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 $\mathrm{PD}(\mathbf{h}_l)$. 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:

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

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 $\mathbb{S}^1$:

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

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 Details

This section presents the detailed algorithms for the training and calibration phases of the CAT-D framework.

B.1 Algorithm for End-to-End Topologically-Guided Training (Stage 1)

The training process, described in Section 3.1 of the main paper, aims to jointly optimize the backbone network and the MTE to produce topologically robust and consistent representations. The complete procedure is detailed in Algorithm 1. The core of this algorithm is the computation of the total loss function, $\mathcal{L}_{\text{total}}$, which integrates standard classification, adversarial robustness, and our proposed topological regularizers. The weights $\lambda_{\text{adv}}, \lambda_{\text{stab}}, \lambda_{\text{consist}}$ are crucial hyperparameters that balance these competing objectives.

Algorithm 1 End-to-End Topologically-Guided Training of CAT-D

```
1: Input: Training dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$, backbone network $f_\theta$, MTE $\phi_{\text{MTE}}$, decoder $\psi_{\text{Dec}}$.
2: Input: Loss weights $\lambda_{\text{adv}}, \lambda_{\text{stab}}, \lambda_{\text{consist}}$.
3: Input: Number of epochs $N_e$, learning rate $\eta$.
4: Initialize parameters $\theta$ of $f_\theta$ and parameters of $\phi_{\text{MTE}}, \psi_{\text{Dec}}$.
5: for epoch = 1 to $N_e$ do
6:     for each batch $(\mathbf{X}, \mathbf{Y}) \in \mathcal{D}$ do
7:                                         ▷ — Generate adversarial and augmented samples —
8:         Generate adversarial batch $\mathbf{X}_{\text{adv}}$ from $\mathbf{X}$ using PGD attack.
9:         Generate two augmented batches $\mathbf{X}_1, \mathbf{X}_2$ from $\mathbf{X}$ using random augmentations.
10:                                        ▷ — Forward passes to extract activations —
11:        Get multi-layer activations $\{\mathbf{h}_l(\mathbf{X})\}$, $\{\mathbf{h}_l(\mathbf{X}_{\text{adv}})\}$, $\{\mathbf{h}_l(\mathbf{X}_1)\}$, $\{\mathbf{h}_l(\mathbf{X}_2)\}_{l=1}^{L}$.
12:                                        ▷ — Compute loss components —
13:        $\mathcal{L}_{\text{class}} = \text{CrossEntropy}(f_\theta(\mathbf{X}), \mathbf{Y})$.
14:        $\mathcal{L}_{\text{adv}} = \text{CrossEntropy}(f_\theta(\mathbf{X}_{\text{adv}}), \mathbf{Y})$.
15:                                        ▷ — Topological losses in latent space —
16:        Compute latent vectors $\mathbf{z}_l(\cdot) = \phi_{\text{MTE}}(\mathbf{h}_l(\cdot))$ for all batches and layers.
17:        $\mathcal{L}_{\text{recon}} = \sum_{l=1}^{L} \text{SWD}\left(\text{PD}(\mathbf{h}_l(\mathbf{X})), \psi_{\text{Dec}}(\mathbf{z}_l(\mathbf{X}))\right)$.
18:        $\mathcal{L}_{\text{stab}} = \sum_{l=1}^{L} \|\mathbf{z}_l(\mathbf{X}_1) - \mathbf{z}_l(\mathbf{X}_2)\|_2^2$.
19:        $\mathcal{L}_{\text{consist}} = \sum_{l=1}^{L} \|\mathbf{z}_l(\mathbf{X}) - \mathbf{z}_l(\mathbf{X}_{\text{adv}})\|_2^2$.
20:                                        ▷ — Total loss and backpropagation —
21:        $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{class}} + \lambda_{\text{adv}}\mathcal{L}_{\text{adv}} + \mathcal{L}_{\text{recon}} + \lambda_{\text{stab}}\mathcal{L}_{\text{stab}} + \lambda_{\text{consist}}\mathcal{L}_{\text{consist}}$.
22:        Update all parameters by backpropagating $\mathcal{L}_{\text{total}}$ with learning rate $\eta$.
23:    end for
24: end for
25: Output: Trained backbone $f_\theta$ and MTE $\phi_{\text{MTE}}$.
```

B.2 Algorithm for Learning Topological Norms and Calibration (Stage 2)

After the main model is trained, its parameters are frozen. This stage learns the class-conditional distributions of topological features and trains the sequence model on clean data. It concludes by calibrating the ADM using a separate validation set. This process is outlined in Algorithm 2.

B.2.1 Topological Evolution Sequence Model (TESM)

The TESM is a sequence-to-sequence model, implemented using an architecture such as an LSTM or a Transformer encoder. Its objective is to learn the conditional probability distribution $p(\mathbf{z}_{k+1}|\mathbf{z}_1, \dots, \mathbf{z}_k)$. It is trained on sequences extracted from all classes in the training set, thereby learning a universal, class-agnostic model of how topological features should evolve through the network layers. The training loss for the TESM is typically the Mean Squared Error (MSE) between the predicted and actual subsequent latent vectors:

$$\mathcal{L}_{\text{TESM}} = \mathbb{E}_{\mathbf{Z}\sim\mathcal{D}_{\text{clean}}} \left[\sum_{k=1}^{L-1} \|\mathbf{z}_{k+1} - \text{TESM}(\mathbf{z}_1, \dots, \mathbf{z}_k)\|_2^2 \right] \tag{A.6}$$

C Online Inference and Explanation Generation

C.1 Algorithm for Online Inference and Defense (Stage 3)

During inference, the framework executes a highly efficient feed-forward process. For each incoming sample $\mathbf{x}_{\text{test}}$, it computes the static and dynamic mismatch scores and uses the pre-calibrated ADM to derive a final, statistically grounded credibility score. The detailed steps are outlined in Algorithm 3. The process is designed to be fast, as it requires only a single forward pass through the main network, followed by computations involving the lightweight MTE, TESM, and pre-fitted distribution models.

C.2 Causal Explanation Generator (CEG) Formulation (Stage 4)

When an input $\mathbf{x}_{\text{test}}$ is rejected, the CEG is invoked to provide two forms of explanation.

C.2.1 Counterfactual Sample Generation

The goal is to find a minimally modified input $\mathbf{x}_{\text{cf}}$ that is visually similar to $\mathbf{x}_{\text{test}}$ but is accepted by the CAT-D framework. This is formulated as a constrained optimization problem. First, we define a target "prototype" topological sequence $\mathbf{Z}_{\text{proto}}$. For a rejected sequence $\mathbf{Z}_{\text{test}}$, its prototype is found by projecting each $\mathbf{z}_l$ onto the closest high-density region of its class-conditional distribution $P_{c_{\text{pred}},l}$. For a GMM, this corresponds to the mean of the most likely component.

The optimization problem to find $\mathbf{x}_{\text{cf}}$ is then:

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

Algorithm 2 Learning Topological Norms and ADM Calibration

```
1: **Input:** Trained and frozen $f_\theta$ and $\phi_{\text{MTE}}$.
2: **Input:** Clean training set $\mathcal{D}_{\text{train}}$, clean validation set $\mathcal{D}_{\text{val}}$.
3: **Input:** Number of classes $N_C$.
4: Initialize TESM model, initialize empty sets $\mathcal{Z}_{c,l}$ for $c = 1..N_C, l = 1..L$.
5:                                   ▷ — Part 1: Collect latent vectors and train TESM —
6: Initialize an empty set of sequences $\mathcal{S}_{\text{seq}}$.
7: **for** each batch $(\mathbf{X}, \mathbf{Y}) \in \mathcal{D}_{\text{train}}$ **do**
8:     Extract latent sequences $\mathbf{Z} = \{\mathbf{z}_1, \dots, \mathbf{z}_L\}$ for each sample in $\mathbf{X}$.
9:     Add sequences in $\mathbf{Z}$ to $\mathcal{S}_{\text{seq}}$.
10:    **for** $c = 1$ to $N_C$ **do**
11:        **for** $l = 1$ to $L$ **do**
12:            Collect latent vectors for class $c$ at layer $l$ into $\mathcal{Z}_{c,l}$.
13:        **end for**
14:    **end for**
15: **end for**
16: Train TESM on $\mathcal{S}_{\text{seq}}$ by minimizing $\mathcal{L}_{\text{TESM}}$ (Eq. A.6).
17:                                   ▷ — Part 2: Fit class-conditional distributions —
18: **for** $c = 1$ to $N_C$ **do**
19:     **for** $l = 1$ to $L$ **do**
20:         Fit a probabilistic model $P_{c,l}$ (e.g., GMM) to the vectors in $\mathcal{Z}_{c,l}$.
21:     **end for**
22: **end for**
23:                                   ▷ — Part 3: Calibrate ADM using EVT —
24: Initialize empty lists $Scores_{\text{static}}, Scores_{\text{dynamic}}$.
25: **for** each $(\mathbf{x}, y) \in \mathcal{D}_{\text{val}}$ **do**
26:     Predict class $c_{\text{pred}} = \text{argmax}(f_\theta(\mathbf{x}))$. Assume $c_{\text{pred}} = y$.
27:     Extract latent sequence $\mathbf{Z}_{\text{val}} = (\mathbf{z}_1, \dots, \mathbf{z}_L)$.
28:     $S_{\text{static}} = \sum_{l=1}^{L} w_l \cdot (-\log P_{c_{\text{pred}},l}(\mathbf{z}_l))$. Append to $Scores_{\text{static}}$.
29:     $S_{\text{dynamic}} = \sum_{k=1}^{L-1} \|\mathbf{z}_{k+1} - \text{TESM}(\mathbf{z}_1, \dots, \mathbf{z}_k)\|_2^2$. Append to $Scores_{\text{dynamic}}$.
30: **end for**
31: For $Scores_{\text{static}}$: find threshold $u_{\text{static}}$, fit GPD parameters $(\sigma_{\text{static}}, \xi_{\text{static}})$.
32: For $Scores_{\text{dynamic}}$: find threshold $u_{\text{dynamic}}$, fit GPD parameters $(\sigma_{\text{dynamic}}, \xi_{\text{dynamic}})$.
33: **Output:** Trained TESM, fitted distributions $\{P_{c,l}\}$, and calibrated EVT models for ADM.
```

subject to $\mathbf{x}'$ remaining a valid image (e.g., pixel values in $[0, 1]$). The hyperparameter γ balances the visual similarity to the original input against the topological similarity to the prototype. This optimization is solved via gradient descent, starting from $\mathbf{x}' = \mathbf{x}_{\text{test}}$.

C.2.2 Causal Attribution Map

To identify the input regions most responsible for the topological anomaly, we compute the gradient of the final credibility score with respect to the input pixels. Since the decision boundary is defined by $\text{Conf}_{\text{topo}} = \tau$, the most informative gradient is that of $\text{Conf}_{\text{topo}}$

Algorithm 3 Online Inference and Defense

```
1: Input: Test sample $\mathbf{x}_{\text{test}}$.
2: Input: Trained models ($f_\theta, \phi_{\text{MTE}}$, TESM), distributions $\{P_{c,l}\}$, EVT models for ADM.
3: Input: Security threshold $\tau \in [0, 1]$.
4:                         ▷ — Step 1: Forward Pass and Representation Extraction —
5: Compute model prediction: $c_{\text{pred}} = \text{argmax}(f_\theta(\mathbf{x}_{\text{test}}))$.
6: Extract multi-layer activations $\{\mathbf{h}_l(\mathbf{x}_{\text{test}})\}_{l=1}^{L}$.
7: Compute the topological latent sequence $\mathbf{Z}_{\text{test}} = (\phi_{\text{MTE}}(\mathbf{h}_1), \dots, \phi_{\text{MTE}}(\mathbf{h}_L))$.
8:                                          ▷ — Step 2: Mismatch Score Calculation —
9: $S_{\text{static}} = \sum_{l=1}^{L} w_l \cdot (-\log P_{c_{\text{pred}},l}(\mathbf{z}_l))$.
10: $S_{\text{dynamic}} = \sum_{k=1}^{L-1} \|\mathbf{z}_{k+1} - \text{TESM}(\mathbf{z}_1, \dots, \mathbf{z}_k)\|_2^2$.
11:                                         ▷ — Step 3: Adaptive Decision via EVT —
12: Calculate $p_{\text{static}}$ from $S_{\text{static}}$ using the corresponding GPD model (Eq. A.5).
13: Calculate $p_{\text{dynamic}}$ from $S_{\text{dynamic}}$ using the corresponding GPD model (Eq. A.5).
14: Fuse p-values: $\text{Conf}_{\text{topo}} = \min(p_{\text{static}}, p_{\text{dynamic}})$.     ▷ A conservative fusion strategy
15:                                                      ▷ — Step 4: Final Decision —
16: if $\text{Conf}_{\text{topo}} \geq \tau$ then
17:     Return Accept, Predicted Class: $c_{\text{pred}}$.
18: else
19:     Return Reject, trigger Causal Explanation Generator (CEG).
20: end if
```

itself. The attribution map $\mathbf{M}$ is defined as:

$$\mathbf{M} = |\nabla_{\mathbf{x}_{\text{test}}} \text{Conf}_{\text{topo}}(\mathbf{x}_{\text{test}})| \tag{A.8}$$

This gradient is computed via backpropagation through the entire inference pipeline, which is possible due to the differentiability of all components. The resulting map $\mathbf{M}$ highlights pixels whose modification would most rapidly increase the input's topological credibility, thus pointing to the source of the anomaly.

D Experimental Configuration Details

This section provides key hyperparameters and settings used in our experiments to ensure transparency and facilitate reproducibility.

D.1 Training Hyperparameters

The following settings were used for training the CAT-D framework across all datasets unless otherwise specified.

D.2 Adversarial Attack Configuration

For evaluating robustness, we use a standardized set of strong attacks. The primary evaluation is performed using AutoAttack (AA). For the PGD-based attacks used in

Table 1: Key hyperparameters for CAT-D training.

Hyperparameter	**Value**
Optimizer	AdamW
Base Learning Rate	1×10^{-3}
Weight Decay	5×10^{-4}
Learning Rate Schedule	Cosine Annealing
Training Epochs	120
Batch Size	128
Loss Function Weights	
λ_{adv} (Adversarial)	1.0
λ_{stab} (Stability)	0.5
λ_{consist} (Consistency)	1.5
MTE and TESM Configuration	
Latent Dimension d	64
Monitoring Layers L (for ResNet-18)	4 (Outputs of the 4 main blocks)
TESM Architecture	2-layer LSTM with 128 hidden units

ablation and strength analysis, the parameters are set to create a challenging threat model.

Table 2: Parameters for PGD-based adversarial attacks.

Parameter	**Value**
Norm Constraint	ℓ_∞
Perturbation Budget (ϵ)	8/255 (CIFAR), 4/255 (Tiny ImageNet)
Step Size (α)	2/255
Number of Iterations	20
Random Starts	5

E Theoretical Analysis and Discussion

This section provides further theoretical justification for key design choices within the CAT-D framework and discusses their implications for robustness and interpretability.

E.1 On the Necessity of a Multi-Modal, Multi-Layer Approach

A critical design choice in CAT-D is the use of topological representations from multiple network layers, rather than relying on a single layer's output (e.g., the final hidden layer). This choice is motivated by the hierarchical nature of feature extraction in deep networks.

- **Early Layers:** The topological structures in early layers typically correspond to simple, low-level image features like edges and textures. An adversarial attack that subtly manipulates these base features might not be detectable from a semantic standpoint but will cause significant, anomalous shifts in the topology of early-layer activations.

- **Deep Layers:** Later layers capture more abstract, semantic information. The topology here relates to the relationships between high-level concepts.

By monitoring the entire hierarchy, CAT-D gains a comprehensive view of an input's structural integrity. A sophisticated attack must successfully mimic the correct topological features at *all* monitored layers to evade detection—a task that is empirically and theoretically much more difficult than fooling the final classification layer alone. The multi-modal approach provides a defense-in-depth capability that is absent in single-layer analysis methods.

E.2 Synergy between Static and Dynamic Mismatch Scores

The use of two distinct mismatch scores, S_{static} and S_{dynamic}, is not redundant but synergistic. They are designed to capture different failure modes.

- **Static Mismatch (S_{static}):** This score assesses whether the topological state at each layer is valid for the predicted class. It is effective at detecting "state corruption," where the representation at a specific layer is anomalous (e.g., an attacker introduces noise that creates thousands of spurious connected components). However, it might fail to detect an attack where each individual layer's topology is plausible, but the transition between them is not.

- **Dynamic Mismatch (S_{dynamic}):** This score evaluates the integrity of the "process" of feature transformation across layers. It is specifically designed to detect "transition corruption." For example, an attack might generate a valid-looking "texture" topology at layer l and a valid-looking "object part" topology at layer $l+1$, but the transformation from the former to the latter violates the learned rules of the TESM. This indicates a breakdown in the natural, learned feature extraction hierarchy.

An adversarial input must simultaneously satisfy both the state-based and transition-based constraints to be accepted, creating a much more constrained solution space for the attacker. The relationship between these scores is summarized in Table 3.

Table 3: Interpretation of mismatch score combinations.

S_{static}	S_{dynamic}	**Interpretation**
Low	Low	Clean, legitimate input.
High	High	Gross anomaly; likely invalid topology at multiple layers.
High	Low	State corruption; topology at one or more layers is invalid.
Low	High	Transition corruption; inter-layer topological evolution is anomalous.

E.3 Theoretical Advantages of Topological Consistency Regularization

The topological consistency loss, $\mathcal{L}_{\text{consist}}$, offers a more structured form of regularization than simply minimizing the classification loss on adversarial examples. Standard adversarial training forces the model to assign the correct label to a perturbed input, which can be achieved by learning a highly complex and non-smooth decision boundary around the data manifold.

In contrast, $\mathcal{L}_{\text{consist}}$ enforces a much stronger condition: that the internal topological representation of an adversarial input should remain close to that of its clean counterpart. Mathematically, this encourages the function $g_l(\mathbf{x}) = \phi_{\text{MTE}}(\mathbf{h}_l(\mathbf{x}))$ to be locally Lipschitz continuous with a small Lipschitz constant in the vicinity of the training data. A small Lipschitz constant implies that small changes in the input $\mathbf{x}$ will only lead to small changes in the output topological representation $\mathbf{z}_l$.

$$\|g_l(\mathbf{x}) - g_l(\mathbf{x}_{\text{adv}})\|_2 \leq K_l \|\mathbf{x} - \mathbf{x}_{\text{adv}}\|_2 \tag{A.9}$$

Minimizing $\mathcal{L}_{\text{consist}}$ is equivalent to minimizing the left-hand side of this inequality, which effectively regularizes and reduces the local Lipschitz constant K_l. This leads to a smoother and more stable representation space, making it inherently more difficult for an attacker to induce a significant topological shift with a small input perturbation. This provides a more fundamental form of robustness than label-level adversarial training alone.