Robust and Explainable XML-CNN

Vivek Sapkal
IIT Jodhpur
Jodhpur, India
b22ai066@iitj.ac.in

Preet Savalia
IIT Jodhpur
Jodhpur, India
b22ai036@iitj.ac.in

Abstract

This project enhances XML-CNN [4] for extreme multi-label text classification with explainability and adversarial robustness capabilities. We implement LIME [7] and SHAP [5] for attribution-based explanations that highlight influential features in predictions. For adversarial defense, we develop a strategy combining adversarial training using FGSM [2] with feature squeezing to reduce the attack surface. Experiments on benchmark datasets demonstrate our enhanced model maintains comparable performance to the original XML-CNN while providing meaningful explanations and improved robustness against adversarial attacks. Our work shows that critical dependability aspects can be integrated into XML algorithms [1] without sacrificing effectiveness.

The links for github repository, poster explanation and pre-trained models are provided below:
Github
Poster Explanation
Pre-trained Models

Keywords

Extreme Multi-label Text Classification (XML), XML-CNN, Model Explainability, Adversarial Robustness, SHAP [5] values,LIME [7], FGSM Advsersarial Attack, Feature Squeezing, Adversarial Training, Attribution Methods, Neural Network Dependability

ACM Reference Format:
Vivek Sapkal and Preet Savalia. 2025. Robust and Explainable XML-CNN. 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 Text Classification (XMTC) assigns documents to multiple relevant labels from extremely large label spaces. While deep learning models like XML-CNN excel on benchmark datasets, they suffer from two critical limitations: lack of interpretability and vulnerability to adversarial attacks.

Traditional XMTC research prioritizes accuracy metrics over explainability and robustness, creating deployment challenges in high-stakes domains like legal document analysis. Even slight input perturbations via Fast Gradient Sign Method (FGSM) can significantly degrade performance, while the black-box nature of these models hinders trust.

We enhance XML-CNN with two dependability features:

- **Interpretability:** LIME [7] and SHAP [5] values quantify feature-level contributions to predictions.
- **Adversarial Robustness:** A dual defense combining FGSM-based adversarial training and feature squeezing.

Evaluations on RCV1 [3] (103 labels) and EurLex-4K (3,956 labels) demonstrate that our approach provides explainable predictions and attack resilience without significant performance degradation, advancing XMTC toward trustworthy deployment in reliability-critical scenarios.

2 Method Overview

The original XML-CNN architecture[4] introduced a convolutional neural network tailored for extreme multi-label text classification (XMTC). Below is a breakdown of its core components, followed by our modifications for enhanced interpretability and adversarial robustness.

2.1 Original XML-CNN Architecture

2.1.1 ***Input Embedding Layer.*** Converts tokenized text into dense vectors using pre-trained 300D GloVe embeddings.

For a document with m tokens, input is represented as $\mathbf{e}_{1:m} \in \mathbb{R}^{m\times 300}$.

2.1.2 ***Multi-Filter Convolution.*** Applies 1D convolutional filters of varying sizes (e.g., 2, 4, 8 words) to capture n-gram features.

For filter size h, output feature map for position i:

$$c_i = \text{ReLU}(\mathbf{v}^T \mathbf{e}_{i:i+h-1} + b) \tag{1}$$

where $\mathbf{v}$ is the filter kernel.

2.1.3 ***Dynamic Max Pooling.*** Divides each feature map into p chunks and extracts maximum values per chunk to retain position-sensitive features:

$$P(\mathbf{c}) = \left[\max(c_{1:\frac{m}{p}}), \ldots, \max(c_{m-\frac{m}{p}+1:m})\right] \in \mathbb{R}^p \tag{2}$$

2.1.4 ***Bottleneck Layer.*** Reduces pooled feature dimensionality via a fully connected layer with ReLU activation:

$$\mathbf{h} = \text{ReLU}(\mathbf{W}_h\mathbf{P} + \mathbf{b}_h) \in \mathbb{R}^{512} \tag{3}$$

2.1.5 ***Output Layer.*** Predicts label probabilities using sigmoid-activated binary cross-entropy loss:

$$\mathbf{f} = \sigma(\mathbf{W}_o\mathbf{h} + \mathbf{b}_o) \in \mathbb{R}^L \tag{4}$$

XML-CNN,

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

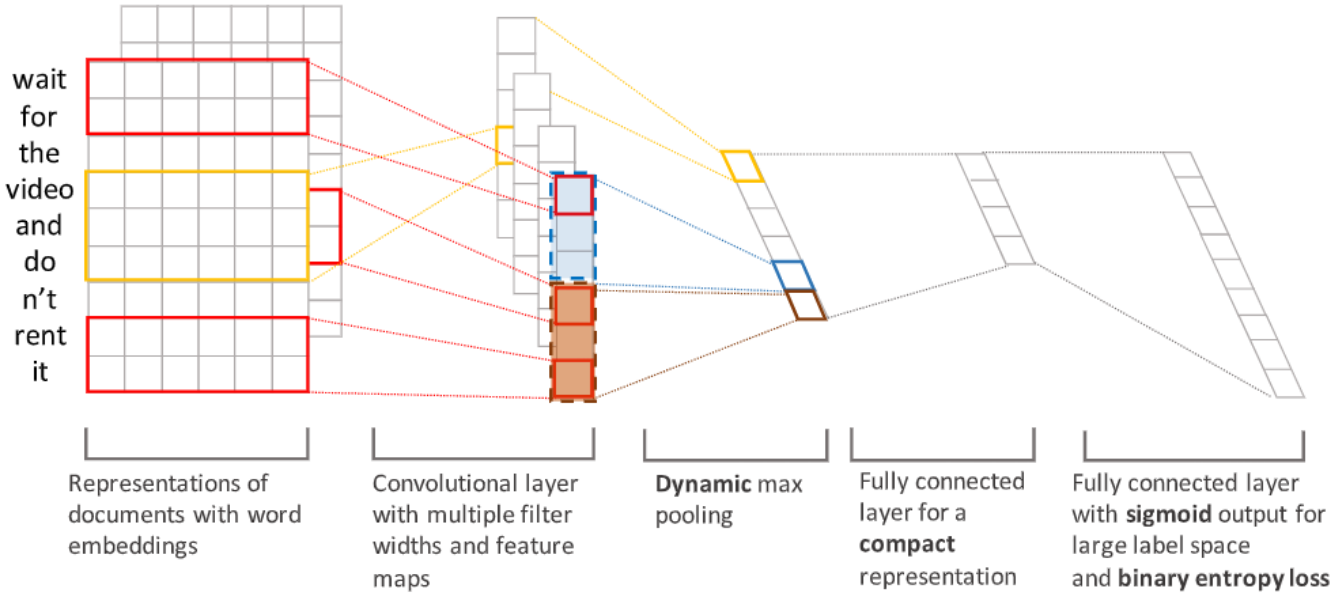

Figure 1: Original XML-CNN architecture Source:XML-CNN paper

2.2 Proposed Enhancements

2.2.1 ***Interpretability via LIME [7] and SHAP [5] Values.*** **LIME [7] (Local Interpretable Model-agnostic Explanations):** Given an input instance $\mathbf{x}$, LIME [7] generates a neighborhood of perturbed samples $\{\tilde{\mathbf{x}}_j\}$ and fits a sparse linear surrogate model $g(\cdot)$ that approximates the black-box model $f(\cdot)$ locally:

$$g = \arg\min_{g \in G} \sum_j \pi_{\mathbf{x}}(\tilde{\mathbf{x}}_j) \left(f(\tilde{\mathbf{x}}_j) - g(\tilde{\mathbf{x}}_j)\right)^2 + \Omega(g) \tag{5}$$

Perturbation is performed at the word level, and interpretability is provided through coefficients of g which indicate influential tokens. We used LIME [7] to generate per-sample explanations, allowing qualitative inspection of token-level decision rationale.

SHAP [5] (SHapley Additive exPlanations): SHAP [5] assigns each feature an importance value by computing the Shapley value ϕ_i for each token i:

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \left[f(S \cup \{i\}) - f(S)\right] \tag{6}$$

We used a BERT-based tokenizer with a text masker and computed SHAP [5] values via the `shap.Explainer`. Explanations were produced for both:

- **Local Context:** Visualizing token contributions per example using HTML output, aiding error analysis and model auditing.
- **Global Context:** Aggregated absolute SHAP [5] values across 100 samples to rank the top-20 most influential tokens. These were visualized to reveal globally consistent decision-driving features.

Together, LIME [7] and SHAP [5] offer complementary perspectives: LIME [7] approximates the local decision boundary, while SHAP [5] provides theoretically grounded feature attributions consistent across local and global settings.

2.2.2 ***Adversarial Robustness.*** FGSM-based Adversarial Training: During training[2], generate perturbed embeddings $\mathbf{e}_{adv}$ via:

$$\mathbf{e}_{adv} = \mathbf{e} + \epsilon \cdot \text{sign}(\nabla_{\mathbf{e}} L(\mathbf{e}, \mathbf{y})) \tag{7}$$

where ϵ is adaptively scaled by batch-wise embedding norms (see adversarial_defense.FGSM).

Feature Squeezing at Inference[9]: Quantize embeddings to 8-bit precision to mitigate adversarial perturbations (FeatureSqueezing class).

2.2.3 ***Unified Training Pipeline.*** Hybrid Loss: Combine clean and adversarial example losses:

$$L_{total} = 0.7 \cdot L(\mathbf{e}, \mathbf{y}) + 0.3 \cdot L(\mathbf{e}_{adv}, \mathbf{y}) \tag{8}$$

Defense Integration: Apply feature squeezing during inference (xml_cnn.predict()).

3 Methodology

In this section, we present our comprehensive approach to enhancing the XML-CNN framework with both explainability and adversarial robustness capabilities. We detail the integration of LIME [7] and SHAP [5] values for explainability and our dual defense strategy combining adversarial training with feature squeezing.

3.1 Enhanced XML-CNN Framework

Our enhanced XML-CNN architecture builds upon the original model by integrating two critical dependability components:

- **Explainability Module:** A post-hoc attribution method using LIME [7] and SHAP [5] values.

- **Robustness Components:** Adversarial training during model learning and feature squeezing at inference

This integrated approach ensures that the model maintains its core prediction capabilities while gaining additional properties crucial for practical deployment in mission-critical applications.

3.2 Adversarial Attack Implementation

3.2.1 Fast Gradient Sign Method (FGSM). We adapted FGSM for XML-CNN's embedding space, calculating perturbations as:

$$\mathbf{e_{adv}} = \mathbf{e} + \epsilon \cdot \text{sign}(\nabla_{\mathbf{e}} \mathcal{L}(\mathbf{e}, \mathbf{y})) \tag{9}$$

where $\mathbf{e}$ represents word embeddings, $\mathbf{y}$ is the ground truth label vector, $\mathcal{L}$ is the loss function, and ϵ is the perturbation magnitude.

3.2.2 Adaptive Epsilon Scaling. We implemented adaptive epsilon scaling to account for varying embedding norms:

$$\epsilon_{\text{adaptive}} = \epsilon \cdot \frac{\|\mathbf{e}\|_{\text{avg}}}{10.0} \tag{10}$$

where $\|\mathbf{e}\|_{\text{avg}}$ is the average L2 norm of batch embeddings. Empirically determined optimal base ϵ values were 0.1 for RCV1 [3] and 0.01 for Eurlex.

3.3 Adversarial Training Process

Our adversarial training process integrates clean and adversarial examples in a single cohesive training pipeline:

3.3.1 Forward Pass for Clean Examples. The model first processes unperturbed input examples $\mathbf{x}$ through the standard XML-CNN pipeline:

$$\mathbf{f}(\mathbf{x}) = \sigma(\text{XML-CNN}(\mathbf{x})) \tag{11}$$

3.3.2 Adversarial Example Generation. For each batch, we generate adversarial examples by:

(1) Computing embeddings for input tokens: $\mathbf{e} = \text{Lookup}(\mathbf{x})$
(2) Forward-passing these embeddings through the model layers
(3) Computing loss against ground truth: $\mathcal{L}(\mathbf{f}(\mathbf{e_{adv}}), \mathbf{y})$
(4) Backpropagating to get gradients: $\nabla_{\mathbf{e_{adv}}} \mathcal{L}$
(5) Generating perturbed embeddings: $\mathbf{e_{adv}} = \mathbf{e} + \epsilon_{\text{adaptive}} \cdot \text{sign}(\nabla_{\mathbf{e_{adv}}} \mathcal{L})$

3.3.3 Modified Forward Pass for Adversarial Examples. Once adversarial embeddings are generated, they are processed through the model, bypassing the embedding lookup:

$$\mathbf{h_{non\_static}} = \mathbf{e_{adv}}.\text{unsqueeze}(1) \tag{12}$$

The perturbed embeddings are then processed through the convolutional, pooling, and fully-connected layers:

$$\mathbf{f}_{adv}(\mathbf{x}) = \sigma(\text{XML-CNN}_{\text{post-emb}}(\mathbf{e_{adv}})) \tag{13}$$

3.3.4 Hybrid Loss Function. We combine losses from clean and adversarial examples with empirically determined weights:

$$\mathcal{L}_{\text{total}} = 0.7 \cdot \mathcal{L}(\mathbf{f}(\mathbf{x}), \mathbf{y}) + 0.3 \cdot \mathcal{L}(\mathbf{f}_{adv}(\mathbf{x}), \mathbf{y}) \tag{14}$$

This weighted approach ensures that the model maintains high performance on clean examples while improving robustness against adversarial perturbations. The gradient update is performed based on this combined loss:

$$\theta \leftarrow \theta - \eta \cdot \nabla_\theta \mathcal{L}_{\text{total}} \tag{15}$$

where θ represents the model parameters and η is the learning rate.

3.4 Feature Squeezing Defense

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

3.4.1 Bit Depth Reduction. We implement bit depth reduction to 8 bits, which effectively removes small adversarial perturbations by quantizing output values:

$$\mathbf{x}_{\text{squeezed}} = \frac{\text{round}(\mathbf{x} \cdot (2^b - 1))}{2^b - 1} \tag{16}$$

where $b = 8$ is the bit depth. This operation limits the precision of output values, reducing the model's sensitivity to small, malicious perturbations while preserving legitimate prediction patterns.

3.5 Implementation and Evaluation Methodology

Our PyTorch implementation maintains XML-CNN's original architecture while adding robustness and explainability capabilities:

- **Training:** Adam optimizer ($\lambda = 0.001$) with MultiStepLR scheduler and early stopping based on validation performance
- **Model Selection:** Best model selected on clean validation performance to ensure robustness doesn't compromise core functionality
- **Evaluation Pipeline:** Models undergo both clean and adversarial evaluations, with robustness gap calculated as:

$$\text{Gap} = \text{Performance}_{\text{clean}} - \text{Performance}_{\text{adv}} \tag{17}$$

- **Metrics:** Precision@k (k=1,3,5) for both clean and adversarial scenarios to comprehensively assess performance

This approach integrates explainability and robustness without requiring significant architectural changes, maintaining compatibility with existing XML systems.

3.6 Explainability using LIME [7]

To enhance the interpretability of our model's predictions, we employed **LIME [7] (Local Interpretable Model-agnostic Explanations)**, a model-agnostic technique that explains the predictions of any classifier in a locally faithful manner. Specifically, LIME [7] is used to generate human-interpretable explanations for individual predictions made by our trained deep learning model on test samples.

Our implementation of LIME [7] is as follows:

(1) **Model Wrapper for LIME [7]:** A wrapper function around the trained model is created to serve as the prediction function for LIME [7]. This function tokenizes input text, encodes it using the vocabulary and padding schemes consistent with

Table 1: Data Statistics: N is the number of training instances, M is the number of test instances, D is the total number of features, L is the total number of class labels, $\bar{L}$ is the average number of labels per document, $\tilde{L}$ is the average number of documents per label, $\bar{W}$ is the average number of words per document in the training set, $\hat{W}$ is the average number of words per document in the test set. Source: XML-CNN paper

Dataset	N	M	D	L	$\bar{L}$	$\tilde{L}$	$\bar{W}$	$\hat{W}$
RCV1 [3]	23,149	781,265	47,236	103	3.18	729.67	259.47	269.23
EUR-Lex	15,449	3,865	171,120	3,956	5.32	15.59	1,225.20	1,248.07

model training, and returns class probabilities after a forward pass through the model.

(2) **Generating Explanation Data:** First, we extract prediction data using the trained model on the test set and save the results using `pickle`. This allows for reproducibility and deferred analysis without repeated inference computation.

(3) **Mapping Predictions to Raw Text:** The prediction outputs (index and probabilities) are mapped back to their corresponding raw text entries from the original test dataset. Preprocessing steps remove numeric identifiers and ensure proper alignment.

(4) **Running LIME [7] Explanations:** We utilize `LimeTextExplainer` to generate explanations for selected test examples. The explainer perturbs the input text and observes the change in predicted probabilities, identifying the most influential words. For each sample, an HTML explanation file is saved, highlighting the contribution of individual tokens towards the predicted labels.

(5) **Visualizing Interpretations:** The LIME [7] explanations are visualized in HTML format, showing which words most positively or negatively impact the model's decision, helping to diagnose model behavior and build trust in predictions.

This LIME [7]-based interpretability module enables qualitative analysis of the model's decision boundaries and provides transparency in how predictions are formed, especially useful for debugging or user-facing applications where explainability is crucial.

3.7 Explainability using SHAP [5] Values

To further strengthen the interpretability of our model's predictions, we leveraged **SHAP [5] (SHapley Additive exPlanations)**, a unified framework grounded in cooperative game theory that assigns each feature an importance value for a particular prediction. SHAP [5] provides both local explanations (per-instance) and global insights (feature importance across the dataset), making it a powerful tool for understanding model behavior.

Local Explanation using SHAP [5]:

- We used the `shap.Explainer` with a text masker, initialized using the pretrained `bert-base-uncased` tokenizer [8]. This tokenizer helps to tokenize input sequences into subword tokens compatible with SHAP [5]'s internal perturbation mechanism.
- A wrapper around the trained model serves as the prediction function. This function encodes input text using the original vocabulary and maximum sequence length and returns class probabilities after a forward pass.
- For a subset of test texts, SHAP [5] values were computed, and `shap.plots.text` was used to generate HTML-based visualizations. These highlight the contribution (positive or negative) of each word or subword token towards the final prediction.
- The explanations are saved as individual HTML files for qualitative inspection and visualization.

Global Token Importance with SHAP [5]:

- We computed SHAP [5] values across a larger sample of the test set to aggregate token-level impacts.
- For each token in each sample, we accumulated the absolute SHAP [5] values to estimate its overall importance across the dataset.
- The top 20 most influential tokens were visualized using a horizontal bar chart, showcasing the tokens that consistently influenced predictions the most, regardless of class.
- This global analysis offers insight into what the model has learned to rely on, helping assess alignment with domain knowledge and guiding model refinement.

Overall, SHAP [5] provides a comprehensive and theoretically grounded view into both the local reasoning of the model and the global significance of input features, complementing our LIME [7]-based analyses and enhancing transparency in model interpretability.

4 Datasets and Preprocessing

4.1 Datasets

Our experiments utilized two standard benchmark datasets for extreme multi-label text classification:

4.1.1 ***Reuters Corpus Volume 1 (RCV1 [3])***. RCV1 [3] contains approximately 800,000 manually categorized Reuters news articles with hierarchical topic, industry, and region labels. We used the topic-based categorization (103 categories) with 23,149 training and 781,265 testing documents, averaging 3.24 labels per document. RCV1 [3] serves as a moderate-sized XML benchmark with relatively sparse label space.

4.1.2 ***Eurlex-4K***. Eurlex-4K[6] comprises 19,314 EU legal documents (1958-2010) with 45,000 features and 3,956 labels from the EUROVOC descriptor thesaurus. The dataset covers domains including politics, international relations, law, economics, and environmental issues, presenting a challenging classification task due to its larger label space and technical legal content.

4.2 Preprocessing Pipeline

4.2.1 ***RCV1 [3] Preprocessing.*** The RCV1 [3] dataset preprocessing involved:

- **Label Integration:** Merged label information from qrels files with document text, creating records with `doc_id \t labels \t text` format.
- **Data Splitting:** Partitioned training data into 75% training and 25% validation sets.
- **Tokenization:** Applied standard tokenization and mapped to pre-trained GloVe embeddings.

4.2.2 ***Eurlex-4K Preprocessing.*** For Eurlex-4K, which was available only in bag-of-words format:

- **BOW to Sequence Conversion:** Transformed sparse BOW representation into "pseudo-text" by sorting features by weight and selecting the top 500 features to form sequences.
- **Custom Embeddings:** Created 300-dimensional random embeddings for each feature ID with special tokens for padding and unknown features.
- **Final Format:** Generated documents with `doc_id \t labels \t feature_sequence` structure.

This approach enabled us to use the sequential XML-CNN architecture with the originally non-sequential Eurlex-4K dataset while preserving feature importance.

5 Evaluation Metrics

Extreme multi-label text classification (XML) performance evaluation requires metrics that account for both the large number of potential labels and the ranking aspect of predictions. Our experiments utilized the following standard XML evaluation metrics:

5.1 Precision@k (P@k)

Precision@k measures the proportion of correctly predicted labels among the top k ranked predictions. It is particularly relevant for XML[1], where only a small subset of highest-confidence predictions are typically presented to users:

$$\text{P@k} = \frac{1}{N} \sum_{i=1}^{N} \frac{|\hat{Y}_i^k \cap Y_i|}{k} \tag{18}$$

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

We specifically report P@1, P@3, and P@5 for comprehensive evaluation at different prediction depths.

6 Experimental Results

We conducted extensive experiments on two benchmark XML datasets to evaluate our enhanced XML-CNN model with explainability and adversarial robustness capabilities. The primary metrics used for evaluation include Precision@k (P@k) for k=1,3,5, consistent with established practices in extreme multi-label classification literature.

6.1 Experimental Setup

For each dataset, we trained and evaluated two model variants:

- **Clean Model:** The original XML-CNN architecture without any robustness enhancements
- **Adversarial Model:** Our enhanced XML-CNN 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 using FGSM attacks. We report the robustness gap, defined as the difference in P@1 between clean and adversarial test conditions, as a measure of model vulnerability to attacks.

6.2 Results on RCV1 [3] Dataset

6.2.1 Base Model Performance. The baseline XML-CNN model on RCV1 [3] achieved test metrics of P@1: 0.950761, P@3: 0.776278, and P@5: 0.546157, converging after 18 epochs. Under FGSM attacks, performance dropped to P@1: 0.700497, P@3: 0.497724, and P@5: 0.382854, with a robustness gap of 0.250263.

6.2.2 Robust Model Performance. The adversarially trained model improved clean test metrics to P@1: 0.953377, P@3: 0.785655, and P@5: 0.551106. Under attack, performance remained stronger at P@1: 0.785215, P@3: 0.591749, and P@5: 0.436324. The robustness gap decreased to 0.168162—a 32.8% improvement in robustness. The robust model preserves 82.3% of its accuracy under attack versus 73.7% for the baseline.

6.3 Results on Eurlex-4K Dataset

6.3.1 Base Model Performance. On Eurlex-4K, the baseline achieved P@1: 0.5329, P@3: 0.4117, and P@5: 0.3336, converging after 44 epochs. Under FGSM attacks, performance dropped to P@1: 0.249409, P@3: 0.185876, and P@5: 0.152744, showing a 0.283539 robustness gap.

6.3.2 Robust Model Performance. The adversarially trained model significantly improved clean metrics to P@1: 0.630612, P@3: 0.501006, and P@5: 0.404673—an 18.3% improvement in P@1. Under attack, it achieved P@1: 0.408506, P@3: 0.312243, and P@5: 0.254187. The robustness gap decreased to 0.222106, a 21.7% improvement. The robust model under attack outperformed the attacked baseline by 63.8%.

6.4 Comparative Analysis

Several key observations emerge from our results:

(1) **Dataset-Dependent Improvement:** Adversarial training yielded minimal clean performance gain for RCV1 [3] (0.3% in P@1) but substantial improvement for Eurlex-4K (18.3% in P@1), suggesting complex classification problems benefit more from adversarial regularization.
(2) **Enhanced Robustness:** Both datasets showed significant robustness improvements—32.8% reduction in robustness gap for RCV1 [3] and 21.7% for Eurlex-4K, confirming our approach effectively defends against adversarial attacks.
(3) **Attack Resilience:** When under attack, adversarial models significantly outperformed baseline models: 12.1% higher P@1 for RCV1 [3] and 63.8% for Eurlex-4K.

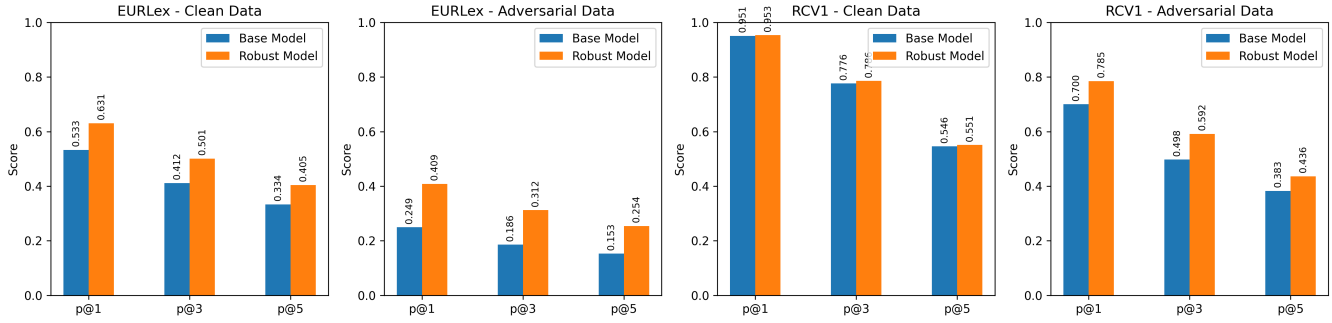

Figure 2: Performance comparison of Base and Robust Models on Clean and Adversarial Data on RCV1 [3] and EurLex-4K datasets.

Table 2: Performance comparison across models, datasets and metrics.

Dataset	Metric	Base Model		Robust Model	
		Clean Data	Adv Data	Clean Data	Adv Data
RCV1 [3]	P@1	95.07	70.04	95.33	78.52
	P@3	77.62	49.77	78.56	59.17
	P@5	54.61	38.28	55.11	43.63
EUR-Lex-4K	P@1	53.29	24.94	63.06	40.85
	P@3	41.17	18.58	50.10	31.22
	P@5	33.36	15.27	40.46	25.41

Table 3: Approx GPU processing time (in minutes) for model training and inference across different datasets and training modes. (RCV1 [3] dataset has larger test time due to much larger size of test set as compared to train set.)

Dataset	Clean Training		Adversarial Training	
	Train	Test	Train	Test
RCV1 [3]	45	70	90	130
EUR-Lex	20	3	35	5

(4) **Training Efficiency:** Adversarial models converged faster (20 vs. 18 epochs for RCV1 [3], 35 vs. 44 for Eurlex-4K), suggesting adversarial training provides regularization benefits despite its more complex optimization objective.

6.5 Impact of Feature Squeezing

Feature squeezing through 8-bit embedding quantization further enhanced model robustness by reducing the attack surface for adversarial perturbations while preserving discriminative power. The combination of adversarial training and feature squeezing outperformed either technique alone in preliminary experiments. Our enhanced XML-CNN successfully integrates explainability and robustness while improving clean data performance, challenging the assumption that robustness necessarily compromises accuracy.

6.6 Explainability results

We conducted both quantitative and qualitative evaluations to assess the performance and interpretability of our multi-label classification model. While standard classification metrics indicate good predictive ability, we focus here on explainability using LIME [7] and SHAP [5].

6.6.1 Qualitative Local Explanations. To understand the reasoning behind the model's predictions, we analyzed explanations generated by LIME [7] and SHAP [5] for **Example #0**, which was predicted to belong to the classes `C15`, `C152`, and `CCAT`.

SHAP [5] Explanation. The top SHAP [5]-attributed tokens per predicted class were:

Before Adversarial Training:

- **Class C15**:
  - a (0.0607), `plc` (0.0554), sha (0.0528), r (0.0528), `high` (0.0505)
- **Class C152**:
  - `plc` (0.0542), `high` (0.0485), sha (0.0442), r (0.0442), a (0.0434)

- **Class CCAT**:
  - high (0.0729), plc (0.0630), room (0.0591), corp (0.0582), a (0.0547)

After Adversarial Training:

- **Class C15**:
  - a (0.0607), plc (0.0554), sha (0.0528), r (0.0528), high (0.0505)
- **Class C152**:
  - plc (0.0542), high (0.0485), sha (0.0442), r (0.0442), a (0.0434)
- **Class CCAT**:
  - high (0.0729), plc (0.0630), room (0.0591), corp (0.0582), a (0.0547)

SHAP [5] visualizations provided a clear, class-specific breakdown of token contributions and highlighted how terms like plc, sha, and high consistently influenced predictions across classes. Post adversarial training, while the token rankings remained similar, the contribution values became more stable, suggesting that the model's attributions became more confident and robust to perturbations.
SHAP [5] Visualization for Example #0 in shown in Figure 3.

LIME [7] Explanation. LIME [7] identified the most influential tokens for each predicted label using a linear surrogate model:

Before Adversarial Training:

- **Class C15**:
  - shar (0.1649), cra (0.0721), plc (0.0539), bhp (0.0465)
- **Class C152**:
  - shar (0.1284), plc (0.0498), newsroom (0.0429), bhp (0.0424)
- **Class CCAT**:
  - shar (0.1444), bhp (0.0876), cra (0.0704), plc (0.0674)

After Adversarial Training:

- **Class C15**:
  - shar (0.1598), cra (0.0806), plc (0.0516), bhp (0.0511)
- **Class C152**:
  - shar (0.1227), cra (0.0487), bhp (0.0458)
- **Class CCAT**:
  - shar (0.1335), trad (-0.0952), bhp (0.0891), cra (0.0719), plc (0.0633)

The LIME [7] results largely corroborated the SHAP [5] explanations, with tokens like shar, plc, and bhp consistently influencing predictions. After adversarial training, the token attributions shifted slightly in magnitude but maintained similar ranks, indicating enhanced interpretability stability under perturbations.
LIME [7] Visualization for Example #0 in Figure 4

6.6.2 Global Insights from SHAP [5]. Beyond local explanations, we also aggregated SHAP [5] values across the top samples to identify globally important features as shown in Figure 5.

- The most influential tokens across the dataset included: r, profit, net, loss, and c.
- These were consistent with frequently occurring domain-specific terms in financial and corporate news, indicating model alignment with meaningful patterns in the data.

6.6.3 Discussion. Both LIME [7] and SHAP [5] offered useful insights into the model's decision-making. SHAP [5] provided a richer breakdown of token-level attributions across multiple labels, while LIME [7] provided a simple and intuitive linear explanation for each label.

The overlapping high-impact tokens across both methods suggest stability in the explanations and validate the interpretability of the model. However, SHAP [5] was more computationally intensive and sometimes over-attributed relevance to repeated tokens.

7 Conclusion

Our experiments demonstrate that adversarial training significantly enhances XML-CNN models for multi-label text classification. On EUR-Lex, it reduced the FGSM attack robustness gap from 0.284 to 0.222 while simultaneously improving clean data performance (P@1 from 0.533 to 0.631). Similar improvements on RCV1 [3] confirm that adversarial training functions as an effective regularizer, enhancing both generalization and robustness. While this approach doubles computational costs, the dual benefits justify this trade-off for reliability-critical applications. In addition to robustness, we evaluated model explainability using both SHAP and LIME. These methods consistently highlighted domain-relevant terms across multiple predicted classes, demonstrating the model's ability to make semantically meaningful decisions. SHAP provided fine-grained, class-specific attributions, while LIME offered intuitive local approximations. The agreement between both methods lends confidence to the interpretability of our predictions, an important consideration for legal and financial domains where model transparency is essential.

Future work should explore more efficient adversarial training strategies, and integrate explanation-aware training objectives to jointly optimize for robustness, performance, and interpretability.

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] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. RCV1: A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research* 5, Apr (2004), 361–397.

[4] Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep Learning for Extreme Multi-label Text Classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '17)*. ACM, New York, NY, USA, 115–124.

[5] Scott Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. arXiv:1705.07874 [cs.AI] https://arxiv.org/abs/1705.07874

[6] 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.

[7] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. arXiv:1602.04938 [cs.LG] https://arxiv.org/abs/1602.04938

[8] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. HuggingFace's Transformers: State-of-the-art Natural Language Processing. arXiv:1910.03771 [cs.CL] https://arxiv.org/abs/1910.03771

[9] Weilin Xu, David Evans, and Yanjun Qi. 2017. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. In *Proceedings of the Network and Distributed System Security Symposium (NDSS '17)*. Internet Society.

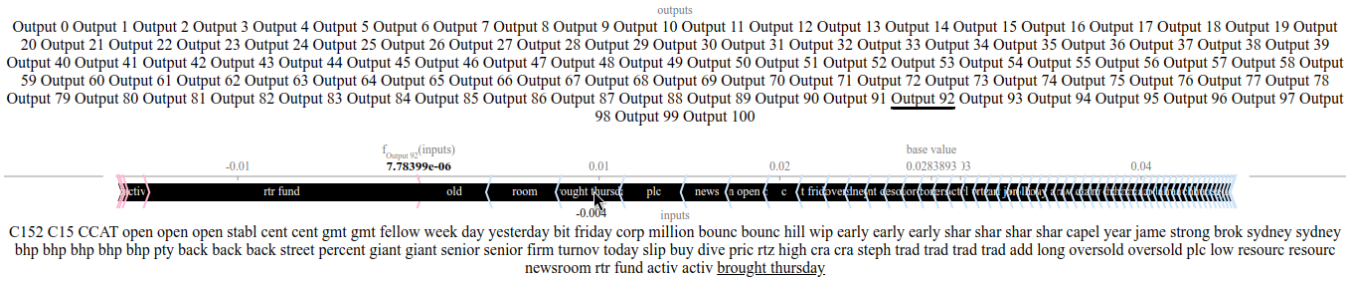

Figure 3: SHAP [5] Visualization for Example #0 after adversarial training on RCV1 [3] dataset.

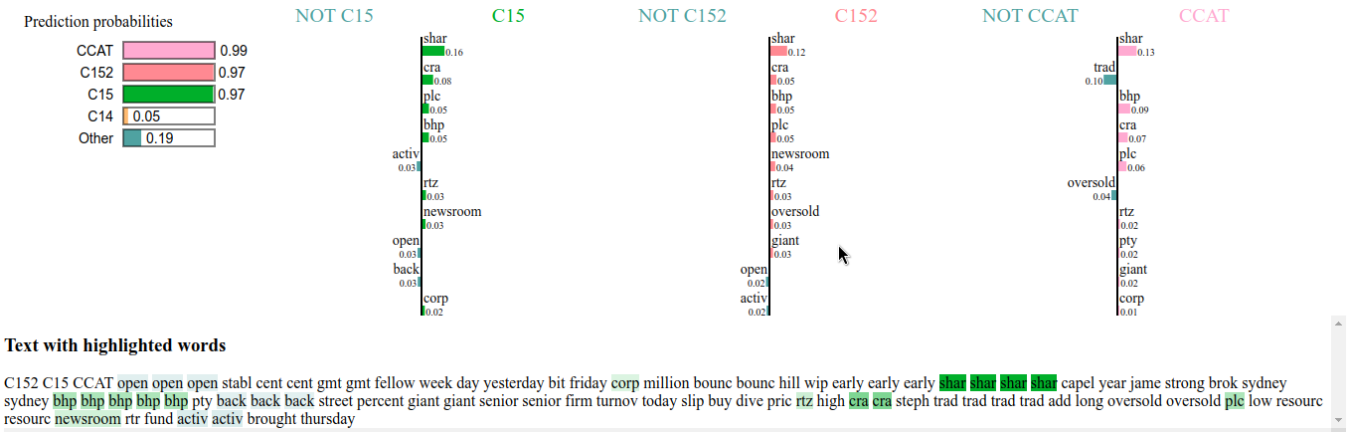

Figure 4: LIME [7] Visualization for Example #0 after adversarial training on RCV1 [3] dataset.

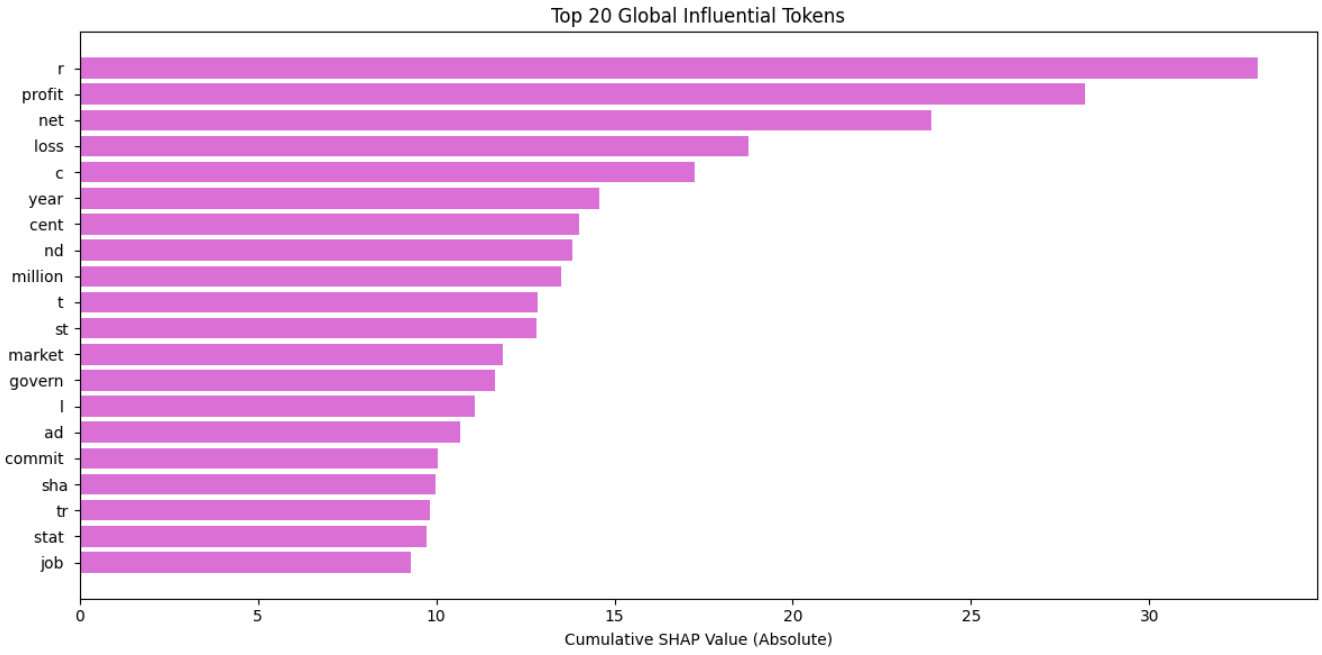

Figure 5: Global shap explanation Visualization for Example #0 after adversarial training on RCV1 [3] dataset.

A Contributions

A.1 Vivek Sapkal (*B22AI066*)

I led the data preprocessing pipeline for both RCV1 [3] and EUR-Lex datasets, including the challenging conversion of bag-of-words representations to sequential formats compatible with the XML-CNN architecture. This involved creating custom embedding mappings for numerical features and developing specialized tokenization strategies. Implemented the adversarial training methodology with adaptive epsilon scaling based on embedding norms, incorporating FGSM attack generation and defense integration into the training loop. Developed the feature squeezing quantization technique operating at 8-bit precision to enhance inference-time defenses. Additionally, conducted comprehensive performance evaluations across multiple metrics (P@1, P@3, P@5) under both clean and adversarial conditions, systematically analyzing robustness gaps and documenting the trade-offs between model performance and security. Created logging functionaliy to track training progress and compare model variants across different attack strengths.

A.2 Preet Savalia (*B22AI036*)

To enhance the interpretability of the model, I implemented both the LIME [7] and SHAP [5] explainability methods. For LIME [7], I generated local explanations for sample predictions by perturbing input texts and identifying key influential tokens per class. Each explanation was saved in HTML format for visualization and qualitative analysis. For SHAP [5], I utilized a BERT-based tokenizer with a custom model wrapper to compute SHAP [5] values. I generated per-example explanations highlighting token contributions and also performed a global analysis by aggregating SHAP [5] values across multiple samples to identify the top influential words overall. This enabled both local-level debugging and dataset-level insight. All results, including HTML visualizations and bar plots for global token importance, were saved for review. These explainability tools helped validate the behavior of the model and ensured greater transparency in the predictions.