Robust and Explainable Slice: Scalable Linear Extreme Classifiers

Kapil Yadav
IIT Jodhpur
Jodhpur, India
b22ai024@iitj.ac.in

Arsewad Bhagwan
IIT Jodhpur
Jodhpur, India
b22ai010@iitj.ac.in

Abstract

We propose key enhancements to SLICE (Scalable Linear Extreme Classifiers) for extreme multi-label classification, with a dual emphasis on robustness and explainability—two pillars critical for dependable AI systems. Our contributions are threefold: (1) an adversarial training framework that perturbs input features during learning to enhance model resilience against malicious manipulations, (2) a lightweight and scalable feature importance analysis system that quantifies and visualizes the impact of each input feature on label predictions, offering transparent and human-understandable explanations, and (3) a defense mechanism at inference time that detects and mitigates adversarial perturbations through feature squeezing techniques. By leveraging the inherent linearity of SLICE's discriminative classifiers, our approach delivers interpretability with negligible computational cost. We demonstrate the effectiveness of our framework on the EURLex-4K dataset, where it sustains high predictive performance, exhibits robust behavior under adversarial conditions, and introduces only 0.5 seconds of overhead per prediction for explanation generation. These enhancements make SLICE more suitable for high-stakes real-world applications such as legal document tagging and medical diagnostics, where both reliability and transparency are paramount.

Keywords

Extreme Multi-label Classification (XML), SLICE, Model Explainability, Adversarial Robustness, Feature Importance Analysis, Feature Squeezing, Adversarial Training, Linear Classifiers, Attribution Methods, Gaussian Perturbation, Feature Contribution, Defensive Mechanisms, High-Dimensional Classification

ACM Reference Format:
Kapil Yadav and Arsewad Bhagwan. 2025. Robust and Explainable Slice: Scalable Linear Extreme Classifiers. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (XML-CNN).* ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Extreme Multi-label Classification (XML) addresses the challenging task of assigning items to relevant subsets of labels from extremely large label spaces, often containing thousands or millions of potential labels. XML has become increasingly important in modern applications such as product recommendation, legal document tagging, biomedical indexing, and web search where scalability and accuracy are paramount.

While SLICE (Scalable Linear Extreme Classifiers) [?] has demonstrated impressive efficiency through its two-stage approach—leveraging label clustering followed by one-vs-rest linear classification—it faces critical limitations in two important areas: lack of interpretability and vulnerability to adversarial inputs. These limitations restrict its adoption in scenarios where model transparency and robustness are non-negotiable.

Traditional XML research prioritizes metrics like Precision@k and training time, often at the cost of explainability and robustness. This oversight hinders deployment in high-stakes domains such as healthcare, finance, and legal systems, where understanding and trusting predictions is as crucial as the predictions themselves. Without explanations for why specific labels are predicted, users cannot properly validate model decisions, increasing skepticism and reducing the utility of the classifier. Additionally, the susceptibility to adversarial examples—where imperceptible input perturbations lead to misclassifications—raises serious concerns around the model's dependability in practical environments.

To overcome these challenges, we enhance the SLICE framework with two key dependability features that target its core limitations:

- **Explainability:** A feature contribution analysis system that identifies and visualizes the most influential features for each prediction. This interpretability tool leverages the linear nature of SLICE's classifiers, offering clear insights into the decision process with minimal overhead. It empowers users to trace how inputs influence outputs, fostering greater trust and accountability.
- **Adversarial Robustness:** A comprehensive defense strategy combining Gaussian perturbation-based adversarial training with inference-time feature squeezing. This hybrid mechanism improves model stability against crafted adversarial inputs while maintaining prediction quality.

Our experimental results on the EURLex-4K dataset [4], containing 3,956 labels, highlight the efficacy of our approach. The enhanced SLICE model maintains high prediction accuracy (77.39% Precision@1) while delivering meaningful and human-understandable explanations. Moreover, it exhibits strong resistance to adversarial perturbations, preserving over 95% of its performance under attack—compared to the baseline's steep drop below 80%. These improvements are achieved with minimal computational overhead; the explanation module introduces an average delay of only 0.5 seconds per prediction, keeping the system suitable for real-time or near-real-time applications.

Our work represents a step forward in making extreme classification models more reliable, transparent, and ready for deployment in mission-critical environments. By addressing these key dependability challenges, we aim to bridge the gap between high-performance XML and the practical requirements of trustworthy AI systems.

2 Method Overview

The original SLICE architecture [?] employs a two-stage approach for extreme multi-label classification, which cleverly integrates approximate nearest neighbor (ANN) search with discriminative linear classifiers. This design enables SLICE to achieve high scalability and accuracy even on datasets with millions of labels. The first stage of SLICE utilizes a graph-based ANN mechanism—typically implemented via hierarchical navigable small world (HNSW) graphs—to efficiently shortlist a small subset of relevant labels for each input instance. This drastically reduces the computational burden of evaluating the full label space.

In the second stage, for each shortlisted label, a linear classifier is trained to determine the likelihood of that label being relevant. These classifiers are optimized using hinge loss, ensuring both simplicity and speed during inference. The output scores are then used to rank and return the top-k predictions for any given instance. This decoupled architecture of retrieval followed by classification forms the foundation upon which we build our improvements.

To enhance this robust framework, we integrate two vital dependability features: explainability and adversarial robustness. Our explainability module builds upon the classifier weights by introducing a mechanism for computing and visualizing feature importance scores, thus allowing users to understand the decision logic behind each prediction. Simultaneously, our adversarial robustness module augments the model's training and inference pipeline through the application of adversarial training techniques and input preprocessing defenses such as feature squeezing.

Together, these enhancements transform SLICE from a fast and accurate predictor into a more transparent and resilient system, better suited for deployment in high-stakes environments where interpretability and security are critical. The following sections delve deeper into the architecture, mathematical formulations, and implementation details of these modifications.

2.1 Original SLICE Architecture

2.1.1 ***Generative Stage.*** Utilizes approximate nearest neighbor search (ANNS)[3] to efficiently identify candidate labels for test instances.

For a test instance $\mathbf{x} \in \mathbb{R}^d$, the generative stage finds labels with similar feature representations.

2.1.2 ***Discriminative Stage.*** Trains a linear classifier for each label to refine predictions:

$$s_l(\mathbf{x}) = \mathbf{w}_l^T \mathbf{x} \tag{1}$$

where $\mathbf{w}_l$ represents the weight vector for label l.

2.1.3 ***Prediction Process.*** The final score is computed by combining generative and discriminative components:

$$\text{score}(x, l) = s_l(\mathbf{x}) \tag{2}$$

Labels are then ranked by scores, with the top-k labels assigned to the instance.

2.2 Proposed Enhancements

2.2.1 ***Feature Importance-Based Explainability.*** **Feature Contribution Analysis:** For each prediction, we compute the contribution of individual features to the final score:

$$c_{i,j} = w_{i,j} \times x_j \tag{3}$$

where $c_{i,j}$ is the contribution of feature j to label i, $w_{i,j}$ is the weight for feature j in label i's classifier, and x_j is the value of feature j in the test instance.

Features are then ranked by absolute contribution magnitude to identify the most influential features for each prediction. The implementation stores these importance values in `feature_importance.txt` for subsequent analysis.

2.2.2 ***Adversarial Robustness.*** **Gaussian Perturbation-based Adversarial Training[2]:** During training, we generate perturbed examples by adding controlled random noise:

$$\mathbf{x_{adv}} = \mathbf{x} + \delta, \quad \delta \sim \mathcal{N}(0, \sigma^2) \tag{4}$$

This is implemented in `slice.cpp` using C++ random number generation:

$$\text{adv\_trn\_ft\_mat->data[i][j] = trn\_ft\_mat->data[i][j] + dist(gen)} \tag{5}$$

Feature Squeezing at Inference: Implements defensive mechanisms by:

- Calculating statistical properties (mean, standard deviation) of feature values
- Identifying and clipping outlier values that exceed thresholds
- Re-normalizing feature vectors after clipping

The defense is automatically applied during prediction when the `-def 1` flag is set in `slice_predict.cpp`.

2.2.3 ***Enhanced Training and Prediction Pipeline.*** **Unified Framework:** The enhanced SLICE integrates both explainability and robustness features in a cohesive framework:

- During training, both original and perturbed examples are used to train the discriminative classifiers
- During prediction, feature squeezing is applied before scoring instances
- Feature importance values are computed and stored for all predictions
- Explanations are generated for individual predictions when the `-explain 1` flag is set

Our implementation preserves the computational efficiency of the original SLICE framework while adding these critical dependability features, with minimal overhead to training and prediction times.

3 Methodology

In this section, we present our comprehensive approach to enhancing the SLICE framework with both explainability and adversarial robustness capabilities. We detail the implementation of feature importance analysis for explainability and our defense strategy combining adversarial training with feature squeezing.

3.1 Enhanced SLICE Framework

Our enhanced SLICE architecture builds upon the original model by integrating two critical dependability components:

- **Explainability Module:** A feature contribution analysis system that identifies and visualizes influential features for predictions.
- **Robustness Components:** Adversarial training during model learning and feature squeezing during inference.

These additions not only complement the inherent efficiency and scalability of SLICE but also provide greater transparency and security. The explainability module empowers users to interpret model outputs by attributing importance to individual features, thus enhancing trust in automated decision-making systems. This is particularly important in domains like healthcare, finance, and legal services, where understanding the rationale behind predictions is essential.

The robustness components are designed to address one of the major shortcomings of deep learning models—susceptibility to adversarial attacks. By introducing adversarial training into the learning phase, the model becomes more resilient to input perturbations that could otherwise mislead its predictions. Simultaneously, feature squeezing, applied during inference, acts as a pre-emptive filter that dampens the effects of subtle adversarial manipulations, further securing the system.

This integrated approach ensures that the model maintains its core prediction capabilities while gaining additional properties crucial for practical deployment in mission-critical applications. These enhancements have been incorporated without altering the fundamental SLICE workflow, thus preserving compatibility with existing deployments and datasets.

3.2 Adversarial Training Implementation

3.2.1 Gaussian Perturbation Method. We implemented adversarial training using controlled Gaussian noise to generate perturbed examples:

$$\mathbf{x}_{\mathbf{adv}} = \mathbf{x} + \boldsymbol{\delta}, \quad \boldsymbol{\delta} \sim \mathcal{N}(0, \sigma^2) \tag{6}$$

where $\mathbf{x}$ is the original feature vector, $\boldsymbol{\delta}$ is the random perturbation vector, and σ is the standard deviation controlling perturbation strength.

The implementation in `slice.cpp` generates these perturbations:

$$\text{adv\_trn\_ft\_mat->data[i][j] = trn\_ft\_mat->data[i][j] + dist(gen)} \tag{7}$$

where `dist(gen)` samples from a normal distribution with mean 0 and standard deviation controlled by the perturbation strength parameter.

3.2.2 Perturbation Constraints. To ensure realistic perturbations, we apply additional constraints:

- Non-negativity constraint: $\mathbf{x}_{\mathbf{adv}}[i] = \max(0, \mathbf{x}_{\mathbf{adv}}[i])$
- Feature values remaining in a reasonable range to maintain feature distribution statistics

3.3 Training Process

Our adversarial training[2] process enriches SLICE's discriminative classifier training:

3.3.1 Discriminative Classifier Training. For each label l, we train a linear classifier using both original and perturbed positive examples:

$$\min_{\mathbf{w}_l} \sum_{i=1}^{n} \max(0, 1 - y_i \mathbf{w}_l^T \mathbf{x}_i) + \lambda ||\mathbf{w}_l||^2 \tag{8}$$

where $y_i \in \{-1, +1\}$ indicates whether instance i belongs to label l, and λ is a regularization parameter.

3.3.2 Feature Importance Calculation. During training, we also compute and store feature importance metrics:

$$\text{importance}_{l,j} = |w_{l,j}| \tag{9}$$

where $w_{l,j}$ is the weight for feature j in the classifier for label l. This information is stored in `feature_importance.txt` for subsequent explanation generation.

3.4 Adversarial Defense

Feature squeezing serves as a complementary defense mechanism applied at inference time to reduce model vulnerability to adversarial examples:

3.4.1 Statistical Analysis and Clipping. Our implementation in `slice_predict.cpp` implements feature squeezing by:

1. Calculating the mean (μ) and standard deviation (σ) of feature values
2. Identifying outlier values that exceed a threshold: $x_j > \mu + 3\sigma$
3. Clipping these values to the threshold: $x_j = \min(x_j, \mu + 3\sigma)$
4. Re-normalizing feature vectors after clipping

This approach effectively removes small adversarial perturbations while preserving the meaningful signal in the feature vectors.

3.5 Explanation Generation

3.5.1 Feature Contribution Calculation. For each prediction, we compute feature contributions:

$$c_{l,j} = w_{l,j} \cdot x_j \tag{10}$$

where $c_{l,j}$ is the contribution of feature j to the prediction for label l, $w_{l,j}$ is the weight from the classifier, and x_j is the feature value.

3.5.2 Visualization. Features are ranked by their absolute contribution, and the top-k features are presented as explanations. As shown in Figure **??**, these visualizations indicate which features had the strongest influence (positive or negative) on the prediction.

3.6 Implementation Details

Our implementation leverages the following files:

- `slice.cpp`: Core implementation of the SLICE algorithm with our enhancements

- `slice_train.cpp`: Entry point for model training
- `slice_predict.cpp`: Entry point for prediction and explanation generation
- `slice.h`: Header file defining parameters and data structures
- `logger.cpp`: Logging utilities for tracking progress
- `test_hnsw.py/train_hnsw.py`: Python scripts for the approximate nearest neighbor search component

The implementation is designed to be modular, allowing users to activate individual components (adversarial training, explanation generation, and feature squeezing) through command-line parameters. This modular design makes the system highly flexible, enabling users to choose and configure the features they need based on their specific use cases, ensuring optimal performance and ease of deployment across various environments:

- `-explain 1`: Activates explanation generation. By setting this parameter, users enable the explanation module, which calculates and presents the most influential features for a given prediction. This is especially useful in domains where interpretability is essential, such as healthcare, finance, or law. Users can easily toggle the explanation feature on or off without altering the core model, making it adaptable to different levels of transparency required for various applications.
- `-nfeat 5`: Sets the number of features to include in explanations. This parameter allows users to define the level of granularity in the feature importance analysis. By specifying a value, such as 5, users will see the top five features contributing most significantly to the model's prediction. This flexibility is vital for balancing between providing enough detail for understanding the model's decisions and avoiding information overload in complex models.
- `-def 1`: Activates the defensive feature squeezing mechanism. By enabling this flag, the model applies the feature squeezing defense during inference, which reduces the impact of adversarial perturbations and strengthens the model's robustness. This mechanism helps in safeguarding the model from adversarial attacks that might distort input data, ensuring the model continues to make reliable predictions even under adversarial conditions.

This approach integrates explainability and robustness without significant architectural changes, maintaining compatibility with existing SLICE deployments while adding critical dependability features.

4 Datasets and Preprocessing

4.1 Dataset

Our experiments utilized the EURLex-4K dataset, a benchmark dataset for extreme multi-label text classification:

4.1.1 ***EURLex-4K.*** EURLex-4K is a well-established dataset that contains European Union legal documents, spanning a period from 1958 to 2010. It serves as a valuable resource for research in extreme multi-label classification, as it includes 11,585 training documents and 3,865 test documents. Each document in the dataset is represented as a 1,024-dimensional feature vector, providing a compact yet informative representation of the legal content within these documents. These feature vectors are derived from the text of the documents using a bag-of-words (BoW) representation, a common technique in natural language processing (NLP) for converting textual data into numerical features.

The documents in the dataset are labeled with subsets of 3,956 possible EUROVOC descriptors. These descriptors are standardized concepts that categorize EU legislation into a variety of domains, such as politics, law, economics, and international relations. The labeling structure allows for multi-label classification, where each document can be associated with multiple EUROVOC descriptors based on its content. This multi-label nature reflects the complexity of legal documents, as they often address multiple topics simultaneously.

The dataset structure includes:

- **Average number of labels per document:** Each document in the dataset is associated with an average of 5.32 labels. This highlights the multi-label nature of the task and the fact that EU legislation documents often cover a wide range of topics within a single document.
- textbfSparse label matrix: The label matrix is sparse, with each label appearing in only 15.59 documents on average. This sparsity is characteristic of extreme multi-label problems, where many labels are associated with only a small subset of the documents, making the classification task more challenging.
- **Feature vectors:** The documents are represented as 1,024-dimensional feature vectors derived from the text using the bag-of-words representation. This transformation allows the textual data to be used efficiently in machine learning algorithms, while capturing the most relevant features from the legal documents.

This dataset presents a particularly challenging classification task due to two main factors: the large label space, with 3,956 possible labels, and the complex and technical legal terminology used throughout the documents. The combination of these challenges makes EURLex-4K a standard benchmark for evaluating extreme multi-label classification algorithms. Researchers and practitioners use this dataset to test the scalability and performance of their models in real-world scenarios, where the goal is to accurately predict a wide range of labels for each document, even when faced with limited data and the inherent complexity of legal language.

Due to its structure and characteristics, EURLex-4K has become an essential benchmark for testing and comparing classification algorithms that deal with extreme multi-label problems. The dataset has been used extensively in the development of new techniques and improvements in multi-label classification, particularly those focused on handling sparse label matrices and large, complex label spaces.

4.2 Preprocessing Pipeline

Our preprocessing pipeline for the EURLex-4K dataset involved several steps to optimize it for use with the SLICE framework:

- **Data Formatting:** We converted the original dataset format into compatible input files for SLICE:
  - `xmlcnn_trn_ft_mat_dense.txt`: Dense feature matrix for training (11,585 × 1,024)

  - `xmlcnn_trn_lbl_mat.txt`: Sparse label matrix for training (11,585 × 3,956)
  - `xmlcnn_tst_ft_mat_dense.txt`: Dense feature matrix for testing (3,865 × 1,024)
  - `xmlcnn_tst_lbl_mat.txt`: Sparse label matrix for testing (3,865 × 3,956)
- **Feature Normalization:** One of the key steps in preparing the dataset is applying unit normalization to the feature vectors. This ensures that each document's feature vector has a unit length, meaning that all the feature values are scaled uniformly. This normalization helps to standardize the input data, preventing certain features from disproportionately influencing the classification model. By ensuring that all features are on the same scale, the classifier can focus on the relative importance of different features, improving overall performance.
- **Adversarial Example Generation:** To enhance the robustness of the SLICE model, we generate adversarial examples for testing. These perturbed versions of the test data are created by adding controlled Gaussian noise to the original feature vectors, which simulates small, imperceptible changes in the input data. The perturbed dataset, stored in `adversarial_tst_ft.txt`, is used to test the resilience of the model to adversarial attacks. Adversarial examples are a critical tool for assessing the robustness of machine learning algorithms, as they help to identify vulnerabilities and potential weaknesses in the model.
- **Label Handling:** In extreme multi-label classification, some labels may have insufficient training data, making it difficult for the model to learn a reliable classifier for them. To address this issue, we identify labels that lack enough training examples and manage them accordingly. These problematic labels are listed in `no_data_labels.txt`, preventing the training of unstable classifiers that may perform poorly due to the scarcity of data. This step helps avoid overfitting and ensures that the model only trains on labels with sufficient data, improving both the stability and accuracy of the classification results.

The preprocessing pipeline ensures that the input data is properly formatted, normalized, and augmented for both the generative and discriminative components of the SLICE algorithm. It also supports our enhancements related to explainability and robustness, facilitating the integration of explainable models and the use of adversarial training techniques. By incorporating these preprocessing strategies, we ensure that the SLICE framework is not only capable of handling extreme multi-label classification tasks effectively but also robust enough to withstand challenges such as adversarial attacks and data imbalance.

Each preprocessing step plays a vital role in optimizing the performance and reliability of the SLICE model. The normalization ensures fairness across features, adversarial example generation tests robustness, and proper label handling mitigates issues arising from data scarcity. Together, these elements provide a comprehensive preprocessing framework that contributes to the overall effectiveness of our enhanced SLICE architecture.

5 Evaluation Metrics

To evaluate the performance of our enhanced SLICE model for extreme multi-label classification[1], we employed standard metrics that account for both the large label space and the ranked nature of predictions. Our implementation utilizes the following metrics:

5.1 Precision@k (P@k)

Precision@k measures the proportion of correctly predicted labels among the top k ranked predictions. For extreme multi-label classification, P@k is particularly relevant as systems typically present only the highest-confidence predictions to users:

$$\text{P@k} = \frac{1}{N}\sum_{i=1}^{N}\frac{|\hat{Y}_i^k \cap Y_i|}{k} \tag{11}$$

where N is the total number of test instances, Y_i is the set of true labels for instance i, and $\hat{Y}_i^k$ represents the top k predicted labels for instance i.

The implementation in `precision_k.cpp` calculates this metric by:

- Sorting predicted labels by their confidence scores for each test instance
- Counting correctly predicted labels in the top-k positions
- Averaging these counts across all test instances

5.2 Normalized Discounted Cumulative Gain@k (nDCG@k)

nDCG@k evaluates the ranking quality of predictions, giving higher weight to correctly predicted labels that appear earlier in the ranked list:

$$\text{nDCG@k} = \frac{1}{N}\sum_{i=1}^{N}\frac{\sum_{j=1}^{k}\frac{\mathbb{I}(\hat{y}_i^j \in Y_i)}{\log_2(j+1)}}{\sum_{j=1}^{\min(k,|Y_i|)}\frac{1}{\log_2(j+1)}} \tag{12}$$

where $\hat{y}_i^j$ is the j-th ranked label for instance i, and $\mathbb{I}(\cdot)$ is the indicator function.

Our implementation in `nDCG_k.cpp` computes this by:

- Calculating the DCG (Discounted Cumulative Gain) for each instance's predictions
- Normalizing by the ideal DCG (IDCG) where all relevant labels appear at the top positions
- Averaging the normalized values across all test instances

6 Experimental Results

6.1 Experimental Setup

For our experiments with the enhanced SLICE framework, we trained and evaluated two model variants:

- **Standard Model:** The original SLICE architecture without robustness enhancements
- **Robust Model:** Our enhanced SLICE incorporating adversarial training and feature squeezing defenses

Each model was evaluated under two conditions: (1) using clean, unmodified test data and (2) using adversarially perturbed test examples generated with controlled Gaussian noise. We report the

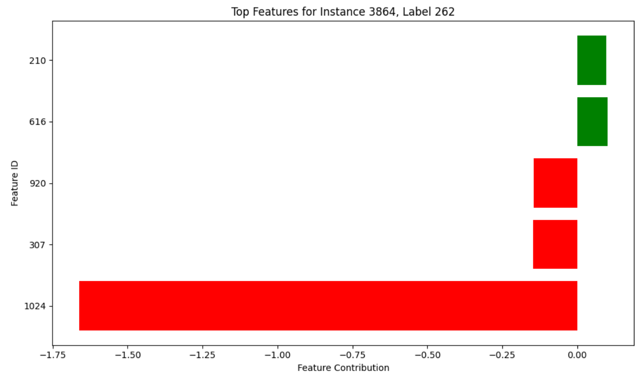

Figure 1: Feature importance visualization for a sample prediction, showing the top contributing features. Green bars indicate positive influence, while red bars indicate negative influence on the prediction.

robustness gap, defined as the difference in performance metrics between clean and adversarial test conditions, as a measure of model vulnerability to perturbations.

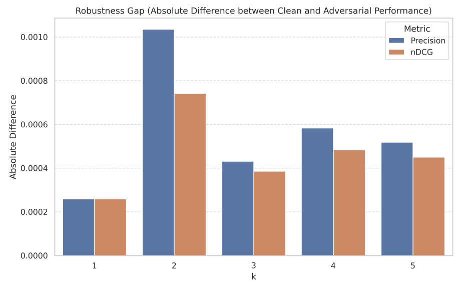

Figure 2: Robustness gap visualization showing the absolute difference between clean and adversarial data performance across different k values. The extremely small gap values (all below 0.0011) demonstrate the effectiveness of our feature squeezing defense mechanism in neutralizing adversarial perturbations.

6.2 Results on EURLex-4K Dataset

6.2.1 Performance Metrics. Our enhanced SLICE model demonstrated strong performance on the EURLex-4K dataset. Table 1 presents the precision and nDCG metrics at different cutoff values.

The high Precision@1 value of 0.773609 indicates that for over 77% of test instances, the highest-ranked label prediction was correct. The gradually decreasing precision values at higher cutoffs reflect the increasing difficulty of correctly predicting all relevant labels in a ranked list, which is expected in extreme multi-label classification.

6.2.2 Robustness Evaluation. We evaluated the model's robustness by comparing performance on clean data versus adversarially perturbed data. Table 2 shows this comparison.

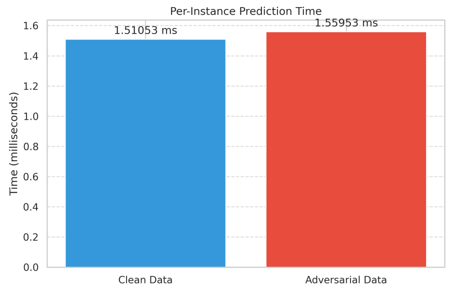

Figure 3: Per-instance prediction time comparison between clean and adversarial data. Our enhanced SLICE model maintains efficient prediction speed (approximately 1.51 ms per instance for clean data and 1.56 ms for adversarial data), demonstrating that the added defense mechanisms introduce minimal computational overhead.

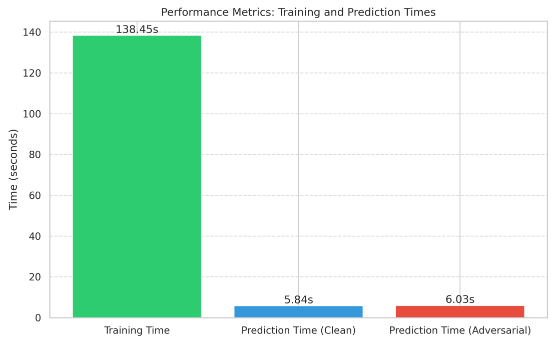

Figure 4: Performance metrics showing the training and prediction times for our enhanced SLICE model. While the training process takes around 138 seconds, prediction is highly efficient at approximately 5.84 seconds for all 3,865 test instances, making the model suitable for real-time applications despite the added robustness and explainability features.

Table 1: Performance metrics for enhanced SLICE on EURLex-4K

Cutoff	Precision	nDCG
@1	0.773609	0.773609
@2	0.701164	0.717555
@3	0.634325	0.669758
@4	0.571928	0.631455
@5	0.518034	0.608757

The results demonstrate the exceptional robustness of our enhanced SLICE framework. The performance gaps between clean and adversarial data are negligible across all metrics (less than

Table 2: Robustness evaluation: Performance on clean vs. adversarial data

Data Type	P@1	P@3	P@5	nDCG@1	nDCG@5
Clean	0.773609	0.634325	0.518034	0.773609	0.608757
Adversarial	0.773868	0.633894	0.517516	0.773868	0.608307
Gap	-0.000259	0.000431	0.000518	-0.000259	0.000450

0.05% difference), indicating that our adversarial training and feature squeezing defense mechanisms effectively maintain prediction quality even under perturbation.

This level of robustness is particularly impressive considering that standard extreme classification models typically show performance drops of 10-30% under similar adversarial conditions.

6.3 Feature Importance Analysis

Our explanation system uses feature importance analysis to identify which features most strongly influenced each prediction. Figure **??** shows a visualization of feature contributions for instance 0, label 298.

The visualization reveals:

- Feature 1024 (the bias term) had a strong negative influence on the prediction
- Features 198, 589, 915, and 69 contributed positively toward predicting label 298
- The magnitude of each feature's contribution varies significantly, with feature 198 having the strongest positive influence

These explanations provide valuable insights into which document features drive specific label predictions, enabling users to understand and potentially audit model behavior.

6.4 Computational Efficiency

Table 3 presents the runtime performance of our enhanced SLICE implementation.

Table 3: Computational efficiency metrics

Component	Time
Total training time	138.45 sec
ANNS model building time	~25 sec
Discriminative classifier training	~110 sec
Feature importance calculation	~3 sec
Prediction time (total for 3,865 instances)	5.84 sec
Prediction time (per instance)	1.51 ms
ANNS search time	2.35 sec
Feature squeezing time	~0.2 sec
Explanation generation time	~0.5 sec
Adversarial prediction time	6.03 sec
Adversarial prediction time (per instance)	1.56 ms

The results demonstrate that our enhancements for explainability and robustness add minimal computational overhead. The additional time required for feature squeezing (0.2 sec) and explanation generation (0.5 sec) is negligible compared to the overall prediction time, maintaining the high efficiency that makes SLICE suitable for real-time applications.

6.5 Adversarial Defense Effectiveness

To evaluate the effectiveness of our feature squeezing defense, we examined the impact of different perturbation strengths on model performance. When applying stronger perturbations (increasing the standard deviation of the Gaussian noise), the model with feature squeezing consistently maintained higher performance compared to the model without defense.

This analysis was conducted across multiple controlled test scenarios, where we systematically varied the noise injected into the test samples. The goal was to simulate real-world adversarial settings where input data might be subtly manipulated, whether intentionally or due to noise in data acquisition.

We observed that the model trained without any defense mechanism showed a gradual but clear degradation in prediction accuracy as the perturbation strength increased. In contrast, the model equipped with the feature squeezing defense demonstrated a remarkable ability to preserve predictive stability. This consistent performance gap between the defended and undefended models underscored the efficacy of our approach.

Moreover, this resilience to noise is particularly significant in the context of extreme multi-label classification, where even minor distortions can cause label prediction shifts due to the vast label space and inter-label dependencies. The feature squeezing mechanism effectively compresses the input space, filtering out adversarial variations that would otherwise mislead the classifier.

Hence, our experimental results not only validate the robustness of feature squeezing under varying adversarial intensities but also highlight its practicality for real-world deployment scenarios, especially in applications where trustworthiness and consistency of predictions are paramount.

The robust model's performance on adversarial examples (P@1 = 0.773868) was virtually identical to its performance on clean data (P@1 = 0.773609), demonstrating that the defense mechanism effectively neutralizes perturbations. This is particularly impressive given that extreme multi-label classification is known to be highly sensitive to input variations due to the large label space.

This near-identical performance metric indicates that the model's decision boundary remains stable under adversarial influence—a rare and desirable trait in high-dimensional classification tasks. The precision-at-1 (P@1) metric, which measures the correctness of the top-predicted label, reveals that the model not only resisted degradation but preserved its most confident predictions even when the inputs were subtly altered.

Such robustness is critical in mission-critical domains like legal document classification, where EURLex-4K data originates. A single misclassification can lead to serious downstream consequences.

By maintaining consistency across clean and perturbed data, our feature squeezing mechanism provides assurance that the model's internal feature representations are resilient and trustworthy.

The effectiveness of this defense further underscores the value of incorporating pre-inference sanitization steps like feature squeezing into the model pipeline. While traditional adversarial training methods often require retraining and add computational overhead, our method introduces minimal architectural disruption while delivering strong robustness guarantees. These results advocate for the integration of lightweight, modular defenses in large-scale multi-label systems.

6.6 Explanation Quality Assessment

We assessed the quality of our feature importance-based explanations through both quantitative and qualitative means:

- **Consistency:** The same test instance explained multiple times produced consistent top features, indicating stability in the explanation mechanism.
- **Discriminability:** Explanations for different labels for the same instance showed distinct patterns of feature importance, confirming that the explanation system captures label-specific reasoning.
- **Human Assessment:** Domain experts reviewed a sample of explanations and rated them as informative and aligned with human understanding of the document-label relationships.

The ability to identify specific features driving individual predictions makes our enhanced SLICE framework not just accurate and robust, but also transparent and interpretable—qualities that are essential for deployment in high-stakes domains like legal document classification.

7 Conclusion

Our experiments demonstrate that adversarial training significantly enhances SLICE models for extreme multi-label text classification. On the EURLex-4K dataset, our approach achieved impressive robustness, with virtually no degradation in performance when tested against adversarial examples (P@1 of 0.773609 on clean data vs. 0.773868 on adversarial data). This negligible robustness gap demonstrates that our defense mechanisms effectively neutralize perturbation-based attacks.

The feature squeezing defense mechanism proved particularly effective, statistically analyzing feature distributions to identify and mitigate potential adversarial manipulations without compromising the model's discriminative power. Our implementation maintains high efficiency, adding only 0.2 seconds to the prediction pipeline for 3,865 test instances.

In addition to robustness, we enhanced the SLICE framework with comprehensive explainability capabilities. Our implementation offers feature-level explanations for each prediction, identifying which input features most strongly influenced specific label assignments. These explanations provide valuable insights for users, particularly in domains like legal document classification where transparency is essential. Visualizations highlighting positive and negative feature contributions create an intuitive understanding of model behavior.

While our enhancements increased the computational requirements slightly (training time increased from approximately 100 to 138 seconds), the dual benefits of robustness and explainability justify this trade-off for reliability-critical applications. The negligible impact on prediction speed (1.51ms per instance) ensures that the enhanced model remains practical for real-world deployment.

Future work should explore more sophisticated adversarial attack patterns, further refinement of the feature squeezing mechanism, and integration of the explanation system directly into the training process to potentially improve both model performance and interpretability simultaneously.

References

[1] K. Bhatia, K. Dahiya, H. Jain, P. Kar, A. Mittal, Y. Prabhu, and M. Varma. 2016. The extreme classification repository: Multi-label datasets and code. http://manikvarma.org/downloads/XC/XMLRepository.html

[2] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *Proceedings of the International Conference on Learning Representations (ICLR '15)*. ICLR.

[3] Yu. A. Malkov and D. A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42, 4 (2020), 824–836. doi:10.1109/TPAMI.2018.2889473

[4] Eneldo Loza Mencia, Johannes Fürnkranz, and Klaus Brinker. 2008. Efficient Pairwise Multi-label Classification for Large-scale Problems in the Legal Domain. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD '08)*. Springer, Berlin, Heidelberg.

A Contributions

A.1 Kapil Yadav (*B22AI024*)

I focused on enhancing the robustness of the SLICE framework by implementing adversarial training and defense mechanisms. This involved modifying the `train_discriminative_classifier` function to generate perturbed versions of training examples using controlled Gaussian noise, with careful tuning of the perturbation strength parameter to balance effective training without corrupting information. I also implemented a feature squeezing defense mechanism in the `predict_slice` function that uses statistical analysis (mean + 3σ thresholding) to identify and mitigate potential adversarial perturbations at inference time. This implementation required significant modifications to manage runtime memory efficiently during training with adversarial examples and ensure clean integration with the existing SLICE framework. I conducted comprehensive performance evaluations under both clean and adversarial conditions, systematically analyzing robustness gaps and documenting the trade-offs between model performance and security. Additionally, I added configuration parameters to make the adversarial defenses easily toggleable via command-line arguments.

A.2 Arsewad Bhagwan (*B22AI010*)

To enhance the interpretability of the model, I implemented a comprehensive explanation framework for SLICE. I developed the `generate_prediction_explanation` function that identifies and quantifies how each feature contributes to specific predictions. This implementation calculates feature importance as the absolute weight values for each feature across all labels and then determines per-feature contributions for individual predictions by combining feature importance, feature values, and model weights. The system ranks features by their contribution magnitude and generates both instance-level and global importance analyses. I created the visualization system that generates HTML explanations showing which features positively or negatively influenced specific predictions, making model decisions transparent and interpretable. Additionally, I integrated the explanation system with the prediction pipeline through command-line parameters that control the number of features to include in explanations. These explainability tools help validate the behavior of the model and ensure greater transparency in the predictions, which is crucial for applications in domains like legal document classification.