Appendix for "CAT-D: A Causal and Adaptive Topological Defense Framework"

A Detailed Formulation of Core Modules

This appendix provides supplementary details on the mathematical foundations and algorithmic implementations of the CAT-D framework.

A.1 Multi-modal Topological Encoder (MTE) Architecture and Training

The MTE is designed to create a low-dimensional, differentiable embedding of the topological features inherent in a network's activation maps. Let $\mathbf{h}_l \in \mathbb{R}^{C_l \times H_l \times W_l}$ be the activation map at layer l.

A.1.1 Topological Feature Extraction

The foundational topological descriptor we use is the Persistence Diagram (PD), denoted $\text{PD}(\mathbf{h}_l)$. The PD is a set of points in $\mathbb{R}^2$ that captures the birth and death times of topological features (e.g., connected components, holes) in the data. To make this process compatible with gradient-based optimization, we employ a differentiable cubical complex filtration on $\mathbf{h}_l$.

A.1.2 Differentiable Representation Learning

The core of the MTE is a neural network, ϕ_{MTE}, that maps $\mathbf{h}_l$ to a latent vector $\mathbf{z}_l \in \mathbb{R}^d$. A decoder network, ψ_{Dec}, then reconstructs a representation of the PD from $\mathbf{z}_l$. The objective is to ensure that $\mathbf{z}_l$ faithfully encodes the topological information. The reconstruction loss is formulated as:

$$\mathcal{L}_{\text{recon}} = \sum_{l=1}^{L} \text{SWD}\left(\text{PD}(\mathbf{h}_l), \psi_{\text{Dec}}(\phi_{\text{MTE}}(\mathbf{h}_l))\right) \tag{A.1}$$

The Sliced-Wasserstein Distance (SWD) is used for its computational efficiency and differentiability. For two PDs, P_1 and P_2, it is defined as the expected distance between their 1D projections over a set of random directions θ from the unit sphere $\mathbb{S}^1$:

$$\text{SWD}(P_1, P_2) = \int_{\theta \in \mathbb{S}^1} W_1(\text{proj}_\theta(P_1), \text{proj}_\theta(P_2)) d\theta \tag{A.2}$$

where proj_θ is the projection operator and W_1 is the 1-Wasserstein distance between the projected 1D distributions. In practice, this integral is approximated via Monte Carlo sampling. This objective forces $\mathbf{z}_l$ to be a canonical and compressed representation of the activation's topological structure.

A.2 Adaptive Decision Module (ADM) with Extreme Value Theory

The ADM provides a statistically robust method for setting detection thresholds, moving beyond manually-tuned global values. It models the tail distribution of the mismatch scores (S_{static} and S_{dynamic}) derived from clean data.

A.2.1 Theoretical Foundation: The Pickands-Balkema-de Haan Theorem

According to Extreme Value Theory (EVT), for a wide class of distributions, the distribution of exceedances over a sufficiently high threshold u can be approximated by a Generalized Pareto Distribution (GPD). Let S be a random variable representing a mismatch score. The distribution of its exceedances, $S - u$, given $S > u$, is:

$$F_u(y) = P(S - u \leq y | S > u) \approx G(y; \sigma, \xi) \tag{A.3}$$

where $G(y; \sigma, \xi)$ is the GPD, defined as:

$$G(y; \sigma, \xi) = \begin{cases} 1 - (1 + \frac{\xi y}{\sigma})^{-1/\xi} & \text{if } \xi \neq 0 \\ 1 - \exp(-\frac{y}{\sigma}) & \text{if } \xi = 0 \end{cases} \tag{A.4}$$

Here, $\sigma > 0$ is the scale parameter and ξ is the shape parameter.

A.2.2 Calibration and Scoring

During the calibration phase (Stage 2), we collect the mismatch scores $\{S_i\}$ from a clean validation set. We select a high quantile (e.g., the 95th percentile) as the threshold u. The parameters (σ, ξ) of the GPD are then fitted to the exceedances $\{S_i - u | S_i > u\}$ using Maximum Likelihood Estimation.

For a new test sample with mismatch score $S_{\text{test}} > u$, we can calculate its p-value, which is the probability of observing a score as or more extreme:

$$p\text{-value} = P(S > S_{\text{test}} | S > u) = 1 - G(S_{\text{test}} - u; \sigma, \xi) = \left(1 + \frac{\xi(S_{\text{test}} - u)}{\sigma}\right)^{-1/\xi} \tag{A.5}$$

This p-value is a well-calibrated measure of anomalousness. The final Topological Credibility Score, $\text{Conf}_{\text{topo}}$, is derived by fusing the p-values from both the static and dynamic mismatch scores, for instance, through Fisher's method or by taking their minimum. An input is rejected if this fused score falls below a specified significance level α (e.g., $\alpha = 0.05$).

B Algorithmic Implementation and Further Details

B.1 Topological Evolution Sequence Model (TESM)

The TESM is designed to capture the normative, class-agnostic evolution of topological features across the network's layers. This provides a dynamic view of the model's internal state, complementing the static, class-specific norms.

B.1.1 Model Architecture and Objective

We implement the TESM using a standard Long Short-Term Memory (LSTM) network, denoted as f_{TESM}, due to its proven efficacy in modeling sequential data. The input to the TESM is a sequence of topological latent vectors $\mathbf{Z} = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_L)$ extracted from a clean input sample.

The model is trained in an auto-regressive manner. At each step k, it predicts the next topological vector $\hat{\mathbf{z}}_{k+1}$ based on the sequence of preceding vectors $(\mathbf{z}_1, \ldots, \mathbf{z}_k)$. The training objective is to minimize the Mean Squared Error (MSE) between the predicted and the actual vectors across all clean samples in the training set. For a dataset of N clean samples, the loss function is:

$$\mathcal{L}_{\text{TESM}} = \frac{1}{N(L-1)} \sum_{i=1}^{N} \sum_{k=1}^{L-1} \left\| \mathbf{z}_{k+1}^{(i)} - f_{\text{TESM}}\left(\mathbf{z}_1^{(i)}, \ldots, \mathbf{z}_k^{(i)}\right) \right\|_2^2 \tag{A.6}$$

where $\mathbf{z}_k^{(i)}$ is the topological vector from the k-th monitored layer for the i-th training sample. By minimizing this loss, the TESM learns the expected trajectory of topological feature transformations. The Dynamic Mismatch score, S_{dynamic}, used in the ADM is precisely the cumulative squared error from this prediction process for a given test sample.

B.2 Causal Explanation Generator (CEG) Formulation

When an input $\mathbf{x}_{\text{test}}$ is rejected, the CEG's goal is to find a minimally perturbed version of it, $\mathbf{x}_{\text{cf}}$, that would be accepted by the framework. This is framed as an optimization problem that modifies the input to align its topological sequence with a valid "prototype" sequence.

B.2.1 Counterfactual Optimization Problem

Let $\mathbf{Z}_{\text{test}} = (\mathbf{z}_{1,\text{test}}, \ldots, \mathbf{z}_{L,\text{test}})$ be the topological sequence of the rejected input, and let c_{pred} be its predicted class. We first define a target prototype sequence, $\mathbf{Z}_{\text{proto}} = (\mathbf{z}_{1,\text{proto}}, \ldots, \mathbf{z}_{L,\text{proto}})$. This prototype is constructed by finding a high-likelihood representation within the learned class-conditional distributions, $P_{c_{\text{pred}},l}$. For instance, $\mathbf{z}_{l,\text{proto}}$ can be the mean of the closest Gaussian component in the GMM for class c_{pred} at layer l.

The CEG then solves for a small perturbation δ by minimizing the following objective function:

$$\delta^* = \arg\min_{\delta} \lambda_{\text{pert}} \|\delta\|_2^2 + \sum_{l=1}^{L} w_l \left\| \text{MTE}(\mathbf{h}_l(\mathbf{x}_{\text{test}} + \delta)) - \mathbf{z}_{l,\text{proto}} \right\|_2^2 \tag{A.7}$$

Here, the first term penalizes the magnitude of the perturbation, ensuring the counterfactual $\mathbf{x}_{\text{cf}} = \mathbf{x}_{\text{test}} + \delta^*$ is visually similar to the original input. The second term is the "repair" loss, which drives the topological sequence of the modified input towards the target prototype. λ_{pert} is a hyperparameter balancing these two objectives, and w_l are optional weights for each layer. This optimization can be solved using gradient-based methods like Adam, leveraging the end-to-end differentiability of the framework.

B.3 Inference and Defense Algorithm

The complete online defense process is summarized in Algorithm 1. It outlines the sequential steps from receiving a test input to making a final, robust decision.

Algorithm 1 CAT-D Online Inference and Defense Workflow

Input: Test sample $\mathbf{x}_{\text{test}}$, trained CAT-D model f_θ, MTE, TESM, class norms $\{P_{c,l}\}$, calibrated ADM, security threshold τ.
Output: Decision $\in$ {Accept, Reject}, Predicted class c_{pred}.
▷ — Stage 1: Forward Pass and Feature Extraction —
Perform a forward pass: $[\text{logits}, \{\mathbf{h}_l\}_{l=1}^{L}] = f_\theta(\mathbf{x}_{\text{test}})$.
Obtain prediction: $c_{\text{pred}} = \arg\max(\text{logits})$.
Extract topological sequence: $\mathbf{Z}_{\text{test}} = (\text{MTE}(\mathbf{h}_1), \ldots, \text{MTE}(\mathbf{h}_L))$.
▷ — Stage 2: Mismatch Score Calculation —
Initialize $S_{\text{static}} = 0$, $S_{\text{dynamic}} = 0$.
▷ Calculate static mismatch against class norms.
for $l = 1$ to L **do**
    $S_{\text{static}} \leftarrow S_{\text{static}} - \log P_{c_{\text{pred}},l}(\mathbf{z}_{l,\text{test}})$.
end for
▷ Calculate dynamic mismatch using TESM.
for $k = 1$ to $L - 1$ **do**
    $\hat{\mathbf{z}}_{k+1} = \text{TESM}(\mathbf{z}_{1,\text{test}}, \ldots, \mathbf{z}_{k,\text{test}})$.
    $S_{\text{dynamic}} \leftarrow S_{\text{dynamic}} + \|\mathbf{z}_{k+1,\text{test}} - \hat{\mathbf{z}}_{k+1}\|_2^2$.
end for
▷ — Stage 3: Adaptive Decision Making —
Compute p-values: $p_{\text{static}} = \text{ADM}_{\text{static}}(S_{\text{static}})$, $p_{\text{dynamic}} = \text{ADM}_{\text{dynamic}}(S_{\text{dynamic}})$.
Fuse scores into a final credibility score: $\text{Conf}_{\text{topo}} = \min(p_{\text{static}}, p_{\text{dynamic}})$.
▷ — Stage 4: Final Decision —
if $\text{Conf}_{\text{topo}} \geq \tau$ **then**
    return Accept, c_{pred}.
else
    return Reject, c_{pred}. ▷ Optionally trigger CEG.
end if

C Training and Experimental Setup Details

This section provides the specific hyperparameters and configurations used for training the CAT-D framework and evaluating its performance, supplementing the descriptions in the main paper.

C.1 Topology-Guided Training Protocol

The end-to-end training process described in Section 3.1 of the main paper is conducted using a carefully selected set of hyperparameters to balance standard accuracy and topological robustness.

C.1.1 Optimization Parameters

For all experiments, we use the Stochastic Gradient Descent (SGD) optimizer with a momentum of 0.9 and a weight decay of 5×10^{-4}. We employ a cosine annealing learning rate schedule, starting from an initial learning rate of 0.1 and decaying to 0 over the course of 120 epochs. A batch size of 128 is used for all datasets.

C.1.2 Loss Function Configuration

The total loss function, as defined in Equation (2) of the main paper, is a weighted combination of four components. The weights are critical for balancing the model's objectives. We set them as follows based on empirical validation:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{class}} + \lambda_{\text{adv}}\mathcal{L}_{\text{adv}} + \lambda_{\text{stab}}\mathcal{L}_{\text{stab}} + \lambda_{\text{consist}}\mathcal{L}_{\text{consist}} \tag{A.8}$$

The specific values for the weighting coefficients are detailed in Table 1. The weight for the adversarial loss, λ_{adv}, is aligned with standard practices in the literature (e.g., TRADES), while the topological regularizers are weighted to guide the model without overpowering the primary classification objective.

Table 1: Hyperparameter settings for the composite loss function.

Hyperparameter	**Value**
Adversarial Loss Weight (λ_{adv})	1.0
Topological Stability Weight (λ_{stab})	0.25
Topological Consistency Weight (λ_{consist})	0.5

C.2 Architectural and Implementation Specifics

C.2.1 MTE and Monitored Layers

The Multi-modal Topological Encoder (MTE) is implemented as a lightweight Multi-Layer Perceptron (MLP) with two hidden layers of 256 and 128 neurons respectively, using ReLU activations. The final output is a latent vector $\mathbf{z}_l \in \mathbb{R}^{64}$. The choice of monitored layers ($L = 4$) is critical. We select layers that capture features at different semantic levels:

- **For ResNet-18 and WRN-34-10:** We use the output of the final activation in each of the four main residual stages.
- **For ViT-B/16:** We select four uniformly spaced intermediate transformer blocks (specifically, blocks 3, 6, 9, and 12) to capture the evolution of representations.

C.2.2 Probabilistic Modeling of Class Norms

The class-specific topological norms, $P_{c,l}$, are modeled using Gaussian Mixture Models (GMMs). For each class c and layer l, we fit a GMM with $K = 5$ components to the set of clean latent vectors $\{\mathbf{z}_l^{(i)}\}_{i \in \text{class } c}$. This number of components was found to be sufficient to capture the main modes of intra-class topological variation without overfitting.

C.3 Evaluation Protocol and Attack Parameters

Our evaluation is designed to be rigorous and comprehensive, employing standard benchmarks as well as tailored adaptive attacks.

C.3.1 Standard Attack Configuration

The primary evaluations use AutoAttack (AA), which is a parameter-free ensemble of diverse attacks. For the PGD-based attacks used in the varying strength analysis (Section 4.5), we use the following setup:

- **Attack Type:** Projected Gradient Descent (ℓ_∞).
- **Number of Steps:** 20.
- **Step Size (α):** 2/255. This is a strong step size relative to the perturbation budget $\epsilon = 8/255$.
- **Random Starts:** 5 restarts to ensure a robust search for adversarial examples.

C.3.2 Topology-Aware Adaptive Attack

To specifically challenge CAT-D, we designed an adaptive attack that aims to generate an adversarial example $\mathbf{x}_{\text{adv}}$ that not only fools the classifier but also mimics the topological structure of the original clean sample $\mathbf{x}$. This attack optimizes a modified objective function within the PGD framework:

$$\max_{\|\delta\|_\infty \leq \epsilon} \left(\mathcal{L}_{\text{CE}}(f_\theta(\mathbf{x} + \delta), y) - \gamma \cdot \mathcal{L}_{\text{consist}}(\mathbf{x}, \mathbf{x} + \delta) \right) \tag{A.9}$$

Here, $\mathcal{L}_{\text{CE}}$ is the standard cross-entropy loss, and $\mathcal{L}_{\text{consist}}$ is the topological consistency loss defined in Equation (4) of the main paper. The hyperparameter γ controls the trade-off. By subtracting the consistency loss, the attacker is incentivized to find a perturbation δ that minimizes the topological distance to the clean sample, thereby attempting to evade our detector. The strong performance of CAT-D, even against this specialized attack (as reported in the main paper), validates the robustness of its multi-faceted detection strategy (static, dynamic, and EVT-based).

D Further Analysis and Discussion

D.1 On the Synergy of Static and Dynamic Topological Features

The core defensive strength of CAT-D arises from the complementary nature of its static and dynamic mismatch scores. This design is rooted in a fundamental hypothesis about the effect of adversarial perturbations on a network's internal representations.
Hypothesis: Adversarial perturbations introduce high-frequency, non-semantic noise into the input. While this noise is small in magnitude, it manifests in two distinct ways within the network's feature hierarchy:

1. **State Corruption:** At individual layers, the perturbation disrupts the local geometric and topological structure of the activation maps. This often leads to the creation of many spurious, short-lived topological features (e.g., small connected components) that are inconsistent with the feature patterns of natural images. This corruption of the instantaneous "state" of a layer's representation is what the **Static Mismatch Score (S_{static})** is designed to detect. It answers the question: "Does the topology at layer l conform to the expected norm for the predicted class c?"

2. **Process Corruption:** As data propagates through the network, its topological representation undergoes a structured transformation. For clean inputs, this evolution is governed by the learned functions of the network and is therefore predictable and consistent. Adversarial perturbations disrupt this sequential transformation process, causing an anomalous "jump" or deviation in the evolutionary trajectory between layers. An attack might be sophisticated enough to produce a final-layer topology that appears valid (low S_{static}), but the path taken to reach that state is unnatural. This is what the **Dynamic Mismatch Score (S_{dynamic})** detects. It answers the question: "Is the transformation of topology from layer k to $k+1$ consistent with the learned rules of topological evolution?"

By combining these two orthogonal detection criteria, CAT-D forces an adversary to solve a much more constrained optimization problem. The attacker must craft a perturbation that not only produces a target output (fooling the classifier) but also ensures that the resulting topological features adhere to both the static, class-specific norms at each layer and the dynamic, class-agnostic rules of evolution between layers. This dual-constraint system significantly increases the difficulty of crafting a successful and stealthy attack, forming the theoretical basis for CAT-D's enhanced robustness.

D.2 Supplementary Experimental Validation

We provide additional experimental results to offer a more granular view of CAT-D's performance.

D.2.1 Performance Breakdown against Specific Attacks

While AutoAttack provides a holistic measure of robustness, it is useful to see how the defense performs against its individual components. Table 2 details the robust accuracy of CAT-D against several key attacks included in the AutoAttack suite, compared to a strong

baseline. This demonstrates that CAT-D's high performance is not due to robustness against only one type of attack but is consistent across different attack strategies.

Table 2: Detailed Robust Accuracy (%) against individual attacks on CIFAR-10 with ResNet-18 ($\ell_\infty, \epsilon = 8/255$).

Model	**APGD-CE**	**APGD-DLR**	**FAB**	**Square Attack**
AWP (SOTA)	58.1	57.3	59.2	58.5
CAT-D (Ours)	**60.2**	**59.5**	**61.4**	**60.8**

D.2.2 Sensitivity Analysis of the ADM Threshold

A practical concern for any detection-based system is its sensitivity to the decision threshold. The use of EVT in our ADM is intended to provide a statistically grounded and stable decision boundary. To verify this, we evaluate the system's performance on CIFAR-100 while varying the security threshold τ (the target p-value). As shown in Table 3, CAT-D's performance is remarkably stable across a range of reasonable threshold values. A stricter threshold (lower τ) slightly increases robust accuracy at the cost of a marginally higher False Positive Rate (FPR) on clean data, while a more lenient threshold shows the opposite trade-off. This graceful degradation demonstrates the robustness of the ADM and allows practitioners to tune the system to their specific security needs (e.g., minimizing false alarms vs. maximizing threat detection) without a drastic performance collapse.

Table 3: Performance of the ADM under different security thresholds (τ) on CIFAR-100. TPR is the true positive rate for detecting attacks; FPR is the false positive rate on clean data.

Threshold τ	**TPR (%)**	**FPR (%)**	**Robust Acc. (%)**
0.01 (Strict)	96.8	7.8	34.6
0.05 (Default)	**96.4**	**7.0**	**34.4**
0.10 (Lenient)	95.9	6.1	34.1

E Qualitative Analysis: Visualizing Explanations

A key advantage of CAT-D is its ability to generate human-interpretable explanations for its rejection decisions. This section describes the qualitative results that can be produced by the Causal Explanation Generator (CEG). While the main paper presented conceptual diagrams, here we detail the nature of the actual visual outputs.

E.1 Causal Attribution Saliency Maps

To understand *which parts* of an input image caused the topological anomaly, we leverage the saliency map generated from the gradient of the topological credibility score with respect to the input pixels, $\nabla_{\mathbf{x}} \text{Conf}_{\text{topo}}$.

E.1.1 Interpretation

This gradient highlights the pixels that, if changed, would most significantly impact the input's topological score. In practice, for a rejected adversarial sample, this map pinpoints the locations where the adversarial perturbation has most effectively corrupted the image's natural structure. Unlike standard saliency maps for classification, which highlight semantically relevant objects (e.g., the "cat" in a cat image), our topological saliency map often reveals a high-frequency, non-semantic pattern corresponding to the additive noise. This provides direct visual evidence of the adversarial manipulation and confirms that the defense is reacting to structural artifacts rather than semantic content. A typical visualization would show the original image, its adversarial counterpart (which may be visually indistinguishable), and the causal attribution map, where the adversarial noise pattern becomes clearly visible.

E.2 Counterfactual "Repair" Visualization

The CEG's optimization process (Equation A.7) produces a tangible counterfactual example, $\mathbf{x}_{\text{cf}}$. This example serves as a constructive explanation, answering the question: "*How* should this image be modified to be considered legitimate?"

E.2.1 Visual Properties

The generated $\mathbf{x}_{\text{cf}}$ is visually almost identical to the rejected adversarial input $\mathbf{x}_{\text{test}}$, as the optimization is constrained to find a minimal perturbation δ. However, subtle changes in the pixel values are made to restore the "correct" topological properties. When visualizing the difference, $\mathbf{x}_{\text{cf}} - \mathbf{x}_{\text{test}}$, the resulting image again reveals the structure of the adversarial noise that was "removed" or "corrected" by the CEG. This process provides a powerful debugging and analysis tool, demonstrating precisely what structural aspects the defense found objectionable and how it would rectify them.

F Limitations and Future Directions

While CAT-D establishes a new state-of-the-art in robust and interpretable defense, we acknowledge several limitations that open avenues for future research.

F.1 Current Limitations

- **Computational Overhead in Training:** The topology-guided training, which requires calculating persistence diagrams and multiple loss terms, introduces a notable computational cost compared to standard adversarial training (as shown in Table 5 of the main paper). While inference is highly efficient, the training phase requires more resources.
- **Dependence on Layer Selection:** The performance of CAT-D is contingent on the choice of monitored layers. Our current approach uses a heuristic of selecting

layers at different semantic depths. A systematic, automated method for identifying the most topologically informative layers for a given architecture could further improve performance.

- **Abstractness of Topological Features:** While we can visualize the *effect* of topological corruption, the topological features themselves (e.g., a Betti-1 hole in a high-dimensional activation space) lack direct semantic meaning. Bridging the gap between these abstract mathematical descriptors and concrete visual concepts remains an open challenge.

F.2 Future Research Directions

- **Efficiency Enhancements:** Exploring approximations for persistent homology or leveraging hardware-accelerated TDA libraries could significantly reduce the training overhead, making the framework more accessible.

- **Extension to Other Domains:** The core principles of CAT-D—monitoring the evolution of internal structural representations—are not limited to images. Future work could adapt this framework to other data modalities, such as graph neural networks (monitoring graph topology) or time-series models (monitoring the topology of windowed subsequences).

- **Theoretical Guarantees:** While our empirical results are strong, developing theoretical bounds on the robustness conferred by topological consistency regularization would be a significant contribution. This could involve connecting the stability of persistence diagrams to the Lipschitz constant of the network layers.

- **Integration with Generative Models:** The CEG provides a "corrective" perturbation. Integrating this with more powerful generative models could enable not just explanation but also real-time purification or reconstruction of adversarial inputs, turning the defense into a proactive correction mechanism.