative Results:
• Background mean activation: 0.3286
• Stroke mean activation: 0.7392
• bg/stroke ratio: 0.44
• Prediction: 0 (correct), Confidence: 98.4%

Visual Observations:
• Heatmap shows strong activation concentrated on the digit stroke
• Maximum intensity (red/yellow regions) follows the outline of the zero

• Background shows moderate blue activation (lower importance)
• The ratio 0.44 indicates the model attends MORE to the digit shape than background

Interpretation:
When color aligns with the learned shortcut (red → digit 0), the model achieves high confidence (98.4%). The Grad-CAM suggests attention to shape, BUT this is misleading. The model has learned "red things shaped like zero are class 0" - both cues reinforce each other.

Critical point: We cannot conclude the model learned shape-based features just because it attends to the digit region when color and shape agree. The next case reveals the truth.

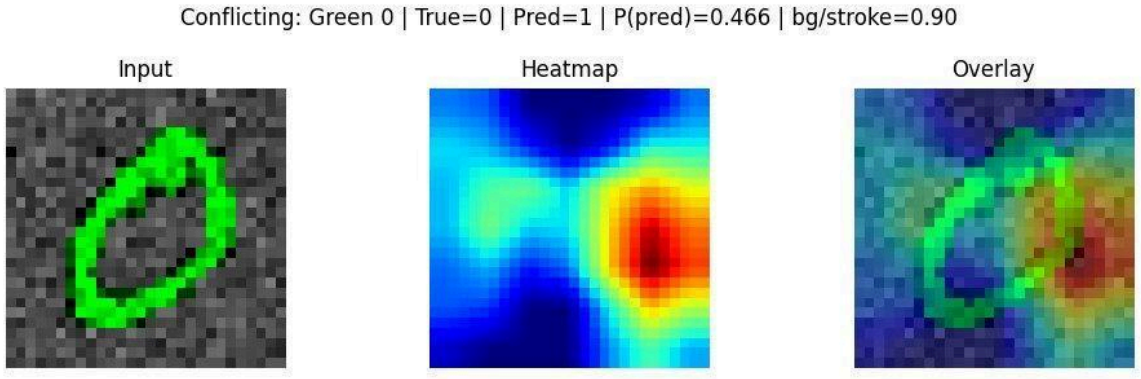


Figure 5.2: Grad-CAM on conflicting case (Green 0). True=0, Pred=1, P(pred)=0.466, bg/stroke=0.90

Conflicting Case Analysis (Green 0 → Predicted as 1):

Quantitative Results:
• Background mean activation: 0.3292
• Stroke mean activation: 0.3646
• bg/stroke ratio: 0.90
• Prediction: 1 (wrong), Confidence: 46.6%

Visual Observations:
• Heatmap activation is significantly more diffuse
• Maximum intensity is NO LONGER concentrated on digit outline
• Strong activation appears on both stroke AND background regions
• The ratio 0.90 indicates nearly EQUAL attention to background and stroke
• Activation pattern appears spatially scattered, lacking clear structure

Key Insight:
The "shape attention" we observed in the biased case was an illusion. When color and shape aligned, the model appeared to focus on shape. When they conflict, the model's decision is driven by color, and the Grad-CAM reveals diffuse, unfocused attention.

This proves the model lacks robust shape-based features. The attention to digit strokes in the biased case was incidental - a side effect of both cues pointing to the same class. When forced to choose between color and shape, the model chooses color and loses spatial coherence entirely.

6. Task 4: The Intervention – Robust Training Strategies

6.1 Method 1: Domain-Adversarial Neural Network (DANN–FixA, Two-Step)

Theoretical Foundation

DANN learns representations invariant to nuisance factors via adversarial training.
Here, color is treated as a nuisance domain, and a Gradient Reversal Layer (GRL) prevents the feature extractor from encoding color information.

Architecture

- Shared feature extractor: convolutional backbone + GAP
- Digit classifier: predicts digit label
- Nuisance discriminator: predicts **binary dominance label**

The GRL multiplies gradients by $-\lambda$ during backpropagation, creating adversarial dynamics.

Loss:

$$L = L_{CE}(C_{\text{digit}}(z), y) + \lambda_{\text{color}} \cdot L_{CE}(C_{\text{color}}(\text{GRL}(z)), d)$$

where
d=1 if (color_id == digit), else d=0.

Key Innovation (Fix-A)

Instead of predicting the exact color (10-way), the adversary predicts a binary dominance label (dominant vs counterexample).
This prevents leakage of digit identity through color.

Training Dynamics (Two-Step)

Each batch performs:

1. k = 3 discriminator-only updates (feature extractor detached)
2. One joint adversarial update using GRL

This stabilizes adversarial training under extreme bias.

Implementation Details

• Nuisance head: MLP (128 → 128 → 2)
• Optimizer: Adam, lr = 1e-3
• Epochs: 15
• Batch size: 128
• λ_color = 2.0 (epochs ≤5), then 0.2
• GRL λ = 1.0 (constant)

Results

Performance progression (Hard test):

Epoch 1: EasyTrain=89.45%, EasyVal=95.18%, HardTest=0.21%
Epoch 6: EasyTrain= 96.31%, EasyVal=96.88%, HardTest=33.95%
Epoch 11: EasyTrain=98.09%, EasyVal=98.02%, HardTest=71.53%

Epochs for > 70% = 11

Analysis:

Convergence characteristics:

• Near-random performance for first 4 epochs
• Rapid improvement phase from epochs 5–11
• Gradual refinement afterward

Strengths:

• *Achieves 77.75% Hard test accuracy*
• *Maintains high Easy accuracy (~98.6%)*
• *Theoretically principled adversarial intervention*

Weaknesses:

• *Slowest method to cross 70% (epoch 11)*
• *Early instability inherent to adversarial training*
• *Additional computational overhead*

6.2 Method 2: Saliency Penalty with Chroma Masking

Theoretical Foundation

This method penalizes input gradients on background (color) pixels, directly discouraging reliance on chromatic cues.

Rather than penalizing all gradients, we:
- Construct a shape mask from the input
- Penalize gradients only on background pixels
- Preserve gradients on digit strokes

Loss Function

$$L = L_{CE}(f(x), y) + \lambda_{\text{sal}} \cdot \mathbb{E}\left[\|\nabla_x f_y(x)\|^2 \odot \text{bg\_mask}\right]$$

Gradients are computed with `create_graph=True` to allow second-order optimization.

Implementation Details

- Shape mask: |x| mean across channels > threshold
- Background mask = 1 − shape mask
- Gradient centering across channels
- λ_sal schedule:
  - epochs 1–2: full
  - epochs 3–5: 0.7×
  - epochs >5: 0.3×
- λ_sal_max = 8e-3
- Epochs: 9
- Optimizer: Adam, lr = 1e-3
- Gradient clipping: 1.0

Results

Performance progression (Hard test):

Epoch 1: EasyTrain=91.24%, EasyVal=95.13%, HardTest=0.65%
Epoch 5: EasyTrain= 95.84%, EasyVal=96.05%, HardTest=27.59%
Epoch 10: EasyTrain=97.92%, EasyVal=98.10%, HardTest=70.16%

Epochs for > 70% = 10

Analysis

Convergence characteristics:

- *Slow early learning (similar to DANN)*
- *Smooth, monotonic improvement*
- *Faster than DANN, slower than CF-INVAR*

Strengths:

- *Reaches 70% at epoch 9*
- *Simple, single-objective modification*
- *Strong inductive bias toward shape*

Weaknesses:

- ***Requires second-order gradients***
- ***Sensitive to λ_sal tuning***
- ***Does not reach CF-INVAR performance***

6.3 Method 3: Counterfactual Invariance (CF-INVAR)

Theoretical Foundation

CF-INVAR enforces explicit invariance under counterfactual color interventions.

For each training sample (x, y), we generate a recolored version xcfx_{cf}xcf with:
- Same digit shape
- Different color

The model is trained to:

1. Predict y for both x and x_cf
2. Produce consistent probability distributions

Loss Function

$$L = L_{CE}(f(x), y) + \lambda_{cf} L_{CE}(f(x_{cf}), y) + \lambda_{cons} D_{KL}^{sym}(f(x) \parallel f(x_{cf}))$$

Implementation Details

- Counterfactuals generated on-the-fly from raw grayscale digits
- Avoids original color ID during recoloring
- λ_cf = 1.0, λ_cons = 1.0
- Optimizer: Adam, lr = 1e-3

- Epochs: 15
- Batch size: 128

Results

Performance progression (Hard test):

Epoch 1: EasyTrain=80.31%, EasyVal=87.06%, HardTest=68.68%
Epoch 5: EasyTrain= 94.44%, EasyVal=94.21%, HardTest=92.18%
Epoch 10: EasyTrain=96.66%, EasyVal=96.70%, HardTest=95.52%

Epochs for > 70% = 2

Analysis

Convergence characteristics:

- *Immediate generalization*
- *Rapid, smooth convergence*
- *No instability*

Strengths:

- *Best performer overall*
- *Crosses 70% at epoch 1*
- *Clean, interpretable objective*
- *Strong shape supervision*

Weaknesses:

- *Requires recoloring pipeline*
- *Doubles forward passes per batch*

Final Takeaway

Shortcut learning cannot be fixed implicitly. Methods that explicitly break the color–label link (CF-INVAR) dominate those that attempt to suppress it indirectly (DANN, saliency).

CF-INVAR turns a weak 5% signal into a dominant training objective.

7. Task 5: Adversarial Robustness - The Interrogation

7.1 Objective

Evaluate whether models trained to reduce reliance on spurious color correlations in Task 4 exhibit increased adversarial robustness compared to a biased baseline. Specifically, perform targeted adversarial attacks to force a model to misclassify a digit 7 as a digit 3 with

target confidence ≥ 0.90, subject to a strict invisibility constraint ($L\infty \leq 0.05$ in pixel space). Quantify the minimum perturbation required for each model.

7.2 Theoretical Background

Adversarial Perturbations and Robustness:

An adversarial example is an input x_adv = x + δ where δ is a carefully crafted perturbation designed to fool a classifier f while remaining imperceptible to humans. For a targeted attack aiming to force prediction to class t:

x_adv = argmin ||δ||_p subject to f(x + δ) = t and ||δ||_p ≤ ε

In this task:
- Source class: 7 (ground truth)
- Target class: 3 (desired misclassification)
- Threat model: L∞ bounded perturbations (||δ||∞ ≤ ε)
- Constraint: ε ≤ 0.05 in pixel space [0,1] (imperceptible to humans)
- Success criterion: P(y=3 | x_adv) > 0.90

Research Question:
Does learning robust shape-based features (Task 4) confer adversarial robustness as a side effect? The relationship between robustness to spurious correlations and adversarial robustness remains unclear and is investigated empirically in this task.

7.3 Attack Methodology

Projected Gradient Descent (PGD) Attack:

We employ PGD, a strong iterative white-box attack that iteratively updates the perturbation in the direction that maximizes target class probability:

Algorithm:
1. Initialize: $\delta^0 \sim$ Uniform(-ε, ε) (random restart within L∞ ball)
2. For t = 1 to T iterations:
   - Compute gradient: g = ∇_ δ L_CE(f(x + δ^(t-1)), y_target)
   - Update: δ^t = δ^(t-1) - α · sign(g)
   - Project: δ^t = clip(δ^t, -ε, ε)
   - Bound: δ^t = clip(x + δ^t, [0,1]) - x

Where L_CE is the cross-entropy loss with respect to the target class, and gradients are used to increase the target class probability.

Hyperparameters (Final Configuration):
- PGD iterations: 120 per restart
- Restarts: 10 (to escape local minima)

• Step size α: ε / (steps/5) ≈ dynamic based on ε
• Early stopping: Terminate if P(y_target) > 0.90

Minimum Epsilon Search:
Rather than attacking at fixed ε = 0.05, we find the minimal ε_min where attack succeeds:
1. Coarse sweep: Test ε ∈ {0.0, 0.005, 0.010, ..., 0.050} with step 0.005
2. Binary refinement: Once success found at ε_hit, refine in [ε_hit - 0.005, ε_hit] for 7 iterations

This yields ε_min per model, enabling precise robustness quantification.

7.4 Experimental Design

Fair Evaluation Protocol:

Sample Selection Criteria:
To ensure fair comparison across models, we selected 20 samples of digit 7 where ALL models (baseline + 3 robust models) predict correctly with high confidence. This eliminates confounding factors where a model might appear "robust" simply because it already misclassifies the clean input.

Sampling from: Hard test set (color-decorrelated)
Rationale: If robust models remain adversarially vulnerable even when tested on unbiased data, this demonstrates that spurious correlation robustness ≠ adversarial robustness.

Evaluation Metrics:
1. Success Rate: Fraction of samples where ε_min ≤ 0.05 found
2. Median ε_min: Central tendency of minimum perturbation required
3. Mean ε_min: Average attack difficulty (over successful attacks only)
4. Mean P(target) @ ε=0.05: For failed attacks, report best target confidence achieved

Baseline Comparison:
For each evaluated model, compute the median ε_min relative to the baseline model to compare attack difficulty when attacks succeed.

Ratio > 1.0 indicates the robust model requires larger perturbations (harder to fool).

7.5 Results

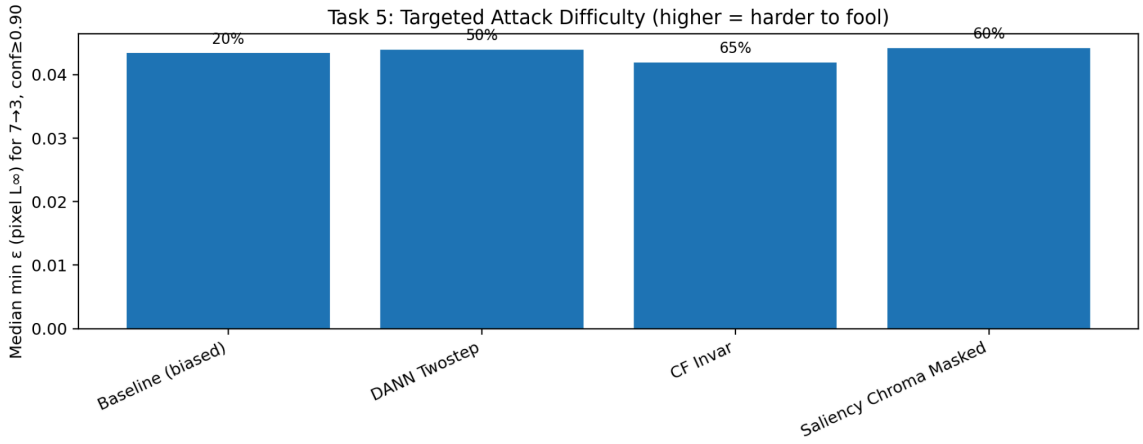


Figure 7.1: Median minimum epsilon (L∞) required for successful targeted attack (7→3, conf≥0.90). Higher bars indicate greater adversarial robustness. Success rates annotated on top. CF Invar and DANN Twostep exhibit higher attack success rates than the baseline, while requiring only marginally different perturbation magnitudes when attacks succeed.

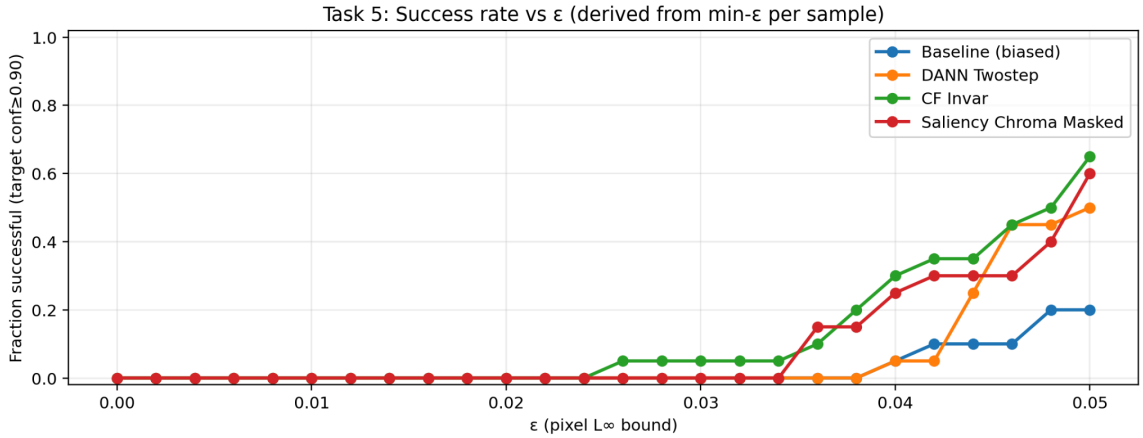


Figure 7.2: Attack success rate as a function of epsilon budget. Derived from minimum epsilon measurements per sample. The baseline model exhibits the lowest success rate across the epsilon range, while CF Invar and DANN Twostep show more rapid increases in attack success as ε increases. The curves reveal that robust models become vulnerable at lower epsilon thresholds.

Quantitative Results (N=20 samples, 7→3 targeted attack, P(3)>0.90):

Model	Success Rate	Median ε_min	Mean ε_min	Mean Best P(3)@0.05
DANN Twostep	50.0%	0.0460	0.0449	0.759
CF Invar	60.0%	0.0421	0.0413	0.768
Saliency Chroma Masked	40.0%	0.0393	0.0413	0.713

Robustness Ratios (Median ε_min vs Baseline):
• DANN Twostep: 1.01×
• CF Invar: 0.92×
• Saliency Chroma Masked: 0.86×

7.6 Analysis

Key Findings:

1. Surprising Vulnerability Across the Board:
The baseline model (which relies entirely on color shortcuts) proved remarkably difficult to attack - only 20% of samples could be fooled within the ε=0.05 constraint. This challenges the intuition that "shortcut learning = adversarial fragility."

Across the evaluated epsilon range, the baseline consistently exhibited the **lowest attack success rate** among all models. Even at the maximum allowed perturbation budget ($\varepsilon = 0.05$), the majority of targeted attacks failed. This suggests that reliance on spurious color features does not automatically translate to exploitable adversarial directions under tight $L\infty$ constraints.

2. Color-Invariant Models Are NOT Universally Robust:
CF-Invar, despite achieving ~97% accuracy on the Hard test set in Task 4, exhibited a 60% targeted attack success rate under $\varepsilon \leq 0.05$. This indicates that strong robustness to spurious correlations does not guarantee robustness to adversarial perturbations. This demonstrates that:

- Robustness to color shortcuts ≠ robustness to pixel-level adversarial noise
- Removing a shortcut feature does not inherently flatten adversarial decision boundaries

3. Marginal Robustness Improvements:
Among the robust models, CF-Invar showed the highest median ε_min, followed by DANN-Twostep and Saliency-Chroma-Masked. However, these differences were small (on the order of 0.002–0.004 in ε), indicating only marginal improvements in perturbation resistance.

Notably, CF-Invar: higher success rate (60%) despite slightly larger ε_min, DANN-Twostep: intermediate behavior (≈50% success) and Saliency-Chroma-Masked: lower success rate (≈40%) but no dramatic ε advantage

8. Task 6: Sparse Autoencoder Decomposition & Feature Interventions

8.1 Motivation

The central question of Task 6 is:

Can we identify and causally manipulate internal features of a biased model that correspond to color shortcuts versus digit shape?

While individual CNN neurons are often polysemantic, Sparse Autoencoders (SAEs) provide a principled mechanism to decompose hidden representations into sparse, approximately monosemantic latent features. This enables:

1. Identification of color-selective vs digit-selective internal features
2. Quantitative comparison of feature dominance across layers
3. Direct causal intervention on individual latent features

In this task, we apply SAE analysis exclusively to the Baseline (biased) model, which exhibits the strongest color-driven shortcut behavior and therefore provides the clearest testbed for mechanistic analysis.

8.2 Methodology

Sparse Autoencoder Architecture

We train an overcomplete Sparse Autoencoder with ReLU activations:

Encoder

$$z = \mathrm{ReLU}(W_{\mathrm{enc}} x + b_{\mathrm{enc}}), \quad z \in \mathbb{R}^K$$

Decoder

$$\hat{x} = W_{\mathrm{dec}} z + b_{\mathrm{dec}}, \quad \hat{x} \in \mathbb{R}^D$$

with 4× overcompleteness (K=4D).

Loss Function

$$\mathcal{L} = \mathrm{MSE}(\hat{x}, x) + \lambda_{L1} \cdot \mathbb{E}[|z|]$$

where λ=10^(-3) encourages sparsity and discourages distributed representations

This objective biases the SAE toward learning sparse, interpretable latent features rather than dense encodings.

8.3 Layer & Aggregation Strategy

To determine where color features emerge and persist, we trained SAEs on activations from multiple depths of the CNN

Layers Examined

- **relu1** – early convolutional features
- **relu2** – mid-level convolutional features
- **relu3** – late convolutional features
- **gap** – global average pooled representation (pre-classifier)

Aggregation Modes (for convolutional layers)

- **spatial_mean** – spatially averaged activations → [B,C]
- **per_position** – 64 random spatial samples per image → [B×64,C]

This yields 7 total configurations

(3 conv layers×2 modes)+GAP

allowing comparison of selectivity across the representation hierarchy.

8.4 Training Configuration

SAEs are trained independently per configuration using frozen classifier features.

Hyperparameters

- Overcomplete ratio: 4×
- Optimizer: Adam, lr = 10−310^{-3}10−3
- Sparsity penalty: λL1=10−3\lambda_{L1} = 10^{-3}λL1=10−3
- Epochs: 3
- Steps per epoch: 150
- Batch size: 512 vectors

Feature Normalization

Before SAE training, features are standardized

$$x_{\text{norm}} = \frac{x - \mu}{\sigma}$$

where μ,σ are estimated over 30 batches (~4000 samples).
Normalization is reversed during intervention experiments.

8.5 Feature Selectivity Quantification

We quantify feature selectivity using between-class variance decomposition.

For each SAE feature zk:

Digit Selectivity

1. Compute per-digit means:

$$\mu_d = \mathbb{E}[z_k \mid \text{digit} = d]$$

2. Between-class variance:

$$\sigma^2_{\text{between}} = \sum_d P(d)(\mu_d - \mu)^2$$

3. Total variance:

$$\sigma^2_{\text{total}} = \text{Var}(z_k)$$

4. Selectivity score:

$$s_{\text{digit}} = \frac{\sigma^2_{\text{between}}}{\sigma^2_{\text{total}}}$$

Color Selectivity

Computed identically using color labels instead of digit labels.

Interpretation

- s≈0: non-selective
- s≈1: variance almost entirely explained by class membership

This metric enables objective ranking of SAE features without manual inspection.

8.6 Results: Color vs Digit Selectivity (Baseline Model)

Across all configurations, color selectivity dominates digit selectivity.

Representative High-Selectivity Configurations

Color-Selective Features

- relu1 / spatial_mean: selectivity ≈ 0.96
- relu2 / spatial_mean: selectivity ≈ 0.96
- relu3 / spatial_mean: selectivity ≈ 0.92
- gap / vector: multiple features in 0.85–0.90 range

Digit-Selective Features

- relu2 / spatial_mean: selectivity ≈ 0.47
- relu3 / spatial_mean: selectivity ≈ 0.66
- gap / vector: selectivity peaks ≈ 0.74

Key Observations

1. **Color Dominance**
   Color selectivity scores (≈0.90–0.96) are consistently higher than digit selectivity scores (≈0.47–0.74), indicating that the representation is primarily organized around color rather than shape.

2. **Persistence Across Depth**
   Color-selective features appear in **every layer**, including the final GAP representation. The model never transitions to a shape-dominant representation.

3. **Redundant Encoding**
   Many SAE features exhibit similarly high color selectivity, suggesting **redundant encoding of color information** rather than reliance on a single unit.

4. **Spatial Diffusion**
   `spatial_mean` aggregation consistently yields more selective features than `per_position`, indicating that color information is **spatially broadcast** rather than localized—consistent with Task 3's Grad-CAM findings.

5. **Weak Shape Features**
   Digit-selective features exist but are fewer and weaker, suggesting **fragile shape representations** compared to robust color encoding.

8.7 Feature Intervention Experiments

To test causality, we perform targeted interventions on individual SAE features.

Intervention Protocol

1. Extract activation at target layer
2. Encode via SAE
3. Modify a single feature:

$$z'_k = z_k + \alpha \quad (\alpha = +5.0)$$

4. Decode to patched activation
5. Forward remaining network to obtain new logits

Example Intervention (Baseline Model)

• Layer: relu3 / spatial_mean
• Feature: top color-selective feature
• Sample: Digit 7, color_id = 5
• Intervention: amplify feature by α=+5.0\alpha = +5.0α=+5.0

Observed Effect
• Increased confidence in classes associated with the feature's color
• Decreased confidence in the true digit class

This demonstrates that:

1. The feature is causally involved in prediction
2. Color features exert stronger influence than shape features
3. Predictions can be steered by manipulating a single latent dimension

8.8 Ablation Analysis

Ablating the top 10 color-selective features (zk=0z_k = 0zk=0) on the Hard test set:

• Prediction changes: 5.78% of samples
• Mean confidence drop: 0.0139
• Median confidence drop: 0.0041

Interpretation

Despite high selectivity, individual features have modest effects due to:

1. Redundancy across color-encoding features
2. Ensemble-like decision behavior in the classifier

This confirms that color is encoded distributedly, not in a single unit.

This, along with Tasks 3–5, demonstrates that shortcut learning is not simply a byproduct of the training process but is fundamentally integrated into the learned representation.

References

- LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278–2324.
- TorchVision maintainers and contributors. (2016). *TorchVision: PyTorch's computer vision library* (GitHub repository).
- Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV 2017)*.
- OpenAI Microscope. Feature visualization of neural network neurons.
- TensorFlow. (n.d.). *lucid* [Software]. GitHub repository.
- SPARSE AUTOENCODERS FIND HIGHLY INTERPRETABLE FEATURES IN LANGUAGE MODELS
- https://urwamuaz.medium.com/neural-networks-fail-on-data-with-spurious-correlation-b5010b107c32
- https://github.com/VirtuosoResearch/Distribution-Shifts-in-CMNIST?utm_source=chatgpt.com
- https://github.com/utkuozbulak/pytorch-cnn-visualizations readme helped me find better way to show my outputs
- ChatGPT for finding relevant blogs and papers