XLNet-CNN: Combining Global Context Understanding of XLNet with Local Context Capture through Convolution for Improved Multi-Label Text Classification

Asif Shahriar
Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
asif.shahriar@bracu.ac.bd

Debojit Pandit
Computer Science and Engineering
BUET
Dhaka, Bangladesh, Bangladesh
debojitpanditdip@gmail.com

M Saifur Rahman
Computer Science and Engineering
BUET
Dhaka, Bangladesh, Bangladesh
mrahman@cse.buet.ac.bd

Abstract

Multi-label text classification (MLTC) is the task of assigning multiple relevant labels to a text, which is particularly challenging due to the complex interdependencies between labels and the imbalanced distribution of label frequencies. Domain-specific BERT variants, such as BioBERT, CT-BERT, and HateBERT, are pretrained on specialized corpora, which enables them to capture the unique terminology and patterns within specific domains, thus enhancing their performance in MLTC. However, pretraining BERT variants on specialized corpora is computationally expensive and limits their generalizability to broader tasks, necessitating models that can leverage both domain-specific insights and general-purpose context efficiently. With this in view, we propose XLNet-CNN, which is based on XLNet, an autoregressive transformer model designed to capture long-range dependencies through permutation-based training. Our model enhances XLNet's global context understanding by integrating a 1D CNN layer to better capture local dependencies and patterns within the text. This combination allows the model to recognize important phrases and word combinations, which are crucial for multi-label text classification. Our experiments on three distinct datasets — Ohsumed (medical abstracts), CAVES (anti-COVID vaccine tweets), and HateXplain (cyberbullying detection)—demonstrate that XLNet-CNN consistently outperforms XLNet and domain-specific BERT models in terms of F1-score. The detailed code and preprocessed data can be found in our github repository: https://github.com/asif-shahriar11/XLNet-CNN.

Keywords

Text Classification, Multi-label, BERT, XLNet, XLNet-CNN, F1-score, NLP

ACM Reference Format:
Asif Shahriar, Debojit Pandit, and M Saifur Rahman. 2024. XLNet-CNN: Combining Global Context Understanding of XLNet with Local Context Capture through Convolution for Improved Multi-Label Text Classification. In *11th International Conference on Networking, Systems, and Security (NSysS '24), December 19–21, 2024, Khulna, Bangladesh.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3704522.3704540

NSysS '24, December 19–21, 2024, Khulna, Bangladesh
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1158-9/24/12
https://doi.org/10.1145/3704522.3704540

1 Introduction

A fundamental task in Natural Language Processing (NLP) is text-classification, which means assigning categories or labels to text documents or pieces of texts. It involves training a machine learning model on a dataset of text examples that have already been labeled. The model learns patterns from this data to automatically categorize new, unseen text. Text classification has a wide range of applications, including spam detection, intent detection, sentiment analysis, topic labelling, etc. Traditional text classification focuses on assigning a single label to a document. For example, a news article might be classified as 'sports', 'politics' or 'technology'. This is often called *Multi-class Classification*, where an input example (a text sample in our case) can belong to only one of multiple categories (classes). However, the real world often presents more complex scenarios where texts can inherently belong to multiple categories simultaneously. This leads us to multi-label text classification, an extension of the conventional approach, designed to handle such complexities.

Multi-label classification. In contrast to multi-class classification, multi-label classification allows an input example to be assigned to multiple categories. For example, if an article discusses the use of technologies like Video Assistant Referee (VAR) in a football match, then this aticle should be classified as both 'sports' and 'technology' simultaneously. The challenge of multi-label text classification (MLTC) lies in its complexity and the interdependencies between labels. It requires sophisticated models that can accurately predict a set of categories for each document, taking into account the correlations and the potential overlap between categories.

Text classification models. Over the years, a variety of classification models have been developed to tackle this problem, ranging from traditional machine learning approaches like decision trees [7], support vector machines (SVM) and k-nearest neighbors (KNN) [5], and ensemple learning [17, 21], to more advanced deep learning techniques that leverage neural networks. The transformer architecture introduced in 2017 by Vaswani et al. [16] shows impressive capabilities in capturing long-range dependencies and understanding the context of words in large texts; making it highly effective for classifying texts into categories based on their content. Transformer-based models, especially BERT, have set new benchmarks for a variety of NLP tasks, including text classification.

BERT. Bidirectional Encoder Representations from Transformers (BERT), introduced by Google in 2018 [3], revolutionized the field of NLP by its innovative use of the Transformers architecture. Unlike previous transformers-based models like GPT [14] that read the text input sequentially, BERT processes the entire sequence of words at once, allowing it to capture the context of a word based on all of its surroundings (left and right of the word). BERT is pre-trained on a large corpus of text from the internet using two unsupervised tasks: Masked Language Model (MLM), which masks random tokens and attempts to predict them based on the context provided by the other non-masked words in the sequence; and Next Sentence Prediction (NSP), which learns to predict if two given sentences logically follow each other in a document. For specific tasks like text classification, BERT can be fine-tuned by adding a simple output layer on top of the BERT model. During fine-tuning, all of the parameters of BERT along with the additional output layer are trained on the downstream task dataset.

Domain-specific Fine-tuned BERT models. These models are adaptations of the original BERT model that have been further pre-trained and fine-tuned on domain-specific corpora. This additional pre-training step allows the models to capture the unique language, terminology, and stylistic features of various specialized fields, leading to improved performance on NLP tasks within those domains. For example, **BioBERT** [8] is a version of BERT that has been further pre-trained on large-scale biomedical corpora,**CT-BERT** (Covid-Twitter BERT) [12] is a specialized adaptation of the BERT-large model pretrained specifically for analyzing COVID-19 related content on Twitter, and **HateBERT** [1] has been further pretrained on hate speech-related data, particularly from Reddit, and is designed to detect and analyze toxic language and hate speech more effectively.

XLNet. XLNet [19], introduced by researchers at Google Brain and Carnegie Mellon University in 2019, addresses certain limitations of previous models like BERT and incorporates several innovative ideas. Instead of masking words, XLNet considers all possible permutations (orders) of the words in a sentence, allowing it to learn to predict a word based on all the words in that sentence. This allows XLNet to better understand the nuances of language, leading to improved performance on tasks like natural language understanding, text classification, and question answering [19]. Furthermore, XLNet's ability to consider all permutations of the input sequence helps it generalize better to new or unseen data compared to BERT. This is particularly useful in tasks where the order of words significantly impacts the meaning of the text. The capabilities of XLNet make it well-suited for multi-label text classification tasks. Its ability to understand the nuanced context and dependencies within text can be particularly beneficial when dealing with texts that cover multiple topics or contain several aspects that need to be classified into different categories.

Our Approach. Transformer-based models like XLNet sometimes struggle with capturing local context efficiently due to their focus on global dependencies [4, 10]. Convolutional filters can learn to recognize key phrases or word combinations that are indicative of certain classifications, thus enhancing transformer-based models' ability to perform better on tasks like text classification that require capturing fine-grained local context [2, 4, 9, 10, 18, 20]. In this work, we attempt to improve XLNet's ability to correctly classify documents in a multi-label setting by passing the embeddings generated by XLNet through a 1D CNN layer, as CNN adds the ability to capture local dependencies and patterns, which can be particularly useful for recognizing phrases or smaller units of meaning within the text. We tested our model, XLNet-CNN's performance in MLTC in three multi-label datasets from three different domains - Ohsumed, which is a collection of medical journal abstracts, CAVES, which contains anti-covid vaccine tweets, and HateXplain, which contains social media comments related to cyberbullying.

Findings. In our experiment, we found that XLNet-CNN significantly improved on the capabilities of XLNet and consistently outperformed domain-specific models like BioBERT, CT-BERT, and HateBERT in terms of F1-score, which is the main MLTC metric [15], thus demonstrating its superior ability in this domain. It was primarily due to XLNet-CNN's higher recall across all datasets, suggesting that XLNet-CNN was able to identify a larger proportion of the true positive labels compared to XLNet and the domain-specific pretrained BERT variants. Although XLNet-CNN's precision was sometimes slightly lower, its excellent recall compensated for this, leading to a much better F1-score. In terms of AUC-ROC or AUPR, despite not always performing the best, XLNet-CNN remained highly competitive, further demonstrating its robustness. The combination of global context (from XLNet) and local feature extraction (from CNN) enabled it to handle diverse datasets effectively, making it a robust choice for multi-label classification tasks.

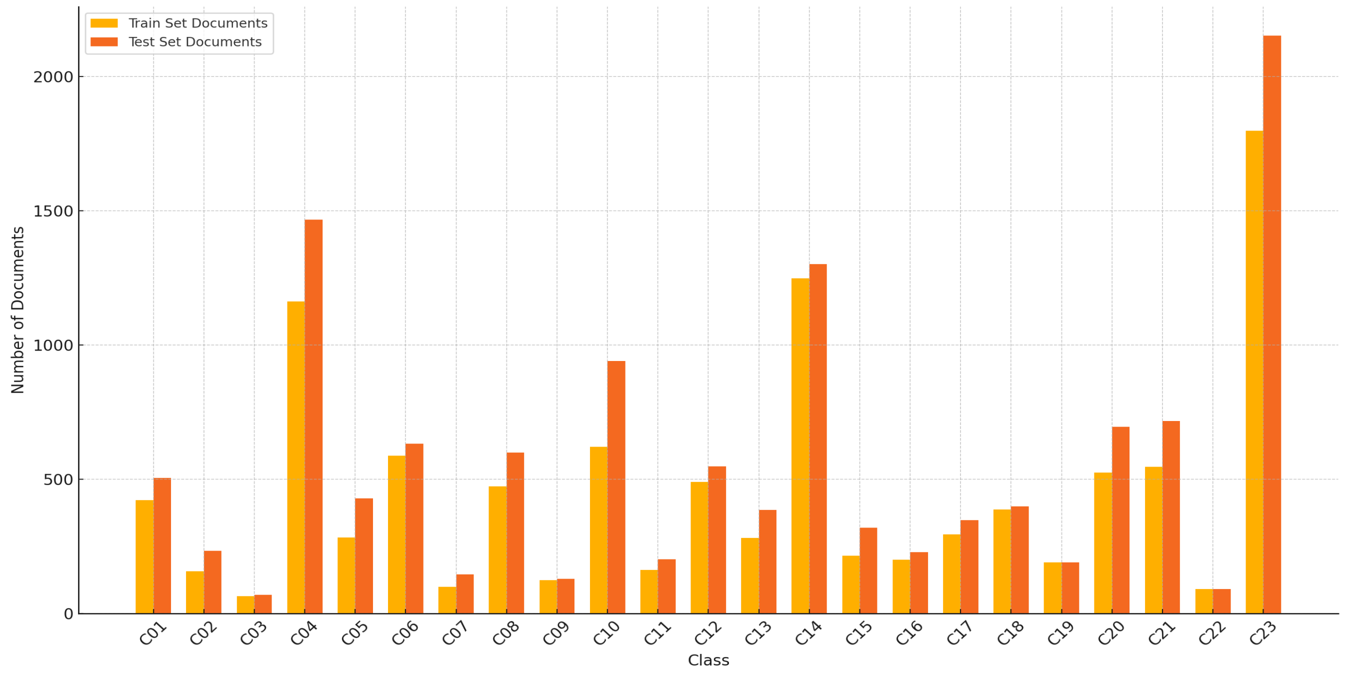
The rest of the paper is organized as follows. In Section 2 we discuss the three datasets used in our experiment. The detailed architecture of our model XLNet-CNN is presented in Section 3. In Section 4 we discuss the methodology for training the models and evaluating their performance, along with explaining why F1-score is the most important metric for MLTC. The results and their interpretation is presented in Section 5. In Section 6 we discuss the constraints encountered during the experiment and how the may affect the results, and present the final conclusions of this study.

2 Datasets

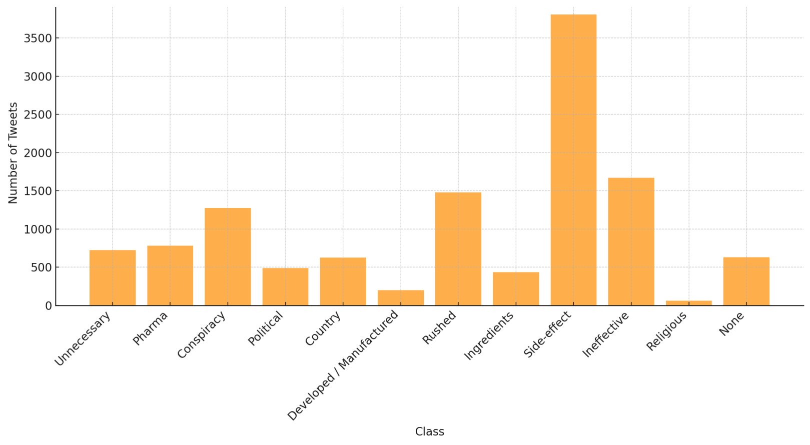
For this multi-label classification experiment we have selected three datasets from three different domains. In this section we briefly describe these datasets. The class distribution of these datasets can be found in Fig. 1.

2.1 Ohsumed Dataset

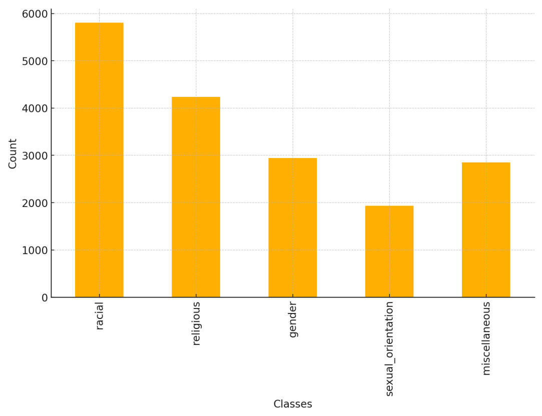
The Ohsumed dataset is a publicly available collection of abstracts of medical journal articles. It is part of the MEDLINE database, which is a large index of medical literature maintained by the National Library of Medicine (NLM). The Ohsumed dataset specifically refers to a subset of MEDLINE documents, selected for certain research purposes, particularly related to information retrieval and natural language processing in the biomedical domain. The documents in the Ohsumed dataset are manually annotated with MeSH *(Medical Subject Headings)* terms, which are a comprehensive controlled vocabulary for the purpose of indexing journal articles and books in the life sciences. There are 23 different annotations, which means there are 23 distinct classes of documents in the dataset. The classes are anotated as C01 through C23. The dataset contains a total of



(a) Ohsumed dataset



(b) CAVES dataset



(c) HateXplain dataset

Figure 1: Classification of all three datasets

13931 documents, which are split into train-test sets. The train set contains 6287 documents and the test set contains 7644 documents.

2.2 CAVES: Anti-covid Vaccine Tweets Dataset

The CAVES dataset [13] is a large-scale dataset containing about 10,000 COVID-19 anti-vaccine tweets, which are manually labeled with 11 established anti-vaccine concerns, such as concerns about the vaccine ingredients, side-effects of vaccines, concerns about monetary motivations of the pharmaceutical companies, doubts about the countries making particular vaccines, and so on (see Fig. 1b for a distribution of these tweets). Since a particular tweet often reflects multiple anti-vaccine concerns, the dataset is multi-labeled. Additionally, the dataset is curated with human-annotated explanations for the labels, indicating which exact parts of the tweet-text indicates a particular anti-vaccine concern.

2.3 HateXplain: Cyberbullying Dataset

This dataset [11] contains annotated social media comments specifically curated for the analysis and detection of cyberbullying. The data is labeled across multiple categories including race, religion, gender, sexual orientation, and miscellaneous attributes. A single comment can have multiple labels assigned to it, making it suitable for multi-label classification task. Original hateXplain dataset was in JSON format, we have used a processed and labelled CSV version. This version contains comments and their corresponding labels indicating whether the comment is normal, offensive, or hatespeech. It also includes additional columns that indicate what particular race, religion, gender, sexual orientation, or other miscellaneous attribute a particular comment is related to. We have done some preprocessing for using the dataset in a multi-label setting. We filtered the dataset to retain only the comments labeled as either offensive or hatespeech, and converted the categorical columns (Race, Religion, Gender, Sexual Orientation, and Miscellaneous) into binary values (see Fig. 1c for a distribution of the hate comments).

3 XLNet-CNN Model Description

In this section, we discuss the motivation for integrating the XLNet Transformer with a 1D CNN and explain the structure of the proposed XLNet-CNN model in detail, including the key mathematical operations involved. Fig. 2 shows the architecture of XLNet-CNN.

3.1 Motivation

The XLNet-CNN model is designed to combine the strengths of both transformer-based and convolutional neural network (CNN) architectures to enhance multi-label text classification. XLNet excels at capturing long-range dependencies and global context through its self-attention mechanism. To further improve the model's ability to capture important local features, such as n-grams or short phrases, convolutional layers are applied. CNNs are particularly well-suited for recognizing these local patterns due to their ability to focus on small, fixed-size regions of the input. By combining XLNet's global context capture with CNN's ability to extract local dependencies, the model aims to perform well in text classification tasks where both long and short-range dependencies are important.

3.2 Model Architecture

Input Preparation. The input data for the XLNet-CNN model is initially loaded from CSV files containing text sequence and corresponding labels. Before feeding data into the XLNet-CNN model, raw text sequences must be tokenized and transformed into a format that XLNet can process. Using a tokenizer, each text sequence is converted into `input_ids`, which represent the tokenized words or subwords, an `attention_mask` to highlight valid tokens, and `token_type_ids` for distinguishing different segments (if necessary). These inputs are then passed to a `DataLoader`, which manages batching and shuffling during training and evaluation. This step ensures the data is ready for efficient processing by the model.

XLNet Layer. The tokenized input is fed into the XLNet model, which produces contextualized embeddings for each token in the sequence. Mathematically, we can represent the input sequence as:

$$X = [x_1, x_2, \ldots, x_T]$$

where T is the sequence length, and each x_i corresponds to a token from the text. XLNet processes this sequence to generate hidden states:

$$\mathbf{H}_{\mathbf{XLNet}} = [h_1, h_2, \ldots, h_T]$$

where each $h_i \in \mathbb{R}^{768}$ represents the contextual embedding for token x_i. These embeddings are rich representations of the token's

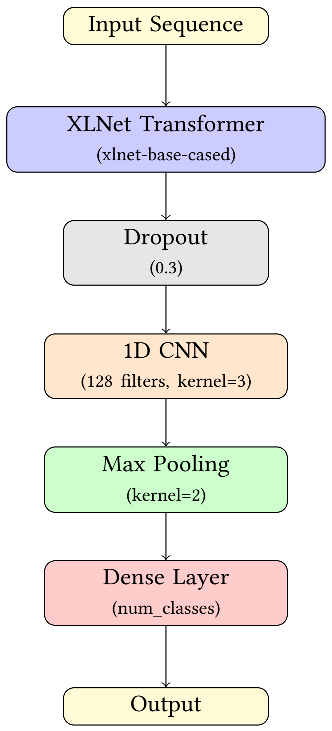


Figure 2: XLNet-CNN model architecture

meaning in the context of the entire sequence. The output tensor from XLNet is of shape $\mathbf{H}_{\mathbf{XLNet}} \in \mathbb{R}^{B\times T\times 768}$, where B is the batch size, T is the sequence length, and 768 is the embedding size.

Dropout Layer. To regularize the model and prevent overfitting, *dropout* is applied to the output of the XLNet model. The dropout layer randomly sets a fraction $p = 0.3$ of the elements of the tensor $\mathbf{H}_{\mathbf{XLNet}}$ to zero:

$$\mathbf{H}_{\mathbf{drop}} = \mathrm{Dropout}(\mathbf{H}_{\mathbf{XLNet}})$$

This tensor, $\mathbf{H}_{\mathbf{drop}} \in \mathbb{R}^{B\times T\times 768}$, is passed on to the CNN layer for further processing.

CNN Layer. The output of XLNet is transformed into a shape compatible with 1D convolution by permuting the tensor dimensions:

$$\mathbf{H}_{\mathbf{perm}} = \mathrm{Permute}_{(0,2,1)}(\mathbf{H}_{\mathbf{drop}}) \in \mathbb{R}^{B\times 768\times T}$$

Here, the batch size remains in the first position (indexed by 0), the hidden size is moved to the second position (indexed by 2), and the sequence length is moved to the third position (indexed by 1), thus aligning the tensor with the required input format for the 1D convolution layer. Afterwards, 1D convolution is applied with 128 filters, each of size 3, to learn local patterns in the sequence of embeddings:

$$\mathbf{C} = \mathrm{ReLU}(\mathrm{Conv1d}(\mathbf{H}_{\mathbf{perm}}))$$

where $\mathbf{C} \in \mathbb{R}^{B\times 128\times T}$. The ReLU activation function introduces non-linearity into the model.

Next, max pooling is applied to down-sample the output, reducing the sequence length by half:

$$\mathbf{P} = \text{MaxPool}(\mathbf{C})$$

where $\mathbf{P} \in \mathbb{R}^{B \times 128 \times \frac{T}{2}}$. This reduces the computational cost and highlights the most significant features from the convolutional filters.

Fully Connected Layer. The pooled output is then flattened:

$$\mathbf{P}_{\mathbf{flat}} = \text{Flatten}(\mathbf{P})$$

where $\mathbf{P}_{\mathbf{flat}} \in \mathbb{R}^{B \times (128 \times \frac{T}{2})}$The flattened vector is passed to the fully connected layer, which projects it into the number of classes (for multi-label classification):

$$\mathbf{O} = \mathbf{P}_{\mathbf{flat}} W_{fc} + b_{fc}$$

where $W_{fc} \in \mathbb{R}^{(128 \times \frac{T}{2}) \times \text{num\_classes}}$ is the weight matrix, $b_{fc} \in \mathbb{R}^{\text{num\_classes}}$ is the bias vector, and $\mathbf{O} \in \mathbb{R}^{B \times \text{num\_classes}}$ is the output.

For multi-label classification, a sigmoid activation is applied to the final output:

$$\hat{y}_i = \text{Sigmoid}(O_i)$$

where $\hat{y}_i$ represents the predicted probability for each label i independently.

This model architecture integrates XLNet to capture long-range dependencies, CNN for local feature extraction, and a fully connected layer for classification. The combination of these components allows for more robust performance in multi-label text classification tasks.

4 Methodology

In this section we discuss about the appropriate performance metrics for multi-label classification tasks, hyperparameters used in our experiment, the detailed training procedure, model evaluation, and testing.

4.1 Metric for Multi-label Classification

The most common performance metric used in multi-class classification is accuracy, measured as the number of correct predictions divided by the number of samples. This is not appropriate in a multi-label setting, mainly for the following two reasons:

- In multi-label classification, each sample can be associated with multiple labels simultaneously. For instance, if a sample has five labels and a model predicts three of them correctly, this accuracy will consider those three correct, resulting in a deceptively high accuracy. A multi-label prediction is considered fully correct only if all the predicted labels match exactly with the true labels for a given sample, known as **subset accuracy**. However, subset accuracy is often too strict, disregarding partially correct predictions.
- Multi-label datasets often suffer from label imbalance, where some labels appear much more frequently than others. A model could achieve high accuracy by predominantly predicting the frequent labels while ignoring the rare ones, leading to misleading evaluation of the model's performance across all labels.

For multi-label classification, the **F1-score** is the most suitable performance metric. The F1-score is the harmonic mean of precision and recall, providing a single metric that balances both. Precision measures the accuracy of the positive predictions, while recall measures the ability of the model to find all the relevant cases within the dataset.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$
$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$
$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

This balance is crucial in multi-label settings where it's important not just to predict labels correctly, but also to ensure that all relevant labels for a sample are predicted. Moreover, F1-score, by accounting for both precision and recall, penalizes a model that simply predicts the most frequent label more than metrics like accuracy would, encouraging the model to correctly predict less frequent labels as well. Macro-average F1 computes the F1 score independently for each label and then takes the average, treating all labels equally. On the other hand, weighted-average F1 calculates F1 per label, weighted by the number of true instances, balancing label imbalance. The weighted F1-score is given by:

$$\text{Weighted F1} = \frac{1}{N} \sum_{i=1}^{N} w_i \times F1_i$$

Where w_i is the weight of each label based on its frequency, and $F1_i$ is the F1-score for label i. Since all three datasets in this experiment are imbalanced, we have used weighted F1-score.

AUC-ROC (Area Under the Receiver Operating Characteristic Curve) is another important metric, which measures the model's ability to distinguish between classes by plotting the true positive rate (recall) against the false positive rate. The AUC-ROC score ranges from 0 to 1, with 1 indicating perfect classification. The AUC-ROC score in multi-label classification is computed by averaging the AUC for each label:

$$\text{AUC-ROC} = \frac{1}{N} \sum_{i=1}^{N} \text{AUC}_i$$

Where N is the total number of labels and AUC_i is the AUC for label i. Additionally, the **AUPC** (Area Under the Precision-Recall Curve) focuses on the trade-off between precision and recall, making AUPC more relevant when dealing with rare labels. AUPC is more sensitive to how well the model performs on these less frequent labels and is crucial for multi-label classification:

$$\text{AUPC} = \frac{1}{N} \sum_{i=1}^{N} \text{AUPC}_i$$

Where AUPC_i is the area under the precision-recall curve for label i. Between AUC-ROC and AUPC, **AUPC** is often more important for multi-label classification, especially in imbalanced datasets, because it emphasizes precision and recall, which are critical for predicting rare labels accurately.

Table 1: Hyperparameters

Hyperparameter	Value
Sequence Length	512 tokens
Train Batch Size	8
Validation Batch Size	8
Test Batch Size	8
Epochs	15
Learning Rate	1e-5
Threshold for Sigmoid Output	0.5

4.2 Model Training

The training procedure follows the typical supervised learning process, but many decisions were made due to resource constraints. Specifically, the training was conducted using a Kaggle P100 GPU with 16GB of memory, which imposed certain limitations on batch size and other optimizations. Consequently, the batch size was set to 8 to fit within the available memory, and several other optimizations were applied to manage the training process effectively. Hyperparameters used in this experiment are mentioned in Table 1.

- **Gradient Accumulation**: The model uses a gradient accumulation strategy, where gradients are computed and accumulated over four steps before updating the model weights, thus effectively simulating a larger batch size without exhausting GPU memory.
- **Mixed Precision**: To reduce memory usage and speed up training time, mixed precision is used by combining 16-bit and 32-bit floating-point operations using `GradScaler`. This allows larger models to fit into the GPU memory while leveraging faster 16-bit operations for most calculations and maintaining accuracy with selective 32-bit precision where necessary.
- **Optimizer**: The model parameters are optimized using *Adam optimizer* [6]. The optimizer is updated every accumulation step, with gradient clipping applied to prevent exploding gradients.
- **Prediction**: During training, the model's outputs are passed through a sigmoid function, and a threshold of 0.5 is applied to round the predictions.
- **Loss Function**: The binary cross-entropy loss function is used, as it is suitable for multi-label classification. The loss is computed and backpropagated using accumulated gradients every few batches.

4.3 Model Evaluation

Model evaluation was performed on the validation set at the end of each training epoch to monitor the model's generalization capability. The model was set to evaluation mode, disabling dropout and batch normalization layers to ensure consistency in predictions. Evaluation metrics included subset accuracy, F1-score, precision, recall, AUC-ROC, and AUPC, providing a comprehensive view of model performance across multiple dimensions. The best model was selected based on the highest F1-score observed on the validation set.

4.4 Testing

Once the model is trained and validated, it is evaluated on the test set. The model with the best validation performance (as determined by the highest F1-score) is loaded for testing. The same evaluation metrics used during validation — accuracy, F1-score, precision, recall, AUC-ROC, and AUPR — are computed for the test set. This provides a final assessment of the model's generalization capabilities on unseen data.

5 Results & Discussion

In this section we compare our proposed model XLNet-CNN's performance against XLNet, BERT, and domain-specific pretrained BERT models - BioBERT [8] for Ohsumed, CT-BERT [12] for CAVES dataset, and hateBERT [1] for hateXplain dataset.

5.1 Comparative Performance

The multi-label classification results of our proposed model XLNet-CNN, along with XLNet, and domain-specific pre-trained BERT models BioBERT, CT-BERT, and HateBERT, for the three datasets discussed in Section 2, are presented in Tables 2, 3, and 4, respectively. Although F1-score combines both precision and recall, we present all three metrics to understand why a model may show better F1-score than another. Additionally, we also present subset accuracy, AUC-ROC, and AUPR. The best F1-scores and AUPRs are bold-faced, while the best F1-score is also underlined since it is the key metric. It should be mentioned that in the Ohsumed dataset, BERT consistently achieved a subset accuracy and F1-score of 0.0 throughout 15 epochs of training, indicating that it failed to make a single correct prediction. Consequently, there was no best model to save, and so we were unable to proceed with evaluating BERT on the test set, as can be seen in Table 2.

From the results we got in our experiment, it is clear that XLNet-CNN outperforms both XLNet and pre-trained BERT variants across all datasets, in terms of F1-score, thus establishing its superiority in multi-label text-classification.

Table 2: Performance on the Ohsumed Test Dataset

Measure	BERT	BioBERT	XLNet	XLNet-CNN
Subset Accuracy	-	14.04	21.39	24.00
F1-score	-	18.73	33.52	<u>**61.41**</u>
Precision	-	49.60	65.08	58.26
Recall	-	17.49	28.59	66.65
AUC-ROC	-	84.57	85.85	89.21
AUPR	-	50.87	55.94	**64.62**

Table 3: Performance on the CAVES Test Dataset

Measure	BERT	CT-BERT	XLNet	XLNet-CNN
Subset Accuracy	26.45	52.60	43.80	41.17
F1-score	23.02	61.36	51.35	**62.47**
Precision	25.62	73.54	70.03	60.43
Recall	20.91	56.19	47.71	66.75
AUC-ROC	70.28	91.87	88.80	89.79
AUPR	38.64	**71.75**	63.53	67.97

Table 4: Performance on the HateXplain Test Dataset

Measure	BERT	HateBERT	XLNet	XLNet-CNN
Subset Accuracy	35.42	52.83	45.59	50.92
F1-score	46.84	73.77	68.25	**74.91**
Precision	73.41	82.38	77.94	77.15
Recall	43.05	68.85	62.92	73.07
AUC-ROC	79.76	88.68	86.15	88.31
AUPR	67.20	**80.93**	77.54	80.42

5.2 Analysis

Based on our evaluation, we have the following observations.

F1-score. As we discussed in section 4.1, for multi-label classification, F1-score is the most crucial metric, as it balances precision and recall. XLNet-CNN consistently achieved the highest F1-scores across all datasets, significantly outperforming both the domain-specific BERT variants and XLNet.

- **Ohsumed**: XLNet-CNN achieved an F1-score of **61.41**, a substantial improvement over BioBERT (**18.73**) and XLNet (**33.52**).
- **CAVES**: The F1-score of XLNet-CNN was **62.47**, surpassing both CT-BERT (**61.36**) and XLNet (**51.35**).
- **HateXplain**: XLNet-CNN lead with an F1-score of **74.91**, outperforming both HateBERT (**73.77**) and XLNet (**68.25**).

The XLNet-CNN model's superiority can be attributed to its ability to combine XLNet's global context understanding with the CNN layer's capacity to capture local patterns like n-grams or short phrases. This combination allows XLNet-CNN to better identify interrelated labels in multi-label classification tasks, which leads to a significant performance boost, particularly in imbalanced datasets.

AUPR. Area Under the Precision-Recall Curve (AUPR) evaluates the trade-off between precision and recall. In our experiemnt, XLNet-CNN performed competitively in terms of AUPR but did not consistently outperform the domain-specific models.

- **Ohsumed**: XLNet-CNN achieved an AUPR of **64.62**, higher than both BioBERT (**50.87**) and XLNet (**55.94**).
- **CAVES**: Although XLNet-CNN (**67.97**) was able to improve on base XLNet (**63.53**) significantly, it fell just short of CT-BERT, which lead with an AUPR of **71.75**.
- **HateXplain**: Here XLNet-CNN's performance (**80.42**) was very comparable with domain-specific pre-trained model HateBERT (**80.93**), while XLNet was at **77.54**.

Though XLNet-CNN did not always have the highest AUPR, it still maintained a solid balance between precision and recall, highlighting the model's robustness in handling rare labels effectively, though domain-specific models like CT-BERT and HateBERT occasionally have an edge.

Precision, recall, AUC-ROC, subset accuracy.

Ohsumed Dataset.

- **Precision and Recall**: XLNet-CNN achieved a recall of **66.65**, which was significantly higher than both BioBERT (**17.49**) and XLNet (**28.59**), showing its superior ability to identify relevant labels. However, XLNet-CNN's precision (**58.26**) was lower than XLNet's (**65.08**), meaning it occasionally predicted more false positives. The combination of a high recall with moderate precision resulted in the highest F1-score for XLNet-CNN.
- **AUC-ROC**: XLNet-CNN achieved an AUC-ROC of **89.21**, higher than XLNet (**85.85**) and BioBERT (**84.57**). This indicated that XLNet-CNN performed well at distinguishing between relevant and irrelevant labels for each sample.
- **Subset Accuracy**: XLNet-CNN achieved a subset accuracy of **24**, the highest among all models. However, as we discussed earlier, subset accuracy is often too strict for multi-label classification as it required all predicted labels to be correct for a sample, and so it does not capture the model's overall performance.

CAVES Dataset.

- **Precision and Recall**: XLNet-CNN achieved a recall of **66.75**, quite a bit higher than both CT-BERT (**56.19**) and XLNet (**47.71**), demonstrating its ability to correctly identify more relevant labels. Its precision, at **60.43**, was lower than CT-BERT's (**73.54**) and XLNet's (**70.03**), indicating that it predicted more false positives. The higher recall compensated for this, resulting in the highest F1-score.
- **AUC-ROC**: XLNet-CNN achieved an AUC-ROC of **89.79**, higher than XLNet (**87.80**) and comparable to CT-BERT (**88.67**), showing competitive performance in distinguishing between relevant and irrelevant labels.
- **Subset Accuracy**: CT-BERT achieved the highest subset accuracy (**52.60**), with XLNet-CNN coming in at **41.17**.

HateXplain Dataset.

- **Precision and Recall**: XLNet-CNN performed well with a recall of **73.07**, significantly higher than HateBERT (**68.85**) and XLNet (**62.92**). Its precision (**77.15**) was slightly lower than HateBERT (**82.38**), indicating that while it made more false-positive predictions, it did a better job of identifying all relevant labels, resulting in the best F1-score (**74.91**).

- **AUC-ROC**: XLNet-CNN achieved an AUC-ROC of **88.31**, quite comparable to HateBERT (**88.68**) and better than XLNet (**80.82**), showing its robustness in distinguishing between relevant and irrelevant labels across this dataset.
- **Subset Accuracy**: XLNet-CNN achieved a subset accuracy of **50.92**, close to HateBERT's **52.83**.

Overall, XLNet-CNN consistently outperformed domain-specific models like BioBERT, CT-BERT, and HateBERT, as well as the base XLNet model, in terms of F1-score. The primary factor contributing to this superior performance was XLNet-CNN's consistently higher recall across all datasets. By capturing more relevant labels with fewer omissions, XLNet-CNN was able to identify a larger proportion of the true positive labels compared to XLNet and the domain-specific pretrained BERT variants. While XLNet-CNN did not always achieve the highest AUC-ROC or AUPR, it remained highly competitive in these metrics, further demonstrating its robustness.

6 Discussion & Conclusion

To accommodate large models such as XLNet, CT-BERT, HateBERT, and BioBERT within the available memory, we employed techniques like gradient accumulation and mixed precision. While these methods may introduce some performance trade-offs due to memory constraints, they were necessary given the limited computational resources. It is likely that, without such resource limitations and the need for gradient accumulation or mixed precision, model performance would differ. Nevertheless, all models were trained under the same conditions and procedures in this experiment, ensuring a fair comparison of the results presented in this paper.

In this paper, we proposed XLNet-CNN, a model combining the global context understanding of XLNet with the local feature extraction capabilities of CNN, for multi-label text classification. Our extensive experiments on three datasets (Ohsumed, CAVES, and HateXplain) demonstrated that XLNet-CNN consistently outperformed both the base XLNet model and domain-specific pretrained BERT variants (BioBERT, CT-BERT, and HateBERT), particularly in terms of the most crucial metric for multi-label classification, the F1-score. The key to XLNet-CNN's success was its significantly higher recall compared to the other models, which allowed it to capture more relevant labels, leading to its superb F1-score. This demonstrates that the integration of CNN helps XLNet in handling diverse datasets with multiple labels by capturing both global and local patterns effectively. However, due to training resource limitations, particularly using a Kaggle P100 GPU with 16GB of memory, we were constrained to a batch size of 8, which is why we could not test the models (both XLNet-CNN and pretrained BERT variants) to the full potential. For future work, we plan to explore improvements in the model architecture to further boost performance. Additionally, we aim to test all models without resource constraints, using larger batch sizes, and assess whether further gains in performance can be achieved through hyper-parameter tuning.

References

[1] Tommaso Caselli, Valerio Basile, Jelena Mitrović, and Michael" Granitzer. 2021. HateBERT: Retraining BERT for Abusive Language Detection in English. In *Proceedings of the 5th Workshop on Online Abuse and Harms (WOAH 2021)*. Association for Computational Linguistics, Online, 17–25. https://doi.org/10.18653/v1/2021.woah-1.3

[2] Xinying Chen, Peimin Cong, and Shuo Lv. 2022. A Long-Text Classification Method of Chinese News Based on BERT and CNN. *IEEE Access* 10 (2022), 34046–34057. https://doi.org/10.1109/ACCESS.2022.3162614

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Vol. 1. 4171–4186. https://arxiv.org/pdf/1810.04805.pdf

[4] Maosheng Guo, Yu Zhang, and Ting Liu. 2019. Gaussian Transformer: A Lightweight Approach for Natural Language Inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 6489–6496. https://doi.org/10.1609/aaai.v33i01.33016489

[5] Zahra Hanifelou, Peyman Adibi, Sayyed Amirhassan Monadjemi, and Hossein Karshenas. 2018. KNN-based multi-label twin support vector machine with priority of labels. *Neurocomputing* 322 (2018), 177–186. https://doi.org/10.1016/j.neucom.2018.09.044

[6] Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR)* (2015). https://arxiv.org/abs/1412.6980

[7] Anwesha Law and Ashish Ghosh. 2022. Multi-Label Classification Using Binary Tree of Classifiers. *IEEE Transactions on Emerging Topics in Computational Intelligence* 6, 3 (2022), 677–689. https://doi.org/10.1109/TETCI.2021.3075717

[8] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* 36, 4 (2020), 1234–1240. https://doi.org/10.1093/bioinformatics/btz682

[9] Junzhe Li, Chenglong Wang, Xiaohan Fang, Kai Yu, Jinye Zhao, Xi Wu, and Jibing Gong. 2022. Multi-label text classification via hierarchical Transformer-CNN. In *Proceedings of the 2022 14th International Conference on Machine Learning and Computing (ICMLC '22)*. Association for Computing Machinery, 120–125. https://doi.org/10.1145/3529836.3529912

[10] Pengfei Li, Peixiang Zhong, Kezhi Mao, Dongzhe Wang, Xuefeng Yang, Yunfeng Liu, Jianxiong Yin, and Simon See. 2021. ACT: an Attentive Convolutional Transformer for Efficient Text Classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 13261–13269. https://doi.org/10.1609/aaai.v35i15.17566

[11] Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2021. HateXplain: A Benchmark Dataset for Explainable Hate Speech Detection. In *AAAI conference on artificial intelligence*, Vol. 35. 14867–14875.

[12] Masashi Nakamura, Satoshi Kawazoe, and Kenji Kurokawa. 2020. COVID-Twitter-BERT: A Natural Language Processing Model to Analyse COVID-19 Content on Twitter. *arXiv preprint arXiv:2005.07503* (2020). https://arxiv.org/abs/2005.07503

[13] Soham Poddar, Siwei Luo, and Long Xing. 2022. CAVES: A Dataset to facilitate Explainable Classification and Summarization of Concerns towards COVID Vaccines. (2022). https://arxiv.org/abs/2204.13746

[14] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-Training. (2018). https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf

[15] Grigorios Tsoumakas and Ioannis Katakis. 2007. Multilabel classification: An overview. *International Journal of Data Warehousing and Mining* 3, 3 (2007), 1–13.

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, Vol. 30. Curran Associates, Inc., 5998–6008. https://arxiv.org/abs/1706.03762

[17] Qingyao Wu, Mingkui Tan, Hengjie Song, Jian Chen, and Michael K. Ng. 2016. ML-FOREST: A Multi-Label Tree Ensemble Method for Multi-Label Classification. *IEEE Transactions on Knowledge and Data Engineering* 28, 10 (2016), 2665–2680. https://doi.org/10.1109/TKDE.2016.2581161

[18] Baosong Yang, Longyue Wang, Derek Wong, Lidia S. Chao, and Zhaopeng Tu. 2019. Convolutional Self-Attention Networks. arXiv:1904.03107 [cs.CL] https://arxiv.org/abs/1904.03107

[19] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *Advances in Neural Information Processing Systems*. 5754–5764. https://arxiv.org/abs/1906.08237

[20] Shaomin Zheng and Meng Yang. 2019. A New Method of Improving BERT for Text Classification. In *Intelligence Science and Big Data Engineering. Big Data and Machine Learning*. Springer International Publishing, 442–452.

[21] Xiaoyan Zhu, Jiaxuan Li, Jingtao Ren, Jiayin Wang, and Guangtao Wang. 2023. Dynamic ensemble learning for multi-label classification. *Information Sciences* 623 (2023), 94–111. https://doi.org/10.1016/j.ins.2022.12.022