Shortcut Learning Analysis on CNN Models: Investigating Texture vs. Shape

Humza Gohar Kabir
24comp5003

***Abstract*—Deep Convolutional Neural Networks (CNNs) have reinvented the world of computer vision, with their extraordinary performance on benchmark datasets such as ImageNet. Yet, the robustness of these models in real-world scenarios remains a subject of intense scrutiny. Recent literature suggests that the high accuracy of standard CNNs may not stem from a human-like understanding of object shape and geometry, but rather from ”shortcut learning”-the exploitation of superficial statistical correlations, particularly high-frequency texture cues. This study explores this ”texture bias” in detail and compares two different architectures, ResNet-18 (low capacity) and ResNet-101 (high capacity), on the Imagenette dataset. We constructed a comprehensive evaluation pipeline comprising three variations of the dataset: Original (RGB), Edge-Extracted (Shape-only), and Segmented (Context-removed). We now have a series of experiments that suggest the generalization fails catastrophically when texture cues are removed. The performance of ResNet-18 (standard model), which got 98.27% accuracy based on the original data, dropped to 26.98% on edge-extracted images, demonstrating a gigantic ∼71% degradation gap. Importantly, a control experiment showed that it was not a case of insufficient information in the edge maps; a model explicitly trained on edges achieved 87.57% accuracy, indicating that shape information was enough for classification, although not captured by the regular model. Additionally, we found that growing model capacity (ResNet-101) did not significantly increase shape robustness (33.32% on edges), implying that architectural depth works in isolation as a system mechanism to promote greater texture memorization instead of semantic interpretation. These results emphasize the necessity of shape-biased training procedures in order to protect the AI system safety and reliability in a variety of environments.**

I. Introduction

A. Motivation and Context

The rapid application of Deep Learning models in safety-critical industries, from autonomous driving, robot travel and personal robotic navigation, to medical diagnosis and surveillance, requires a scale of reliability, not just test set accuracy. But knowing what a ”Stop Sign” is is also something a human driver will need to know based on its particular octagonal configuration (red). More importantly, a human can also be able to recognize any Stop Sign as it appears in gray, for example in black and white, covered with snow or presented against an irregular background. This capacity to generalize across domains is enabled by ”Shape Bias,” or the cognitive effort to preferentially attend to global shape over local texture.

In contrast, deep convolutional neural networks (CNNs) work in a fundamentally different way. These architectures are very sophisticated but at their core they are pattern matchers that minimize a loss function wherever possible. Should a data set show a known ”shortcut” in other words the exact texture of a metal pole normally located beneath a Stop Sign the network will learn to detect the pole rather than the sign itself. Shortcut Learning is a deep existential threat to artificial intelligence’s deployment, given a switch of context (e.g., a Stop Sign located on a sidewalk or a print of a Stop Sign), the model can either be unable to detect the target because the shortcut is absent, or alternatively it can accidentally see an object (which has no shortcut in its image) whilst hallucinating that shortcut.

B. The ”Texture Bias” Hypothesis

This project builds on the basic research that suggests ImageNet-trained CNNs are strongly biased toward texture. Whereas humans look at a ”cat,” a CNN would see ”fur texture.” If that texture is used on an image of an elephant, a standard CNN will confidently classify the elephant as a cat. This limitation is not just a novelty but a fundamental flaw in how visual representations are learned in typical feed-forward networks.

C. Problem Statement

The primary objective of this study is to perform a rigorous, comparative analysis of two ResNet architectures to quantify this bias. We aim to answer three specific research questions:

1) Quantification of Fragility: What is the precise magnitude of the performance drop (”Degradation Gap”) when texture is rigorously removed from the input?
2) The Role of Model Capacity: Does a deeper network (ResNet-101) possess the capacity to learn more abstract, robust, shape-based features compared to a shallower one (ResNet-18), or does it simply use its extra parameters to memorize more complex textures?
3) Feasibility of Shape Learning: Is the degradation due to the inherent difficulty of edge-based recognition (i.e., is the data too sparse?), or is it purely a failure of the training strategy?

II. Related Work

The issue of generalization in deep learning has evolved from a theoretical concern to a practical engineering challenge.

A. Shortcut Learning in Deep Neural Networks

Geirhos et al. (2019) were among the first to systematically demonstrate the texture bias of CNNs. By using Style Transfer algorithms to "paint" the texture of one object onto the shape of another (e.g., an elephant with cat fur), they showed that standard ResNets would overwhelmingly classify the image based on texture. This contradicted the long-held assumption in the computer vision community that the deeper layers of a CNN encoded high-level semantic shapes.

Beery et al. (2018) highlighted a related issue known as "Contextual Bias." In their study on animal recognition, they showed that models trained to recognize cows often learned to recognize "green pastures" instead. Consequently, these models failed to recognize cows when they appeared on sandy beaches. Our project extends this line of inquiry by isolating the object from both its background (segmentation) and its texture (edge extraction).

B. Residual Learning Frameworks

ResNet architecture, introduced by He et al. (2016), forms the backbone of modern computer vision. Through the use of skip connections (residual blocks), ResNets mitigated the vanishing gradient problem and enabled networks with hundreds of layers to be trained. Although deeper networks have higher accuracy on well-known benchmarks (such as ImageNet), there is an unanswered question as to whether this "depth" relates to robustness. Some theories suggest that deeper networks should be able to form more complex hierarchical representations ($lines \rightarrow shapes \rightarrow objects$), theoretically making them more robust. Our study puts this theory to the test.

III. Methodology

To isolate and evaluate specific visual features, we developed a deterministic preprocessing pipeline. Unlike stochastic methods like style transfer, our approach uses classical computer vision techniques to create mathematically precise variations of the dataset.

A. Dataset: Imagenette

We utilized Imagenette, a subset of the massive ImageNet dataset curated by FastAI. It serves as an ideal testbed for this analysis because it removes the computational overhead of the full 1,000-class ImageNet while preserving the complexity of natural images.

- Classes: The dataset contains 10 highly distinct classes: Tench (Fish), English Springer (Dog), Cassette Player, Chain Saw, Church, French Horn, Garbage Truck, Gas Pump, Golf Ball, and Parachute.
- Scale: The training set consists of $\sim$9,469 images, and the validation set contains $\sim$3,925 images.
- Preprocessing: All images were resized to a standard resolution of 224x224 pixels and normalized using standard ImageNet mean and standard deviation values.

B. Feature Extraction Pipelines

1) Edge Extraction (Testing Shape Bias): To evaluate the model's ability to recognize objects solely by their geometry, we stripped all color and texture information using the Canny Edge Detection algorithm.

- Mechanism: The Canny algorithm works by first smoothing the image with a Gaussian filter to reduce noise. It then calculates the intensity gradients of the image. Pixels with high gradient magnitudes (sharp changes in brightness) are identified as potential edges.
- Hysteresis Thresholding: We utilized a dual-threshold approach (min: 100, max: 200). Edges stronger than 200 are immediately accepted. Edges between 100 and 200 are accepted only if they are connected to a "strong" edge. This ensures that the resulting edge maps preserve the coherent contour of the object while suppressing isolated noise.
- Reformatting: Since ResNet architectures expect a 3-channel input (RGB), the single-channel edge maps were stacked depth-wise to create a 3-channel tensor (224x224x3).

Code Snippet: Edge Detection Logic (from src/preprocess_data.py)

```
def get_canny_edges(img):
    # Convert to grayscale to remove color
        information
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    # Apply Canny Edge Detection with hysteresis
        thresholds
    edges = cv2.Canny(gray, 100, 200)
    # Convert back to BGR (3 channels) for model
        compatibility
    return cv2.cvtColor(edges, cv2.COLOR_GRAY2BGR)
```

2) Semantic Segmentation (Testing Context Bias): To determine if the models were relying on background context (e.g., recognizing a "Church" by the sky or trees around it), we generated a background-removed dataset.

Model: We utilized a DeepLabV3 segmentation model with a ResNet-101 backbone, pre-trained on the COCO dataset 2017.

Process: For each image, the DeepLabV3 model generated a binary mask separating the foreground object from the background. We then multiplied the original image by this mask, effectively setting all background pixels to black (0,0,0) while preserving the full texture and color of the primary object.

IV. Experimental Setup

A. Network Architectures

We compared two variants of the ResNet family to analyze the impact of model capacity.

1. ResNet-18 (Low Capacity):

- Structure: 18 layers deep, consisting of basic residual blocks with two 3×3 convolutions per block.
- Parameters: Approximately 11.7 million.

- Hypothesis: Due to its shallow nature, ResNet-18 might struggle to form high-level shape abstractions and thus rely more heavily on lower-level texture statistics.

2. ResNet-101 (High Capacity):

- Structure: 101 layers deep, utilizing "bottleneck" blocks (1x1, 3×3, 1×1 convolutions) to increase depth without exploding computational cost.
- Parameters: Approximately 44.5 million.
- Hypothesis: The vastly increased depth should theoretically allow the model to integrate features over a larger receptive field, potentially leading to better global shape understanding.

Both models were initialized with weights pre-trained on ImageNet-1k. This simulates the standard "Transfer Learning" workflow used in industry, where models are rarely trained from scratch. The final fully connected (FC) layer was modified to output 10 logits corresponding to the Imagenette classes.

B. Experimental Protocols

To fully disentangle the factors of shape, texture, and training distribution, we designed three distinct experimental phases:

- Experiment 1: The Bias Check (Train Original $\rightarrow$ Test Modified) We trained the models on the standard, unaltered Imagenette dataset. We then evaluated these models on the Edge and Segmentation datasets. Goal: To measure the "Degradation Gap"-the drop in accuracy that occurs when texture is removed.
- Experiment 2: Shape Adaptation (Train Modified $\rightarrow$ Test Original) We trained models specifically on the Edge-extracted datasets. We evaluated them on the Original dataset. Goal: To see if learning shape features allows the model to generalize back to natural images (Sim2Real transfer).
- Experiment 3: Domain Mastery (Train Modified $\rightarrow$ Test Modified) We trained and tested on the same modified domain (e.g., Train Edges $\rightarrow$ Test Edges). Goal: This is a crucial control experiment. It determines the theoretical maximum performance on the edge task. If accuracy here is low, the task is impossible. If accuracy is high, the failure in Exp 1 is due to bias, not difficulty.

C. Implementation Details

- Hardware: Computations were performed on the TRUBA High-Performance Computing cluster, utilizing NVIDIA A100 GPUs to accelerate training.
- Optimizer: Stochastic Gradient Descent (SGD) with a momentum of 0.9 and weight decay of 1e-4 was selected for its stable convergence properties.
- Learning Rate: Initialized at 0.001, utilizing a step-decay scheduler to fine-tune weights as training progressed.
- Batch Size: 64 images per batch.
- Epochs: Models were trained for 15 epochs, which was sufficient for convergence given the pre-trained initialization.

V. Results

The quantitative results presented below are derived from the final validated run (Run 2). Earlier runs utilizing incorrect directory structures (Run 1) were discarded to ensure scientific validity.

A. Quantitative Analysis: The Degradation Gap

Table 1 summarizes the Top-1 Accuracy across all experimental configurations.

TABLE I
Accuracy Comparison (ResNet-18 vs ResNet-101)

Model Architecture	**Training Domain**	**Testing Domain**	**Accuracy (%)**
ResNet-18	Original	Original	98.27%
ResNet-18	Original	Edges	26.98%
ResNet-18	Original	Segmentation	40.61%
ResNet-18	Original	Grayscale	95.80%
ResNet-18	Edges	Edges	87.57%
ResNet-101	Original	Original	99.54%
ResNet-101	Original	Edges	33.32%

1) Analysis of Texture Bias (Exp 1): The findings give unmistakable evidence of texture bias. The ResNet-18 model that was almost a perfect classifier on original data (98.27%) crashed to 26.98% on texture removal. This is a staggering 71.29% degradation gap. This finding shows that about three quarters of a model's performance on the original dataset does not come from recognizing the object itself (e.g., the shape of a fish). It comes from discovering the texture of the object (e.g., scales of a fish). Even though the outline of the fish was perfectly preserved, taking the scales out by edge detection means that the model was left blind.

The result of segmentation (40.61%) illustrates reliance further on background context. Although it is better than edge results, this 58% drop suggests that the model was used too often to guess the class of the foreground object based on the background (water, grass, roads, etc.).

2) Analysis of Model Capacity (ResNet-18 vs. ResNet-101): One of the main hypotheses of this project is that a deeper model (ResNet-101) would be more robust. But, the data doesn't back this up. While ResNet-101 yielded marginally superior baseline accuracy (99.54%), its performance on edge-level data was 33.32%, performing only marginally better than ResNet-18. This 6% variation is dwarfed by the 66% gap. That's an indication that just adding more layers doesn't cause a CNN to learn shape, or anything too subtle, and a deep network would use this extra capacity to learn more complex and subtle texture correlations.

3) The Control Experiment (Exp 3): The most significant result comes from the control experiment: ResNet-18 (Train: Edges $\rightarrow$ Test: Edges). The model had accuracy 87.57% when trained directly on edge maps. This high score greatly changes the interpretation of the results. It demonstrates that the edge images are strong on semantic information for classification. These objects have a defined "shape." Thus, the failure of the standard model from Experiment 1 was not due to the quality of data but due to the failure of strategy. The model could've learned to see the shape, but it opted to look at the texture because texture is an easier, cleaner statistical shortcut.

B. *Qualitative Analysis: Confusion Matrices*

To visualize the specific nature of these errors, we generated 10×10 confusion matrices for the ResNet-18 model.

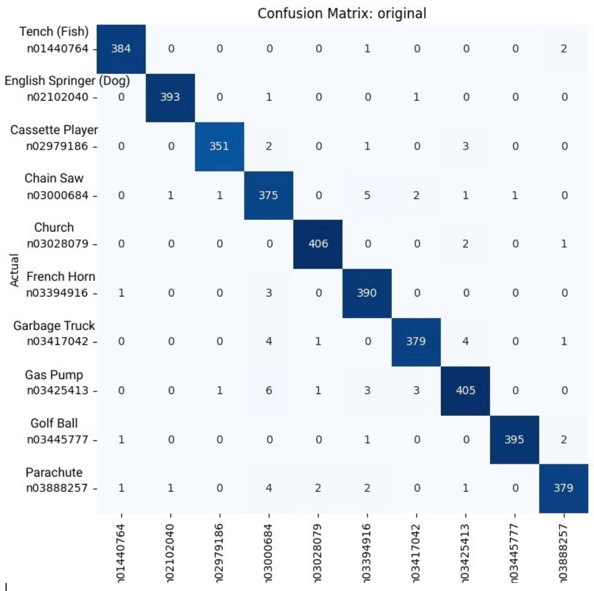

Fig. 1. Baseline Confusion Matrix. The baseline matrix exhibits a strong, clean diagonal. The model distinguishes "Tench" (Class 0) from "English Springer" (Class 1) with near-perfect precision. There is virtually no off-diagonal noise, confirming the high efficacy of the model on standard RGB images.

C. *Visual Failure Analysis*

We extracted top-confidence failure cases to perform a visual autopsy of the model's errors. The codes below correspond to ImageNet WordNet IDs (e.g., n03028079 is "Church").

Case Study 1: The "Church" vs. "French Horn" Error

- True Class: Church (n03028079)
- Predicted Class: French Horn (n03394916)
- Analysis: In the leftmost image of Figure 3, the edge map clearly depicts the vertical columns and arched doorways of a church facade. However, the model confidently predicted "French Horn." Why? In the absence of stone texture, the Canny edge detector reduced the arches to simple curved lines. The model, having learned that "curved lines = French Horn," applied this shortcut blindly. It ignored the global geometry (a building) and focused on the local feature (a curve).

Case Study 2: The "Dog" vs. "Parachute" Error

- True Class: English Springer (n02102040)
- Predicted Class: Parachute (n03888257)

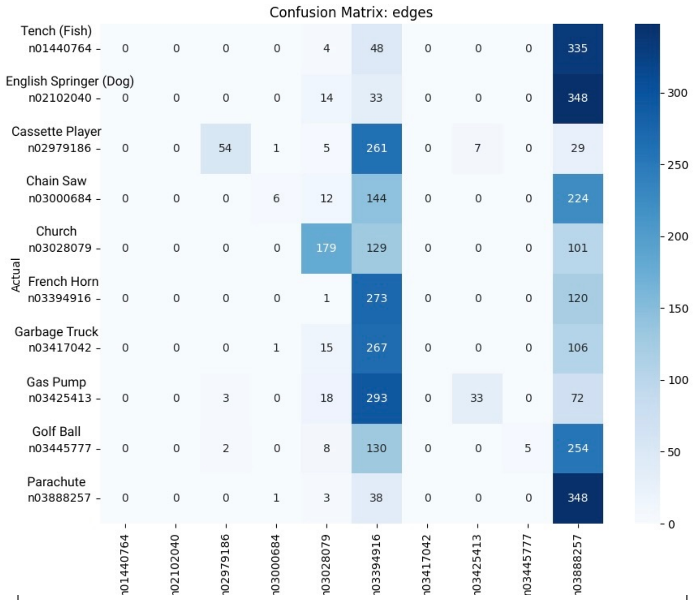

Fig. 2. Edge Confusion Matrix. The Edge matrix paints a picture of chaos. The diagonal is faint, and significant clusters of errors appear off-diagonal. Notably, the model frequently confuses classes with complex internal geometries. For example, the class "French Horn" (Class 5) and "Parachute" (Class 9) see high false-positive rates. Without color and metallic texture, the complex curves of a horn and the ripples of a parachute canopy appear as similar high-frequency edge noise to the network.

- Analysis: In the second image, the subject is a dog. The edge map is extremely noisy due to the dog's fur generating thousands of small edge fragments. The model classified this as a parachute. This is likely because parachutes (specifically their rippled canopies) also generate high-density, noisy edge maps. Without the brown/white color of the dog's fur or the texture of hair, the model could not distinguish "fur noise" from "canopy noise."

VI. Discussion

The results of this study have profound implications for the theory and practice of Deep Learning.

A. *The "Clever Hans" Effect in AI*

Our data shows a new iteration of the "Clever Hans" effect: the horse that seemed to do math but which was in fact reading its trainer's body language. Conversely, our CNNs seem to be able to distinguish between objects, yet they tend to recognize the "body language" of the image (including texture or background) instead of the object itself. When we take away the texture, the illusion of intelligence collapses.

B. *Implications for Safety-Critical Systems*

This fragility is a safety hazard. Consider an autonomous vehicle trained to recognize pedestrians. If your model is used to the "texture" of clothes (denim, cotton, etc.), it might not see a pedestrian sporting an unusual material (a reflective silver

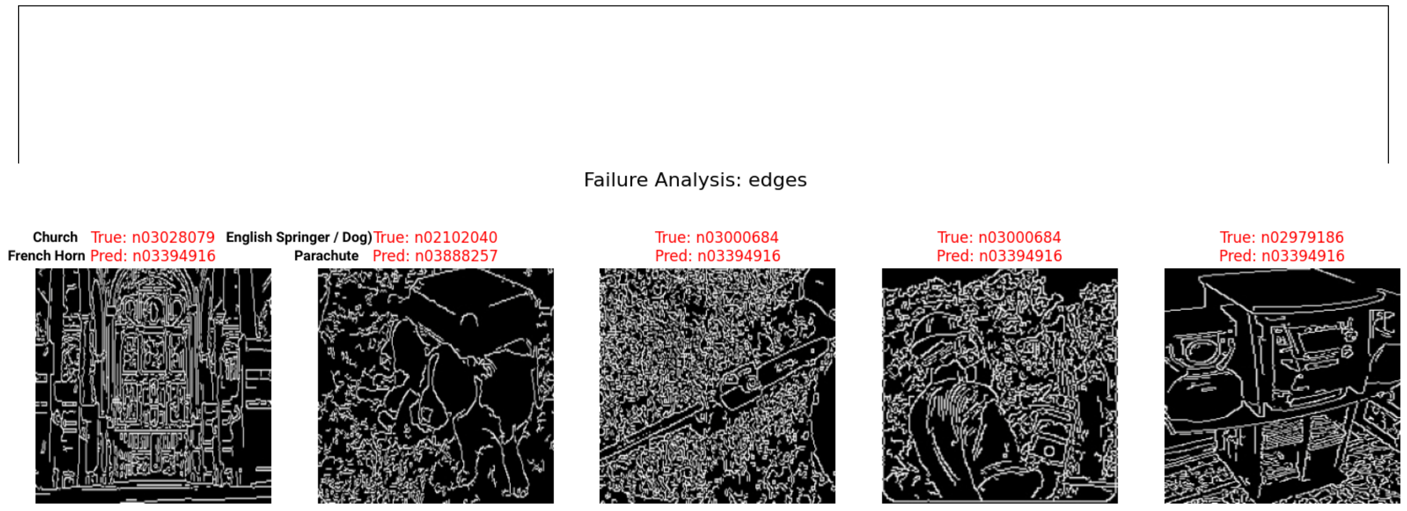

Fig. 3. Visual Failure Examples

costume), or, instead, a pedestrian silhouetted against a bright light where texture has been washed out. We model our edge-detection experiment this way (worst-case visibility scenario), and the result is 26% accuracy, which suggests that we do not have robust models yet to be deployed without constraints.

C. The Failure of Scaling Laws

Perhaps it's surprising to note a similarity between ResNet-18 and ResNet-101. There may be a belief in AI that 'bigger is better'-that scaling up parameters and data solves all problems. But our data rebuts that perspective. Increasing capacity without changing the training objective or the data distribution doesn't solve the problem of shortcut learning-it just creates a more powerful machine for exploiting shortcuts.

VII. Conclusion and Future Work

This project systematically evaluated the robustness of Convolutional Neural Networks against domain shifts involving texture and shape. We demonstrated that standard ResNet models are dangerously biased towards texture, suffering a $\sim$71% performance drop when tested on shape-only edge maps. We further showed that this bias is not mitigated by increasing model depth, as ResNet-101 performed comparably to ResNet-18 on the robust test sets.

However, our work also provides a roadmap for a solution. The high accuracy (87.57%) of the model trained directly on edges proves that CNNs are capable of learning robust, shape-based representations if the training data forces them to do so.

Future Work: To build truly robust AI, we cannot rely on standard ImageNet training. Future research should focus on ”Shape-Biased Training” protocols. This involves:

1) Data Augmentation: Randomly converting a percentage of training images to Canny edge maps or grayscale during training. This creates a ”mixed-signal” curriculum that forces the model to learn features that are invariant to texture.
2) Style Randomization: Implementing ”Style Augmentation” (randomizing textures while keeping shape constant) to penalize the model for relying on texture statistics.
3) Architectural priors: Exploring architectures that explicitly separate shape and texture processing streams (e.g., dual-stream networks) to mimic the human visual cortex.

By prioritizing shape over texture, we can move closer to computer vision systems that do not just pattern-match, but truly perceive the world.

References

[1] Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A., & Brendel, W. (2019). ”ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness.” International Conference on Learning Representations (ICLR).
[2] He, K., Zhang, X., Ren, S., & Sun, J. (2016). ”Deep Residual Learning for Image Recognition.” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770-778.
[3] Beery, S., Van Horn, G., & Perona, P. (2018). ”Recognition in Terra Incognita.” European Conference on Computer Vision (ECCV).
[4] Canny, J. (1986). ”A Computational Approach to Edge Detection.” IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6), 679-698.
[5] Chen, L. C., Papandreou, G., Schroff, F., & Adam, H. (2017). ”Rethinking Atrous Convolution for Semantic Image Segmentation.” arXiv preprint arXiv:1706.05587.
[6] Howard, J., & Gugger, S. (2020). ”FastAI: A Layered API for Deep Learning.” Information, 11(2), 108.

Appendix

Technical Implementation Details

To ensure reproducibility, we provide the specific logic used in our evaluation scripts.

A. Evaluation Logic (src/visualize.py) This script was critical for generating the 10-class confusion matrices. It explicitly targets the validation directory structure to ensure the model evaluates the 10 specific object classes (n01440764, etc.) rather than the directory roots.

```
# Code snippet from src/visualize.py
# Purpose: Point directly to 'val' folder to fix 2-
    class bug

if args.test_mode in ['original', 'grayscale', '
    occlusion']:
    base_dir = os.path.join(args.data_root, 'raw', '
        imagenette2-160')
else:
    base_dir = os.path.join(args.data_root, '
        processed', args.test_mode)
data_dir = os.path.join(base_dir, 'val') #
    Explicitly entering validation set
dataset = datasets.ImageFolder(data_dir,
    get_transforms(args.test_mode))
print(f"Loaded␣{len(dataset.classes)}␣classes␣from␣{
    data_dir}")
```

B. Failure Detection Logic (src/analyze_failures.py) This snippet was used to generate Figure 3. It iterates through the dataset, compares the prediction (preds) to the ground truth (labels), and captures high-confidence errors for visualization.

```
# Code snippet from src/analyze_failures.py
# Purpose: Identify and save images where the model
    failed
with torch.no_grad():
    for inputs, labels in loader:
        outputs = model(inputs)
        _, preds = torch.max(outputs, 1)
        # Find indices where predictions do NOT
            match labels
        wrong_idx = (preds != labels).nonzero(
            as_tuple=True)[0]
        for idx in wrong_idx:
            if len(failures) < 5:
                # Save the failure: (Image Tensor,
                    True Label, Pred Label)
                failures.append((inputs[idx].cpu(),
                    labels[idx].item(), preds[idx].
                    item()))
```