Don't Just Pay Attention, PLANT It: Transfer L2R Models to Fine-tune Attention in Extreme Multi-Label Text Classification

Anonymous ACL submission

Abstract

State-of-the-art Extreme Multi-Label Text Classification (XMTC) models rely heavily on multi-label attention layers to focus on key tokens in input text, but obtaining optimal attention weights is challenging and resource-intensive. To address this, we introduce PLANT — **P**retrained and **L**everaged **A**tte**NT**ion — a novel transfer learning strategy for fine-tuning XMTC decoders. PLANT surpasses existing SOTA methods across all metrics on `MIMIC-III-full`, `MIMIC-III-top50`, `MIMIC-IV-full`, `EURLEX-4K` and `WIKI10-31K` datasets. It particularly excels in few-shot scenarios, outperforming previous models specifically designed for few-shot scenarios by over 50 percentage points in F1 scores on `MIMIC-III-rare50` and by over 36 percentage points on `MIMIC-III-few`, demonstrating its superior capability in handling rare codes. PLANT also shows remarkable data efficiency in few-shot scenarios, achieving precision comparable to traditional models with significantly less data. These results are achieved through key technical innovations: leveraging a pretrained Learning-to-Rank model as the planted attention layer, integrating mutual-information gain to enhance attention, introducing an inattention mechanism, and implementing a stateful-decoder to maintain context. Comprehensive ablation validate the importance of these contributions in realizing the performance gains.

1 Introduction

Extreme Multi-Label Text Classification (XMTC) addresses the problem of automatically assigning each data point with most relevant subset of labels from an extremely large label set. One major application of XMTC is in the global healthcare system, specifically in the context of the International Classification of Diseases (ICD) (WHO, 2019). ICD coding is the process of assigning codes representing diagnoses and procedures performed during a patient visit using clinical notes documented by health professionals (Table 1). ICD codes are used for both epidemiological studies and billing of services. XMTC has been utilized to automate the manual ICD coding performed by clinical coders which is time intensive and prone to human errors (Nguyen et al., 2018).

428.0 : *Congestive heart failure*
··· DIAGNOSES: 1. **Acute congestive heart failure** 2. Diabetes mellitus 3. Pulmonary edema ···
202.8 : *Other malignant lymphomas*
··· a 55 year-old female with **non Hodgkin's lymphoma** and acquired C1 esterase inhibitor deficiency ···

Table 1: Examples of clinical text fragments and their corresponding ICD codes (Li and Yu, 2020).

Main Challenge: Building XMTC models is challenging because datasets often consist of texts with multiple lengthy narratives – more than 1500 tokens (i.e., words) on average. However, only a small fraction of tokens are most informative with regard to assigning relevant labels. Automatically assigning labels become even more challenging when, (1) the label space is extremely high dimensional, and, (2) the label distribution is heavily skewed. For example, in automatic ICD coding, there are over 18000 and 170000 codes in ICD-9-CM and ICD-10-CM/PCS (CDC, 2024), respectively. The skewness of ICD-9-CM label distribution in the MIMIC-III dataset (Johnson et al., 2016) is evident from the fact that approximately 5411 out of all the 8929 codes appear less than 10 times.

How SOTA models address the main challenge in XMTC? (Figure 1a: dotted orange box) In XMTC, attention mechanisms play a vital role in addressing the challenges of high-dimensional label spaces and skewed label distributions. XMTC models (Lu et al., 2023; Li et al., 2023; Nguyen et al., 2023; Yang et al., 2023; Chen et al., 2023; Zhang and Wang, 2024; Luo et al., 2024) consistently feature a multi-label attention layer, dynam-

ically allocating label-specific attention weights to the most informative tokens in input text. Refer to the dotted orange components in Figure 1a, which illustrate the functionality of this critical attention layer. Regardless of the specific encoder architecture, removing this attention layer leads to a significant drop in performance.

Main Shortfall in Figure 1a Dotted Orange Box: Current SOTA XMTC models begin with random attention weights, requiring ranking all tokens for each label from scratch, which is data-intensive due to the high-dimensional label space. Skewed label distributions worsen this issue, as rare labels need even more data. Insufficient data leads to longer training times and a higher risk of overfitting (Figure 5). Studies like (Edin et al., 2023) show that SOTA models struggle to predict rare ICD diagnosis codes (Figure 2a). Models perform similarly across codes with comparable frequencies, indicating that the high proportion of rare codes significantly impacts performance. Correlations between code frequency and F1 score are moderately high, showing that rare codes are predicted less accurately than common ones. This underscores the need for efficient attention mechanisms, as starting with random weights isn't sufficient.

Main Contributions:

1. **ICD Coding:** We evaluated PLANT on the MIMIC-III and MIMIC-IV datasets, widely used in automatic ICD coding research. **PLANT outperformed 21 SOTA models across 7 evaluation metrics**, demonstrating significant performance improvements on the `MIMIC-III-full`, `MIMIC-III-top50`, `MIMIC-III-rare50`, and `MIMIC-IV-full` datasets (Tables 2, 10, 3, 4, 5).

2. **Data Efficiency:** PLANT excels in few-shot settings, handling high-dimensional skewed label distributions with significantly less data, **matching traditional attention models' precision with only $\frac{1}{10}$ of the data for precision at 5 and $\frac{1}{5}$ for precision at 15** (Figure 3).

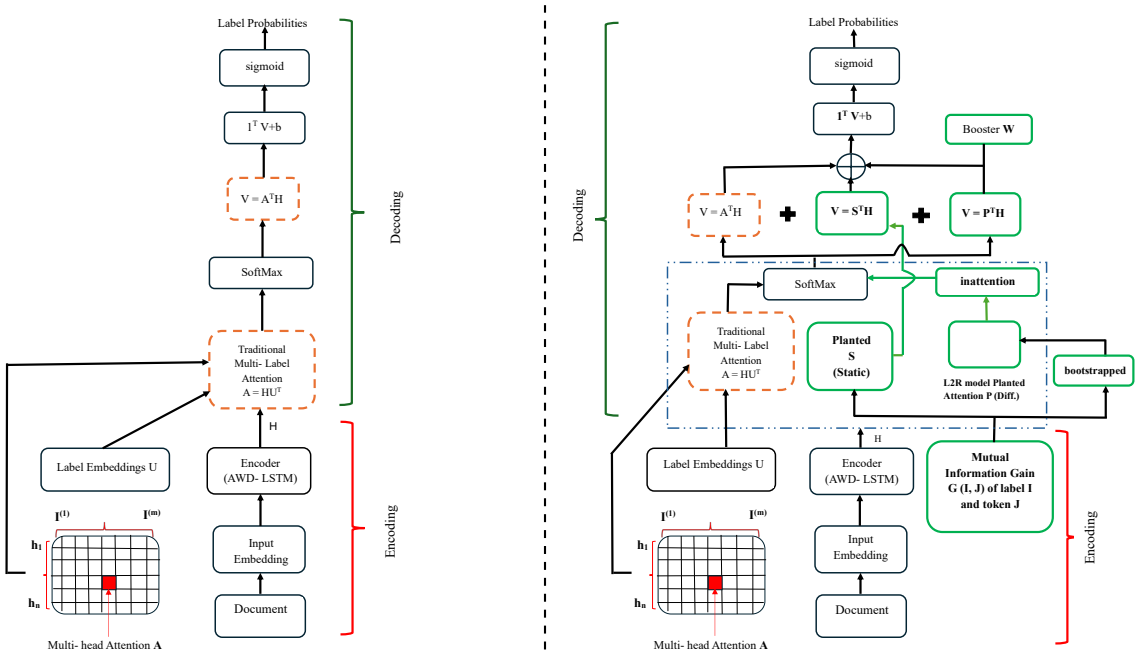
3. **Few-Shot:** PLANT shows exceptional performance on the `MIMIC-III-rare50` and `MIMIC-III-few` datasets, **outperforming previous few-shot SOTA models by over 50 percentage points in F1 scores on `MIMIC-III-rare50` and by over 36 percentage points on `MIMIC-III-few`**. It also achieves significant gains in precision and recall, establishing itself as the most effective solution for rare and few-shot ICD coding tasks (Figure 2b, Table 3 and 4). **Unlike Figure 2a (study performed by Edin et al. (2023)), which shows near-zero macro-F1scores for codes with frequencies less than 10, Figure 2b zooms in on these rare codes to reveal that PLANT significantly outperforms other models in predicting them** (see Section 4.2: Figure 2a Vs Figure 2b).

4. **XMTC:** PLANT achieves SOTA performance on publicly available XMTC datasets—`EURLEX-4K` and `WIKI10-31K`. Specifically, P@1 of 89.92% on `EURLEX-4K` and a P@1 of 90.10% on `WIKI10-31K`—**outperforming previous SOTA by 1.51 and 0.56% percentage points**, respectively (Table 6). We have made our trained models and code available at `https://anonymous.4open.science/r/xxx-111/`.

Technical Contributions (Green Box Figure 1b): Extensive ablation analysis in Table 7 of Section 5 demonstrates the impact of each of the following technical contribution in achieving the main contributions.

1. **Learning-to-Rank (L2R) Model**: Introduced PLANT, a novel transfer learning approach that uses a pretrained L2R model to fine-tune attention in XMTC. By leveraging L2R activations as attention weights, PLANT ensures the decoder starts with well-informed weights, leading to efficient convergence. Worth noting that "pretrained L2R model" does not involve any external model or additional data typically associated with pretrained models. Instead, it is a step in our main pipeline. We train the L2R model directly on the same datasets used to train the main pipeline for PLANT. In Section 5 (*Analysis*), we perform "*Ablation on Pretrained L2R in PLANT*" to specifically highlight its efficacy in both few-shot and full dataset scenarios.

2. **Mutual-Information Gain**: Bootstrapped the L2R model using mutual information gain to enhance attention. This ensures that the most relevant tokens are prioritized, optimizing the model's focus on critical features for improved performance in XMTC tasks.



(a) Contemporary SOTA XMTC Architecture

(b) Our Proposed PLANT Architecture

Figure 1: Left—architecture for contemporary SOTA models; Right—PLANT model with additional components. PLANT integrates traditional SOTA components (black-bordered box), multi-label attention (dotted orange box), and the novel planted attention and mutual information gain (green-bordered boxes) for enhanced label prediction.

3. **Inattention**: Introduced the *inattention* technique to filter out less relevant tokens, sharpening the model's focus on key elements within a token sequence.

4. **Stateful Decoder**: Accumulates information across segments, enabling cumulative predictions. This approach improves adaptability to large documents, eliminates text truncation, ensures stable GPU memory usage, and enhances performance.

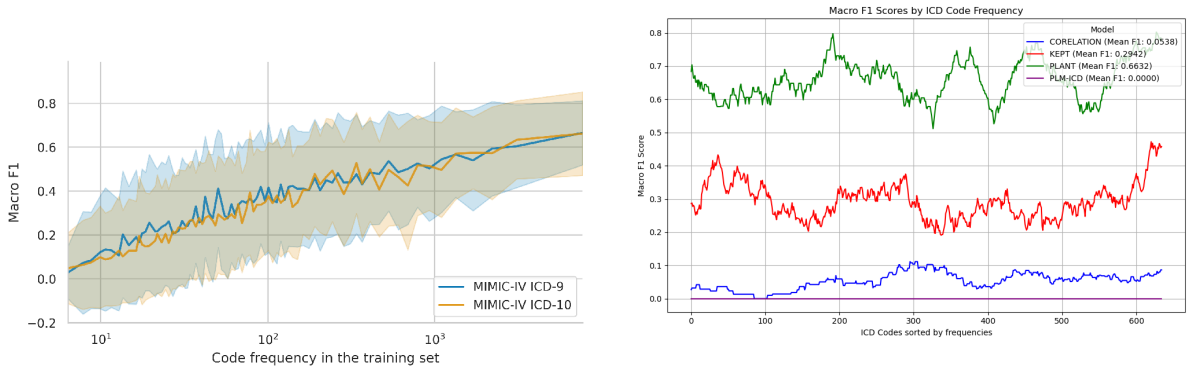
Is PLANT's Complexity Justified?

Although the inclusion of the L2R model introduces computational overhead, the resulting efficiency gains more than justify this complexity. PLANT outperforms few-shot SOTA Models. As shown in Figure 2b, the improvements are significant—**PLANT achieves a mean macro-F1score of** 0.6632**, more than double that of KEPT and GP (Yang et al., 2023; Luo et al., 2024), models specifically designed for few-shot learning, which reach only** 0.2942. This clearly demonstrates that the "juice is worth the squeeze" in terms of both performance and computational trade-offs (Refer to section 4.2).

Interpretability and Overfitting: To address concerns regarding the complexity of attention mechanisms and their effect on interpretability, we conducted an evaluation detailed in Appendix A.12. The results indicate that PLANT 's attention mechanism effectively identifies medically relevant terms. Additionally, the overfitting study in Appendix A.9.3 was motivated by the challenges of data scarcity in ICD coding.

2 Related Work

ICD: Deep neural models (Li and Yu, 2020; Vu et al., 2021; Biswas et al., 2021) have advanced through supplementary information and hierarchical structures, as demonstrated by Zhou et al. (2021) and Yuan et al. (2022) with label descriptions, alongside Cao et al. (2020) and Vu et al. (2021) exploring hierarchical learning. Recent developments include LSTM-based tree structures, adversarial learning (Xie et al., 2019), and condensed memory networks (Prakash et al., 2017), as well as hierarchical GRUs (Baumel et al., 2017).



(a) (b)

Figure 2: (a) Comparative analysis of model performance from (Edin et al., 2023) on rare versus common ICD diagnosis codes, highlighting that rare codes have near zero macro-F1 scores. (b) Comparison of the Macro-F1 scores for rare codes between PLANT and other models on the `MIMIC-III-few` dataset.

Further contributions feature convolutional and multi-scale attention networks (Xie et al., 2019; Li and Yu, 2020), graph convolution and hyperbolic representations (Cao et al., 2020), and LSTM-based attention models (Vu et al., 2021). Shared representation networks (Zhou et al., 2021), effective convolutional networks (Liu et al., 2021), and multi-synonym attention networks (Yuan et al., 2022) have also been proposed to enhance ICD coding performance. Recently, Zhang and Wang (2024) introduced AHDD for distilling medical notes using hierarchical code descriptions, while Luo et al. (2024) developed CoRelation to improve ICD code learning through contextual relationships. Additionally, Lu et al. (2023) employed contrastive learning to address data variability, and Li et al. (2023) tackled data imbalance with a knowledge-enhanced Graph Attention Network, leveraging a heterogeneous text graph and auxiliary healthcare tasks.

Pretrained Large Language Models (PLMs): PLMs in ICD face challenges like high computational costs and overfitting (Huang et al., 2022; Michalopoulos et al., 2022; Ng et al., 2023; Kang et al., 2023). Efforts like KEPT (Yang et al., 2022) and HiLAT (Liu et al., 2022) have improved PLM performance using prompt-based predictions and hierarchical encoding, but efficiency issues persist. KEPT, for example, uses Longformer (Beltagy et al., 2020) with contrastive learning and a prompt framework, but its reliance on extensive parameters and long inputs limits training practicality.

Few Shot: Song et al. (2021) introduced a GAN-based framework for zero-shot ICD coding, generating latent features for unseen codes by leveraging the ICD hierarchy and reconstructing code-relevant keywords. Yang et al. (2022) developed KEPT-Longformer, a prompt-based fine-tuning model that injects domain-specific knowledge and uses contrastive learning to improve rare code assignments. Chen et al. (2023) proposed a relation-enhanced code encoder that strengthens inter-code connections through hierarchical structures, improving rare code predictions without relying on extensive external knowledge. Yang et al. (2023) addressed the long-tail challenge by transforming ICD coding into an autoregressive generation task, using a novel prompt template and SOAP structure to effectively handle few-shot scenarios.

XMTC: Zhu and Zamani (2023) proposed ICXML, which utilizes in-context learning for zero-shot extreme multi-label classification, enhancing model adaptability. Xiong et al. (2023) focused on re-ranking to improve prediction accuracy, while Shi et al. (2024) contributed RDE and MatchXML, addressing residual connections and match-based strategies. Gupta et al. introduced a dual encoder (DE) framework for improved classification, and Kharbanda et al. (2023) proposed InceptionXML, which integrates inception modules for better feature extraction. Kharbanda et al. (2024)'s GANDALF presents a generative approach to label correlation. Chai et al. (2024) tackled compositional generalization in XMTC with data-augmentation.

3 Our Approach

Intuition behind PLANT (Figure 1b): The intuitive flow starts with document tokenization into embeddings processed by a pretrained AWD-

LSTM to grasp textual contexts. The decoder introduces *planted attention* (green box), leveraging a L2R model's ability to rank token significance by label relevance, enriching the model with a pre-understanding of token-label dynamics. This is adeptly paired with multi-label attention (dotted orange box), merging learned and pretrained insights for feature prominence. Additionally, mutual information gain is utilized to enhance the decision-making process by calculating the relevance of each token to the potential labels, providing an informed basis for further attention refinement. A subsequent boost attention phase fine-tunes this for label-specific discernment, culminating in a sigmoid-derived label probability prediction. Section 3.1 provides a detailed description of the L2R model components and the use of Mutual Information Gain to bootstrap the L2R model, while Section 3.2 explains how we utilize the pretrained L2R model for planted attention, illustrating the integration of the green boxes in Figure 1b.

3.1 Pretraining L2R Model

L2R Model: An L2R model is used to determine which words in a text are most relevant to specific labels. We start with a set of labels (e.g., medical diagnoses) and a set of words from medical texts. Each word is given a relevance score for each label, indicating how important that word is for the label. Both labels and words are represented using word embeddings, which are numerical representations that capture their meanings. For each combination of a label and a word, we create a feature vector by combining their embeddings, capturing the relationship between the label and the word. The L2R model uses these feature vectors to learn a ranking function, which is trained to output a score for each word-label pair, indicating how relevant the word is to the label. During training, the model learns to rank words based on their relevance to labels, improving over time at identifying which words are most important for each label. The L2R model ensures that attention mechanism in the decoder starts with well-informed weights rather than random ones. This helps the model focus on the most relevant parts of the text right from the beginning, leading to faster and more effective training.

Mutual Information Gain: Mutual Information Gain bootstraps our L2R model, helping it understand the relationship between labels and tokens in our data. We treat the presence or absence of a label (e.g., a specific medical diagnosis) and the presence or absence of a token (a word in a medical text) as random events. Mutual Information Gain measures how much knowing the presence of a token gives us information about the presence of a label, quantifying the strength of their relationship. We calculate it by comparing the joint probability of the label and token occurring together to the probabilities of each occurring independently. These scores are used as relevance scores, indicating important word-label pairs. Using these scores, we bootstrap our L2R model, starting training with a good understanding of which words are important for which labels.

3.2 Leveraging L2R as Pretrained Attention

Pretrained and Fine-tuned AWD-LSTM: We use a pretrained AWD-LSTM model as our LLM to process word sequences. This model, pretrained on a large corpus, captures general language patterns. We further fine-tune it, adapting the model to our specific task to enhance its ability to extract relevant information.

Decoder – PLANT L2R as Attention: Allocation of label-specific attention weights to the most informative tokens in a sequence is a four-step process.

Step 1 (Traditional Learned Attention): We extract hidden features from each word, organize them into a matrix, and compute label-specific attention weights by comparing these features with label embeddings. Applying softmax column-wise emphasizes the most relevant words, creating a matrix where each column represents a label's focus. These learned attention weights help the model highlight significant tokens.

Step 2 (Our Planted Attention): We utilize two types of attention weights: static-planted ($\boldsymbol{S}$) and differentiable-planted ($\boldsymbol{P}$). Static-planted attention remains constant throughout the model's training. It is determined based on mutual information gain, which provides a fixed measure of how relevant each word is to each label. $\boldsymbol{S}$ consists of vectors $\boldsymbol{g}^{(i)}$ for each label. Each element $g_j^{(i)}$ in these vectors indicates the relevance of token t_j with respect to label $l^{(i)}$. By using these fixed relevance scores, the model can consistently prioritize tokens that are inherently important to each label, as defined in the L2R model section. Unlike static-planted attention, differentiable-planted attention involves trainable parameters that can be adjusted during model training. We calculate this attention by first

creating feature vectors for each label-token pair, using word embeddings to represent their meanings. These feature vectors are computed as per the L2R model. The pretrained L2R model uses these feature vectors to calculate scores for differentiable-planted attention. The matrix $\boldsymbol{P}$ is formed, containing vectors $\boldsymbol{p}^{(i)}$ for each label, where each score reflects the dynamic relevance of a token to a label. This attention type allows the model to adapt and fine-tune the importance of tokens as it learns.

Step 3 (Inattention Technique): We introduce *inattention*, a method that enhances attention by filtering out less relevant tokens. By applying a threshold to the differentiable-planted attention scores before the softmax function, we zero out weights for less important tokens, focusing the model on the top k relevant ones. The optimal threshold k is consistently found through hyperparameter tuning within the range $[1, 10k']$, where k' is derived from the nDCG@k loss function used in the L2R model pretraining.

Step 4 (Boosting Attention): We combine learned, static-planted, and differentiable-planted attention weights to compute label-specific vectors. This involves a linear combination of token hidden features, followed by an element-wise multiplication with a trainable weight matrix $\boldsymbol{W}$. The resulting matrix $\boldsymbol{V}$ captures attention insights, with each row representing the label specific key information.

Predictions and Training Objective: To make predictions, we sum the label-specific information, add a label-specific bias, and pass it through a sigmoid activation to produce probability scores for each label: $\hat{\boldsymbol{y}} = \text{sigmoid}(\mathbf{1}\boldsymbol{V}^T + \boldsymbol{b})$. These scores indicate the likelihood of each label applying to the token sequence. The model is trained by minimizing binary cross-entropy loss, which measures the difference between predicted probabilities and actual labels, thereby improving prediction accuracy.

Stateful Decoder: To enhance the decoder's ability to maintain context across token sequences we employ a *stateful* mechanism inspired by Howard and Ruder (2018). To incorporate this state into the decoder, we modify the prediction equation to include a hidden state that accumulates contextual information across the sequence of tokens. After each time step t, we update the decoder state $\boldsymbol{D}_t$ using the previous state $\boldsymbol{D}_{t-1}$ and the current label-specific vector $\boldsymbol{V}_t$: $\boldsymbol{D}_t = \phi(\boldsymbol{D}_{t-1}, \boldsymbol{V}_t)$.
Here ϕ represents a nonlinear function that allows the state to accumulate information from each token's contribution to the labels. After processing all n tokens in the sequence, we use the final accumulated state $\boldsymbol{D}_n$ to make the prediction, $\hat{\boldsymbol{y}} = \text{sigmoid}\left(\mathbf{1}\left(\boldsymbol{D}_n^T\right) + \boldsymbol{b}\right)$

Discriminative Fine-tuning and Gradual Unfreezing: To fine-tune our pretrained model for attention planting, we use two key strategies. First, we apply *discriminative fine-tuning*, assigning different learning rates to parameter groups (encoder, planted decoder, and other components) to optimize areas needing the most adjustment. We use a smaller learning rate for the pretrained L2R model parameters. Second, we implement *gradual unfreezing*, fine-tuning the model layer by layer, starting from the last layer and moving toward the first.

4 Experiments

Datasets, Baselines & Implementation: We evaluated PLANT against SOTA ICD coding models using the `MIMIC-III-full` and `MIMIC-IV-full` datasets, which consist of discharge summaries annotated with ICD-9 and ICD-10 codes, respectively. For few-shot learning, we used specialized subsets like `MIMIC-III-rare50` and `MIMIC-III-few` to focus on rare codes. We also evaluated PLANT on publicly available XMTC datasets like `EURLEX-4K` and `WIKI10-31K`. Detailed dataset descriptions, implementation details, evaluation metrics and baselines are provided in Appendix A.6, A.8 and A.7, respectively.

4.1 Discussion of Main Results (ICD)

`MIMIC-III-full` (Table 2) **`MIMIC-III-top50`** (Table 10) **`MIMIC-III-rare50`** (Table 3) **`MIMIC-III-few`** (Table 4) **`MIMIC-IV-full`** (Table 5): PLANT outperforms SOTA models across multiple ICD datasets. On the `MIMIC-III-full` test set, PLANT achieves the highest scores in macro and micro AUC (96.1% and 99.9%), macro and micro F1 (14.5% and 60.2%), and precision at various ranks, including P@5 (85.1%), P@8 (77.7%), and P@15 (61.8%). Similarly, on the MIMIC-III-top50 test set, PLANT leads in macro and micro AUC (95.1% and 95.9%), macro and micro F1 (69.7% and 73.1%), and outperforms other models in P@8 (55.9%) and P@15 (36.3%). In few-shot scenarios, PLANT excels even further. On the MIMIC-III-rare50 test set, it delivers outstanding macro and micro F1 scores (82.6% and 84.2%), with AUC scores of 95.6% and 96.0%, far surpassing other models. Notably, these

results were achieved with only unfrozen PLANT layers, highlighting its efficiency and potential. On the MIMIC-III-few test set, PLANT achieves macro and micro F1 scores of 66.3% and 71.0%, more than doubling the performance of closest competitors, and excels in precision and recall, with macro precision at 65.1%, micro precision at 68.6%, and recall scores of 81.0% (macro) and 81.7% (micro). On the MIMIC-IV test set, PLANT achieves the highest macro and micro AUC scores (98.1% and 99.6%) and leads in macro and micro F1 (21.5% and 58.9%), as well as precision at P@5 (78.1%), P@8 (70.6%), and P@15 (55.6%).

4.2 Figure 2a vs. Figure 2b: Why PLANT Is Superior to Few-Shot SOTA Models?

Figure 2a shows codes with frequencies < 10 have near-zero macro-F1 scores, highlighting the challenge of predicting rare codes—a problem PLANT aims to address. To evaluate this, we used the `MIMIC-III-few` dataset, which contains 685 codes, each appearing in < 5 instances. Figure 2b focuses specifically on these rare codes, essentially zooming in on the leftmost portion of Figure 2a. It compares macro-F1 scores of models not specialized for rare code prediction (PLM-ICD) with models like Corelation and KEPT, the current SOTA for rare codes. PLANT (green line) achieves a mean macro-F1 score of 0.6632, more than double that of KEPT (red line), which has a mean score of 0.2942. CoRelation (blue line) and PLM-ICD (purple line) lag significantly behind, with mean macro-F1 scores of 0.0538 and 0.0000, respectively. Figure 2b clearly demonstrates PLANT outperforms even models explicitly designed for few-shot learning like KEPT.

Model	AUC		F1		P@k		
	Macro	Micro	Macro	Micro	P@5	P@8	P@15
CAML/DR-CAML	89.7	98.6	8.8	53.9	-	70.9	56.1
MSATT-KG	91.0	99.2	9.0	55.3	-	72.8	58.1
MultiResCNN	91.0	98.6	8.5	55.2	-	73.4	58.4
HyperCore	93.0	98.9	9.0	55.1	-	72.2	57.9
LAAT/JointLAAT	92.1	98.8	10.7	57.5	81.3	73.8	59.1
ISD	93.8	99.0	11.9	55.9	-	74.5	-
Effective-CAN	92.1	98.9	10.6	58.9	-	75.8	60.6
MSMN	95.0	99.2	10.3	58.4	-	75.2	59.9
DiscNet	95.6	99.3	14.0	58.8	-	76.5	61.4
AHDD	95.2	99.3	10.9	58.9	-	75.3	-
CoRelation	95.2	99.2	10.2	59.1	83.4	76.2	60.7
JointLAAT (/w Contrastive)	-	-	11.4	58.8	-	75.6	60.2
KEMTL	95.3	99.6	12.7	58.3	-	75.6	-
PLM-ICD	92.6	98.9	10.4	59.8	84.4	77.1	61.3
PLANT (Ours)	**96.1**	**99.9**	**14.5**	**60.2***	**85.1***	**77.7***	**61.8***

Table 2: Results (in %) on the `MIMIC-III-full` test set. Refer to Appendix A.10 for detailed setup.

Model	AUC		F1	
	Macro	Micro	Macro	Micro
MultiResCNN (/w Contrastive)	-	-	22.8	23.3
HyperCore (/w Contrastive)	-	-	23.4	25.2
EffectiveCAN (/w Contrastive)	-	-	27.1	28.0
PLM-ICD (/w Contrastive)	-	-	30.3	29.5
Hierarchical (/w Contrastive)	-	-	32.0	31.3
MSMN (/w Contrastive)	-	-	31.2	30.6
KEPTLongformer	82.7	83.3	30.4	32.6
PLANT (Ours)	**95.6***	**96.0***	**82.6***	**84.2***

Table 3: Results on the `MIMIC-III-rare50` test set.

Model	F1		Precision		Recall	
	Macro	Micro	Macro	Micro	Macro	Micro
MSMN	4.3	8.5	4.5	70.9	4.2	4.5
AGMHT	18.7	29.2	17.6	49.4	19.9	20.7
GP	30.2	35.3	27.9	38.5	32.9	32.6
PLANT (Ours)	**66.3***	**71.0**	**65.1***	**68.6***	**81.0***	**81.7***

Table 4: Results on the `MIMIC-III-few` test set.

5 Ablation Analysis

(A)Ablating Pretrained L2R in PLANT Demonstrating the impact of Step 2 (Our Planted Attention) in Section 3.2, by conducting two types of ablation analysis on the Pretrained L2R activations. **(A1)Ablating Pretrained L2R in few-shot setting** (Figure 3): We evaluate PLANT and LAAT in contexts with skewed label distributions. PLANT uses pretrained L2R activations $\boldsymbol{P}$ and mutual information gain $\boldsymbol{S}$, initializing the decoder's attention weights. While LAAT relies solely on learned attention $\boldsymbol{A}$, initialized randomly and learned from scratch. In other words LAAT omits "*Our Planted Attention*"—$\boldsymbol{P}$ and $\boldsymbol{S}$. This comparison between PLANT and LAAT serves as an ablation study on the effect of the pretrained L2R activations (P).
Specifically chose LAAT for this comparison: As it uses an identical architecture to ours, with the only difference being the pretrained L2R activations. This provides a clear understanding of the role that P plays in our model's performance.

PLANT & LAAT were trained across varying fractions of a balanced training dataset for up to 5 epochs. The test set remains constant, and we measured P@5 and P@15 for both models. PLANT consistently matched or surpassed the LAAT model's performance across all training sizes, even with significantly less data. `MIMIC-IV-full`—PLANT achieved a P@5 of 0.50 and P@15 of 0.37 with a smaller training split of 1090 and 2743 instances, respectively, matching the performance of LAAT trained on a significantly larger split of $10,337$ and $12,902$ instances. `MIMIC-III-full`—PLANT achieved a

Model	AUC		F1		P@k		
	Macro	Micro	Macro	Micro	P@5	P@8	P@15
CAML/DR-CAML	91.1	98.5	16.0	55.4	-	66.8	52.2
MultiResCNN	94.5	99.0	21.1	56.9	-	67.8	53.5
LAAT/JointLAAT	95.4	99.0	20.3	57.9	-	68.9	54.3
PLM-ICD	91.9	99.0	21.1	58.5	-	69.9	55.0
CoRelation	97.2	**99.6**	6.3	57.8	-	70.0	-
PLANT (Ours)	**98.1***	**99.6**	**21.5***	**58.9***	**78.1***	**70.6***	**55.6***

Table 5: Results on the `MIMIC-IV-full` test set. The comparitive results are reported from Edin et al. (2023).

Model	EURLEX-4K			WIKI10-31K		
	P@1	P@3	P@5	P@1	P@3	P@5
PECOS	88.41	75.97	63.18	88.69	80.17	70.91
ICXML	87.10	74.32	62.7	87.61	79.11	69.78
XRR	87.96	78.88	68.52	89.54	85.38	**81.34**
X-Transformer w/ RDE	84.60	72.61	61.35	86.15	76.99	68.75
MatchXML	88.12	75.00	62.22	89.30	80.45	70.89
DE	87.60	74.39	67.80	88.21	80.29	69.91
InceptionXML+GANDALF	86.98	75.89	68.78	88.76	80.32	69.89
CG	87.82	76.71	68.42	87.29	79.81	68.45
PLANT (Ours)	**89.92***	**80.42***	**69.38***	**90.10***	**86.67***	81.33

Table 6: XMTC Results.(Discussed in Appendix A.9.2)

P@5 of 0.47 and P@15 of 0.30, trained with only 136 and 235 instances, respectively. This performance matches that of the LAAT model trained on datasets with 1342 and 1578 instances. Figure 3 visually demonstrates the substantial efficiency gains of PLANT regarding training data requirements while maintaining or enhancing performance. PLANT matches LAAT with significantly less data and fewer instances per label.

(A2)Ablating Pretrained L2R in a full setting (Table 7): `MIMIC-III-full`—training PLANT for 25 epochs with planted attention improved P@15 from **59.6** to **61.8**, compared to traditional attention for same number of epochs. `MIMIC-IV-full`—training PLANT for 25 epochs, applying 1/3 of the lr to the pretrained L2R parameters , led to an increase in P@15 from **54.1** to **55.6**.

Ablating Inattention & Stateful Decoder (Table 7): Investigated the impact of the inattention threshold k on `MIMIC-III-full` and `MIMIC-IV-full`. The training splits comprised 22,525 instances (average 49 instances per label) and 49,579 instances (average 97 instances per label) for the respective datasets. We trained each model for 5 epoch and measured P@15. `MIMIC-III-full`—the model without inattention ($k = 72$) achieved a P@15 of 50.95, while the model with inattention ($k = 56$) achieved a slightly higher P@15 of 51.05. `MIMIC-IV-full`—the model without inattention attained a P@15 of 42.4, which improved to 42.51 with inattention ($k = 8$). `MIMIC-III-full`—using the stateful decoder for three epochs yielded a P@15 of 52.9, a slight improvement over 52.8 without it.

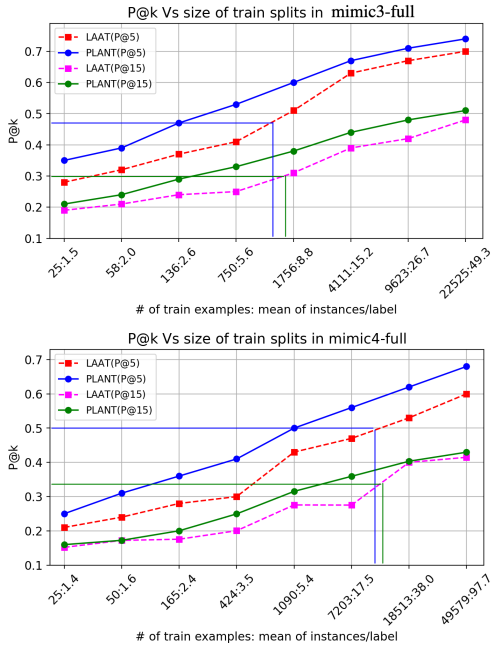


Figure 3: P@15 for PLANT & LAAT with varying numbers of training examples.

Ablation	`MIMIC-III-full`	`MIMIC-IV-full`
Without PLANT	59.60	54.10
With PLANT	61.8	55.60
Without Inattention	50.95	42.40
With Inattention	51.05	42.51
Stateless	52.80	43.38
Stateful	52.90	44.22
− disc	51.40	43.29
+ disc	52.21	44.34
full unfreezing	57.78	49.78
gradual unfreezing	58.31	50.97

Table 7: P@15 Ablations (train-split `MIMIC-IV-full` PLANT ablation 89, 142 and 49, 579 for rest).

`MIMIC-IV-full`—(training split of 49, 579), employing the stateful decoder for seven epochs significantly boosted P@15, from 43.28 to 44.22.

Ablating Discriminative Fine-tuning & Gradual Unfreezing (GU): `MIMIC-III-full`—training PLANT for 1 epoch with discriminative fine-tuning, applying 1/2 the lr to L2R parameters, improved P@15 from 51.40 to 52.21 on the test set. `MIMIC-IV-full`—(train split 49, 579), training PLANT for 7 epochs with a 1/3 of the lr for L2R parameters increased P@15 from 43.29 to 44.34. For GU we explored 2 scenarios: one gradually unfreezing the model layer by layer, and the other unfreezing the entire model simultaneously. Both models were trained for 10 epochs. `MIMIC-III-full`—GU increased P@15 from 57.78 to 58.31; `MIMIC-IV-full`— from 49.78 to 50.97.

Limitations

The PLANT method, while effective, presents a notable trade-off in terms of computational resources. The necessity to pretrain and load the L2R model imposes a substantial memory overhead compared to traditional attention mechanisms. Consequently, our memory constraints limited the number of epochs for which PLANT could be trained. This aspect of PLANT, particularly its scalability to larger XMTC datasets, warrants further investigation. Future work will explore strategies to optimize memory usage, ensuring that the benefits of PLANT can be harnessed more broadly without the current limitations on training duration and dataset size.

Broader Impacts and Ethical Considerations

Our research contributes to the broader field of natural language processing (NLP) and machine learning (ML), advancing the SOTA in XMTC. In the context of XMTC, our research has the potential to significantly impact various sectors, including healthcare, finance, and e-commerce. By automating labor-intensive tasks such as medical coding and diagnosis, these models can enhance healthcare accessibility, particularly in underserved communities. This can lead to improved patient outcomes and reduced disparities in healthcare access. Additionally, in education, XMTC models can support personalized learning experiences by categorizing educational resources and recommending tailored learning materials to students. Furthermore, XMTC can contribute to policy development by analyzing public opinion and sentiment from social media and news sources, providing valuable insights for policymakers to develop evidence-based policies and interventions. These applications demonstrate the diverse and far-reaching societal implications of XMTC technology. However, we acknowledge the importance of ensuring that automated systems do not perpetuate biases or discrimination present in the data. Therefore, we prioritize fairness, transparency, and accountability in our model development process. In summary, while our research presents exciting opportunities for automation and efficiency gains, we recognize the importance of ethical considerations and broader societal impacts. By upholding ethical principles and promoting responsible AI development, we aim to maximize the positive impact of our work while mitigating potential risks.

References

Tal Baumel, Jumana Nassour-Kassis, Raphael Cohen, Michael Elhadad, and Noémie Elhadad. 2017. Multi-label classification of patient notes a case study on icd code assignment. *arXiv preprint arXiv:1709.09587*.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Biplob Biswas, Thai-Hoang Pham, and Ping Zhang. 2021. Transicd: Transformer based code-wise attention model for explainable icd coding. In *Artificial Intelligence in Medicine: 19th International Conference on Artificial Intelligence in Medicine, AIME 2021, Virtual Event, June 15–18, 2021, Proceedings*, pages 469–478. Springer.

Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81.

Pengfei Cao, Yubo Chen, Kang Liu, Jun Zhao, Shengping Liu, and Weifeng Chong. 2020. HyperCore: Hyperbolic and co-graph representation for automatic ICD coding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3105–3114, Online. Association for Computational Linguistics.

CDC. 2024. International classification of diseases, tenth revision, clinical modification (icd-10-cm). Accessed: 2024-10-14.

Yuyang Chai, Zhuang Li, Jiahui Liu, Lei Chen, Fei Li, Donghong Ji, and Chong Teng. 2024. Compositional generalization for multi-label text classification: A data-augmentation approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17727–17735.

Jiamin Chen, Xuhong Li, Junting Xi, Lei Yu, and Haoyi Xiong. 2023. Rare codes count: Mining inter-code relations for long-tail clinical text classification. In *Proceedings of the 5th Clinical Natural Language Processing Workshop*, pages 403–413.

Xiang Dai, Ilias Chalkidis, Sune Darkner, and Desmond Elliott. 2022. Revisiting transformer-based models for long document classification. *arXiv preprint arXiv:2204.06683*.

Joakim Edin, Alexander Junge, Jakob D Havtorn, Lasse Borgholt, Maria Maistro, Tuukka Ruotsalo, and Lars Maaløe. 2023. Automated medical coding on mimic-iii and mimic-iv: A critical review and replicability study. *arXiv preprint arXiv:2304.10909*.

Nilesh Gupta, Fnu Devvrit, Ankit Singh Rawat, Srinadh Bhojanapalli, Prateek Jain, and Inderjit S Dhillon. Dual-encoders for extreme multi-label classification. In *The Twelfth International Conference on Learning Representations*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Chao-Wei Huang, Shang-Chi Tsai, and Yun-Nung Chen. 2022. Plm-icd: automatic icd coding with pretrained language models. *arXiv preprint arXiv:2207.05289*.

Alistair EW Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Steven Horng, Tom J Pollard, Sicheng Hao, Benjamin Moody, Brian Gow, et al. 2023. Mimic-iv, a freely accessible electronic health record dataset. *Scientific data*, 10(1):1.

Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9.

Beichen Kang, Xiaosu Wang, Yun Xiong, Yao Zhang, Chaofan Zhou, Yangyong Zhu, Jiawei Zhang, and Chunlei Tang. 2023. Automatic icd coding based on segmented clinicalbert with hierarchical tree structure learning. In *International Conference on Database Systems for Advanced Applications*, pages 250–265. Springer.

Siddhant Kharbanda, Atmadeep Banerjee, Devaansh Gupta, Akash Palrecha, and Rohit Babbar. 2023. Inceptionxml: A lightweight framework with synchronized negative sampling for short text extreme classification. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 760–769.

Siddhant Kharbanda, Devaansh Gupta, Erik Schultheis, Atmadeep Banerjee, Cho-Jui Hsieh, and Rohit Babbar. 2024. Gandalf: Learning label-label correlations in extreme multi-label classification via label features. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, page 1360–1371, New York, NY, USA. Association for Computing Machinery.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Fei Li and Hong Yu. 2020. Icd coding from clinical text using multi-filter residual convolutional neural network. In *proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8180–8187.

Xinhang Li, Xiangyu Zhao, Yong Zhang, and Chunxiao Xing. 2023. Towards automatic icd coding via knowledge enhanced multi-task learning. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 1238–1248.

Leibo Liu, Oscar Perez-Concha, Anthony Nguyen, Vicki Bennett, and Louisa Jorm. 2022. Hierarchical label-wise attention transformer model for explainable icd coding. *Journal of biomedical informatics*, 133:104161.

Yang Liu, Hua Cheng, Russell Klopfer, Matthew R. Gormley, and Thomas Schaaf. 2021. Effective convolutional attention network for multi-label clinical document classification. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5941–5953, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Chang Lu, Chandan Reddy, Ping Wang, and Yue Ning. 2023. Towards semi-structured automatic icd coding via tree-based contrastive learning. *Advances in Neural Information Processing Systems*, 36:68300–68315.

Junyu Luo, Xiaochen Wang, Jiaqi Wang, Aofei Chang, Yaqing Wang, and Fenglong Ma. 2024. Corelation: Boosting automatic icd coding through contextualized code relation learning. *arXiv preprint arXiv:2402.15700*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*.

George Michalopoulos, Michal Malyska, Nicola Sahar, Alexander Wong, and Helen Chen. 2022. Icdbigbird: a contextual embedding model for icd code classification. *arXiv preprint arXiv:2204.10408*.

James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. 2018. Explainable prediction of medical codes from clinical text. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1101–1111, New Orleans, Louisiana. Association for Computational Linguistics.

Clarence Boon Liang Ng, Diogo Santos, and Marek Rei. 2023. Modelling temporal document sequences for clinical icd coding. *arXiv preprint arXiv:2302.12666*.

Anthony N Nguyen, Donna Truran, Madonna Kemp, Bevan Koopman, David Conlan, John O'Dwyer, Ming Zhang, Sarvnaz Karimi, Hamed Hassanzadeh, Michael J Lawley, et al. 2018. Computer-assisted diagnostic coding: effectiveness of an nlp-based approach using snomed ct to icd-10 mappings. In *AMIA Annual Symposium Proceedings*, volume 2018, page 807. American Medical Informatics Association.

Thanh-Tung Nguyen, Viktor Schlegel, Abhinav Kashyap, Stefan Winkler, Shao-Syuan Huang, Jie-Jyun Liu, and Chih-Jen Lin. 2023. Mimic-iv-icd: A new benchmark for extreme multilabel classification. *arXiv preprint arXiv:2304.13998*.

Aaditya Prakash, Siyuan Zhao, Sadid Hasan, Vivek Datla, Kathy Lee, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2017. Condensed memory networks for clinical diagnostic inferencing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Jiang-Xin Shi, Tong Wei, and Yu-Feng Li. 2024. Residual diverse ensemble for long-tailed multi-label text classification. *Science CHINA Information Science*.

Congzheng Song, Shanghang Zhang, Najmeh Sadoughi, Pengtao Xie, and Eric Xing. 2021. Generalized zero-shot text classification for icd coding. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 4018–4024.

Thanh Vu, Dat Quoc Nguyen, and Anthony Nguyen. 2021. A label attention model for icd coding from clinical text. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI'20.

WHO. 2019. *International Statistical Classification of Diseases and Related Health Problems*, 11th edition.

Xiancheng Xie, Yun Xiong, Philip S. Yu, and Yangyong Zhu. 2019. Ehr coding with multi-scale feature attention and structured knowledge graph propagation. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 649–658, New York, NY, USA. Association for Computing Machinery.

Jie Xiong, Li Yu, Xi Niu, and Youfang Leng. 2023. Xrr: Extreme multi-label text classification with candidate retrieving and deep ranking. *Information Sciences*, 622:115–132.

Zhichao Yang, Sunjae Kwon, Zonghai Yao, and Hong Yu. 2023. Multi-label few-shot icd coding as autoregressive generation with prompt. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 5366–5374.

Zhichao Yang, Shufan Wang, Bhanu Pratap Singh Rawat, Avijit Mitra, and Hong Yu. 2022. Knowledge injected prompt based fine-tuning for multi-label few-shot icd coding. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, volume 2022, page 1767. NIH Public Access.

Hsiang-Fu Yu, Kai Zhong, Jiong Zhang, Wei-Cheng Chang, and Inderjit S Dhillon. 2022. Pecos: Prediction for enormous and correlated output spaces. *Journal of Machine Learning Research*, 23(98):1–32.

Zheng Yuan, Chuanqi Tan, and Songfang Huang. 2022. Code synonyms do matter: Multiple synonyms matching network for automatic ICD coding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 808–814, Dublin, Ireland. Association for Computational Linguistics.

Bin Zhang and Junli Wang. 2024. A novel icd coding framework based on associated and hierarchical code description distillation. *arXiv preprint arXiv:2404.11132*.

Shurui Zhang, Bozheng Zhang, Fuxin Zhang, Bo Sang, and Wanchun Yang. 2022. Automatic ICD coding exploiting discourse structure and reconciled code embeddings. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 2883–2891, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Tong Zhou, Pengfei Cao, Yubo Chen, Kang Liu, Jun Zhao, Kun Niu, Weifeng Chong, and Shengping Liu. 2021. Automatic ICD coding via interactive shared representation networks with self-distillation mechanism. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5948–5957, Online. Association for Computational Linguistics.

Yaxin Zhu and Hamed Zamani. 2023. Icxml: An in-context learning framework for zero-shot extreme multi-label classification. *arXiv preprint arXiv:2311.09649*.

A Appendix

A.1 Skewness of Codes

Figure 4 demonstrates skewness of ICD codes.

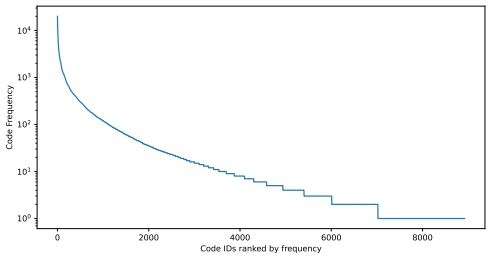


Figure 4: The skewness of ICD-9-CM code distribution for MIMIC-III (Johnson et al., 2016).

A.2 L2R Model (continued from Section 3.1)

We use superscript to denote the id of a label and subscript to denote the id of a token. The training set of the L2R model contains a set of labels $\mathcal{L} = \left\{l^{(1)}, l^{(2)}, \cdots, l^{(m)}\right\}$, and a set of tokens $\mathcal{T} = \{t_1, t_2, \cdots, t_n\}$. Furthermore, $\boldsymbol{G} = \left[\boldsymbol{g}^{(1)}, \boldsymbol{g}^{(2)}, \cdots, \boldsymbol{g}^{(m)}\right] \in \mathbb{R}^{n \times m}$, and $\boldsymbol{g}^{(i)} = \left[g_1^{(i)}, g_2^{(i)}, \cdots, g_n^{(i)}\right]^T \in \mathbb{R}^n$, where $g_j^{(i)}$ denotes the relevance of the token t_j with respect to label $l^{(i)}$. We represent each label $l^{(i)}$ and token t_j with

word embeddings $\boldsymbol{e}_{l^{(i)}}$ and $\boldsymbol{e}_{t_j}$, respectively. A feature vector

$$\boldsymbol{x}_{\boldsymbol{j}}^{(\boldsymbol{i})} = \Psi\left(\boldsymbol{e}_{l^{(i)}}, \boldsymbol{e}_{t_j}\right) \tag{1}$$

is created from each label-token pair $\left(l^{(i)}, t_j\right), i = 1, 2, \cdots, m; j = 1, 2, \cdots, n$, by concatenating the corresponding word embeddings $\boldsymbol{e}_{l^{(i)}}$ and $\boldsymbol{e}_{t_j}$. The feature matrix, $\boldsymbol{X}^{(\boldsymbol{i})} = \left[\boldsymbol{x}_{\boldsymbol{1}}^{(\boldsymbol{i})}, \cdots, \boldsymbol{x}_{\boldsymbol{n}}^{(\boldsymbol{i})}\right]$ and the corresponding scores, $\boldsymbol{g}^{(\boldsymbol{i})} = \left[g_1^{(i)}, g_2^{(i)}, \cdots, g_n^{(i)}\right]^T$ then form an 'instance'. The training set can be denoted as $\left\{\left(\boldsymbol{X}^{(\boldsymbol{i})}, \boldsymbol{g}^{(\boldsymbol{i})}\right)\right\}_{i=1}^{m}$. The L2R model is associated with a ranking function, $f : \boldsymbol{x}_{\boldsymbol{j}}^{(\boldsymbol{i})} \mapsto \mathbb{R}$. At any point in the training, the model outputs the score $\boldsymbol{z}^{(\boldsymbol{i})} = \left[f\left(\boldsymbol{x}_{\boldsymbol{1}}^{(\boldsymbol{i})}\right), \cdots, f\left(\boldsymbol{x}_{\boldsymbol{n}}^{(\boldsymbol{i})}\right)\right]^T \in \mathbb{R}^n$. We direct readers to Appendix A.2 for detailed specifics about the L2R model, including our methods for bootstrapping it with mutual information gain and subsequent training procedures.

The ranking function, $f : \boldsymbol{x}_{\boldsymbol{j}}^{(\boldsymbol{i})} \mapsto \mathbb{R}$, of the L2R model is an L layered feed forward network,

$$f(\boldsymbol{x}_{\boldsymbol{j}}^{(\boldsymbol{i})}) = y^L, y^{(l)} = a(W^{(l)} \cdot y^{(l-1)} + b^{(l)}), \tag{2}$$

where $y^{(l)}$ is layer l output, $y^{(0)} = x$ is input, $W^{(l)}$ is layer l weight matrix, $b^{(l)}$ is layer l bias vector, and $a(\cdot)$ is the activation function. In our experiments we trained the L2R model with $L = 2$.

At any point in the training, the model outputs the score $\boldsymbol{z}^{(\boldsymbol{i})} = \left[f\left(\boldsymbol{x}_{\boldsymbol{1}}^{(\boldsymbol{i})}\right), \cdots, f\left(\boldsymbol{x}_{\boldsymbol{n}}^{(\boldsymbol{i})}\right)\right]^T \in \mathbb{R}^n$. The objective of the L2R model is to minimize the total loss,

$$\sum_{i=1}^{m} \mathsf{nDCG@k}\left(\boldsymbol{z}^{(\boldsymbol{i})}, \boldsymbol{g}^{(\boldsymbol{i})}\right), \tag{3}$$

where nDCG@k is the maximum allowable DCG@k, which is defined as:

$$\mathsf{DCG@k}\left(\boldsymbol{z}^{(\boldsymbol{i})}, \boldsymbol{g}^{(\boldsymbol{i})}\right) := \sum_{l \in \mathsf{rank}_k(\boldsymbol{z}^{(\boldsymbol{i})})} \frac{2^{g_l^{(i)}}}{\log(l+1)}.$$ [1]

Bootstrapping L2R Model: Let (I, J) be a pair of random variables for the label $l^{(i)}$ and token t_j over the space $\mathcal{I} \times \mathcal{J}$, where $\mathcal{I} = \{\text{label } i \text{ present}, \text{label } i \text{ not present}\}$ and $\mathcal{J} = \{\text{token } j \text{ present}, \text{token } j \text{ not present}\}$. Then, g_j^i is defined as the mutual information gain of I and J:

$$g_j^{(i)} = \sum_{x \in \mathcal{I}, y \in \mathcal{J}} P_{(I,J)}(x, y) \log\left(\frac{P_{(I,J)}(x, y)}{P_I(x) P_J(y)}\right),$$

where $P_{(I,J)}$ is the joint, and P_I, P_J are the marginal probability mass function of I and J, respectively.

Training L2R Model: Gradient update rule to train the L2R model on $\left\{\left(\boldsymbol{X}^{(\boldsymbol{i})}, \boldsymbol{g}^{(\boldsymbol{i})}\right)\right\}_{i=1}^{m}$ are defined as follows. Let $I^{(i)}$ denote the set of pairs of token indices $\{j, k\}$, such that $g_j^{(i)} > g_k^{(i)}$. Also, let $z_j^{(i)} = f\left(\boldsymbol{x}_{\boldsymbol{j}}^{(\boldsymbol{i})}\right)$ and $z_k^{(i)} = f\left(\boldsymbol{x}_{\boldsymbol{k}}^{(\boldsymbol{i})}\right)$. The parameters of L2R model, $w_p \in \mathbb{R}$, are updated as (Burges, 2010):

$$\delta w_p = -\eta \sum_j \lambda_j \frac{\partial z_j^{(i)}}{\partial w_k},$$

$$\lambda_j = \sum_{k:\{j,k\} \in I^{(i)}} \lambda_{jk} - \sum_{k:\{k,j\} \in I^{(i)}} \lambda_{kj},$$

$$\lambda_{jk} = -\frac{\sigma}{1 + e^{\sigma\left(z_j^{(i)} - z_k^{(i)}\right)}} |\Delta\mathsf{nDCG@k}|_{jk},$$

where $|\Delta\mathsf{nDCG@k}|_{jk}$ denotes the change in nDCG@k by swapping j and k in $\mathsf{rank}(\boldsymbol{z}^{(\boldsymbol{i})})$.

A.3 Language Model: AWD-LSTM

We use the AWD-LSTM architecture (Merity et al., 2017) [2] as LM in our experiments. That means, AWD-LSTM model learns hidden features from a sequence of n tokens $\langle t_1, t_2, \cdots, t_n \rangle$, where each token is represented by word embedding $\boldsymbol{e}_{t_j} \in \mathbb{R}^{s_e}$. The hidden feature learned by AWD-LSTM corresponding to the jth token is represented as:

$$\boldsymbol{h_j} = \mathsf{AWD\text{-}LSTM}(\langle \boldsymbol{e}_{t_1}, \cdots, \boldsymbol{e}_{t_j} \rangle), \boldsymbol{h_j} \in \mathbb{R}^{s_e} \tag{4}$$

Note that all the pretrained word embeddings $\boldsymbol{e}_{t_j}$ and the parameters of the AWD-LSTM model are finetuned on the target task using the ULMFiT mechanisms proposed in Howard and Ruder (2018).

A.4 XMTC Decoder – PLANT L2R as Attention

To allocate label-specific attention weights to the most informative tokens in the sequence $\langle t_1, t_2, \cdots, t_n \rangle$ we take the following three steps.

[1] here $\mathsf{rank}_k(\boldsymbol{z}^{(\boldsymbol{i})})$ returns the k largest indices of $\boldsymbol{g}^{(\boldsymbol{i})}$ ranked in descending order.

[2] We used the pretrained LM from https://docs.fast.ai/text.models.awdlstm.html

First, the hidden features $\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_n$ of the sequence $\langle t_1, t_2, \cdots, t_n \rangle$ are concatenated to formulate the matrix $\boldsymbol{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_n]^T \in \mathbb{R}^{n \times s_e}$. To transform $\boldsymbol{H}$ into label-specific vectors, we compute label-specfic attention weights as:

$$\boldsymbol{A} = \mathsf{softmax}(\boldsymbol{H}\boldsymbol{U}^T), \boldsymbol{A} \in \mathbb{R}^{n \times |\mathcal{L}|} \quad (5)$$

where $\boldsymbol{U} \in \mathbb{R}^{|\mathcal{L}| \times s_e}$ is the label embedding matrix.The ith column in $\boldsymbol{A}$ represents the attention weights corresponding to the ith label in $\mathcal{L}$ for each of the n tokens. To ensure the bulk of the weight is placed on the most informative tokens, the softmax is applied at the column level. Here $\boldsymbol{A}$ denotes the *learned* attention weights.

Second, we perform attention planting by utilizing two types of attention weights: *static-planted* ($\boldsymbol{S}$) and *differentiable-planted* ($\boldsymbol{P}$). The static-planted attention ($\boldsymbol{S}$) remains constant and is based on mutual information gain, while the differentiable-planted attention ($\boldsymbol{P}$) comprises trainable parameters. These mechanisms enhance the model's ability to prioritize relevant tokens. We determine the static-planted attention as $\boldsymbol{S} = \left[\boldsymbol{g}^{(1)}, \boldsymbol{g}^{(2)}, \cdots, \boldsymbol{g}^{(|\mathcal{L}|)}\right] \in \mathbb{R}^{n \times |\mathcal{L}|}$, is comprised of individual vectors $\boldsymbol{g}^{(i)} = \left[g_1^{(i)}, g_2^{(i)}, \cdots, g_n^{(i)}\right]^T \in \mathbb{R}^n$. Each element $g_j^{(i)}$ of these vectors represents the relevance of token t_j with respect to label $l^{(i)}$, as precisely defined in section 3.1. We determine the differentiable-planted attention by computing feature vectors $\boldsymbol{x}_{\boldsymbol{j}}^{(\boldsymbol{i})} = \Psi\left(\boldsymbol{e}_{l^{(i)}}, \boldsymbol{e}_{t_j}\right)$ for each label-token pair $\left(l^{(i)}, t_j\right), i = 1, 2, \cdots, |\mathcal{L}|; j = 1, 2, \cdots, n$ as per equation 1. Then utilizing pretrained embeddings $\boldsymbol{e}_{l^{(i)}}$ and $\boldsymbol{e}_{t_j}$ from the L2R model in section 3.1, the pretrained L2R model computes scores $\boldsymbol{P} = \left[\boldsymbol{p}^{(1)}, \boldsymbol{p}^{(2)}, \cdots, \boldsymbol{p}^{(|\mathcal{L}|)}\right] \in \mathbb{R}^{n \times |\mathcal{L}|}$, where $\boldsymbol{p}^{(\boldsymbol{i})} = \left[f\left(\boldsymbol{x}_{\boldsymbol{1}}^{(\boldsymbol{i})}\right), \cdots, f\left(\boldsymbol{x}_{\boldsymbol{n}}^{(\boldsymbol{i})}\right)\right]^T \in \mathbb{R}^n$, and f is the ranking function from equation 2. In a departure from the standard attention approach, we introduce *inattention*, a pre-softmax thresholding technique that strategically elevates the significance of attention weights. By effectively zeroing out less relevant tokens, this method ensures maximal focus on pivotal tokens:

$$\boldsymbol{P} = \mathsf{softmax}(\mathsf{threshold}(\boldsymbol{P}, k)) \quad (6)$$

where both threshold (Appendix A.5) and softmax are applied at the column level.

Third, to compute the label-specific vectors, we perform linear combinations of the hidden features $\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_n$ using the attention weights from three sources: the *learned* attention weights in each column of $\boldsymbol{A}$, the *static-planted* attention weights in each column of $\boldsymbol{S}$, and the *differentiable-planted* attention weights in each column of $\boldsymbol{P}$. This is followed by element-wise matrix multiplication with a weight matrix $\boldsymbol{W} \in \mathbb{R}^{|\mathcal{L}| \times s_e}$:

$$\boldsymbol{V} = (\boldsymbol{A}^T\boldsymbol{H} + \boldsymbol{S}^T\boldsymbol{H} + \boldsymbol{P}^T\boldsymbol{H}) \odot \boldsymbol{W}, \boldsymbol{V} \in \mathbb{R}^{|\mathcal{L}| \times s_e} \quad (7)$$

The purpose of $\boldsymbol{W}$ is to boost attention. The ith row $\boldsymbol{v}_i$ of $\boldsymbol{V}$, can be thought of as the information regarding the ith label captured by *attention* from the token sequence $\langle t_1, t_2, \cdots, t_n \rangle$. Finally, this label-specific information is summed and added with a label-specific bias followed by sigmoid activation to produce predictions:

$$\hat{\boldsymbol{y}} = \mathsf{sigmoid}(\boldsymbol{1}\boldsymbol{V}^T + \boldsymbol{b}); \boldsymbol{1} \in \mathbb{R}^{s_e}; \boldsymbol{b}, \hat{\boldsymbol{y}} \in \mathbb{R}^{|\mathcal{L}|} \quad (8)$$

The training objective is to mimimize the binary cross-entropy loss between $\hat{\boldsymbol{y}}$ and the target $\boldsymbol{y}$ as:

$$\text{Loss}(\boldsymbol{y}, \hat{\boldsymbol{y}}, \theta) = -\sum_{i=1}^{|\mathcal{L}|} y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i),$$

where θ denotes all trainable model parameters.

A.5 Threshold

$$\mathsf{threshold}(\boldsymbol{p}^i, k) = \begin{cases} p_j, & \text{if } p_j > k^{th} \text{ largest } p \\ 0 & \text{otherwise.} \end{cases}$$

A.6 Implementation Details

Datasets

We compare PLANT to SOTA ICD coding models using the MIMIC-III (Johnson et al., 2016) and MIMIC-IV (Johnson et al., 2023) datasets, which include rich textual and structured records from ICU settings, primarily discharge summaries annotated with ICD-9 (MIMIC-III) and ICD-10 (MIMIC-IV) codes. MIMIC-III contains 52,722 discharge summaries with 8,929 unique ICD-9 codes, and MIMIC-IV includes 122,279 summaries with 7,942 ICD-10 codes. We follow established methodologies for patient ID-based splits and frequent code subsets. For few-shot learning, we evaluate PLANT on the MIMIC-III-rare50 dataset (Yang et al., 2022), which features 50 rare ICD codes, and the MIMIC-III-few dataset (Yang et al., 2023), a subset with

685 unique ICD-9 codes occurring between 1 and 5 times in the training set. We denote these datasets as MIMIC-III-full, MIMIC-III-top50, MIMIC-III-rare50, MIMIC-III-few, and MIMIC-IV-full (refer to Table 8 for statistics).

	MIMIC-III-full	MIMIC-IV-full
Number of documents	52,723	122,279
Number of patients	41,126	65,659
Number of unique codes	8,929	7,942
Codes pr. instance: Median (IQR)	$14(10-20)$	$14(9-20)$
Words pr. document: Median (IQR)	$1,375(965-1,900)$	$1,492(1,147-1,931)$
Documents: Train/val/test [%]	90.5/3.1/6.4	72.9/10.9/16.2

Table 8: Descriptive statistics for MIMIC-III-full and MIMIC-IV-full discharge summary training sets.

	EURLEX-4K	WIKI10-31K
Number of train documents	$15,449$	$14,146$
Number of test documents	$3,865$	$6,616$
Number of unique labels	$3,956$	$30,938$
Average number of labels per instance	5.30	18.64
Average number of instances per label	20.79	8.52

Table 9: Descriptive statistics for publicly available XMTC datasets EURLEX-4K and WIKI10-31K.

Preprocessing

Following prior research (Mullenbach et al., 2018; Xie et al., 2019; Li and Yu, 2020), we tokenize and lowercase all text while eliminating non-alphabetic tokens containing numbers or punctuation. A distinctive feature of our approach is the absence of preprocessed word embeddings. Instead, we fine-tune a pretrained AWD-LSTM model on our target dataset, allowing for parameter refinement, including word embeddings, and the generation of context-specific embeddings for new words in the dataset. While the concept of fine-tuning pretrained models is not new (Howard and Ruder, 2018), our innovation lies in its application to the XMTC domain. Contrary to previous practices (Li and Yu, 2020), we refrain from truncating text, as our experiments and findings align with those of Zhang et al. (2022), which demonstates substantial performance variation due to truncation. To handle longer texts, we employ our stateful decoder (refer to Section 3.2).

Implementation and Hyperparameters

We ensure robustness across diverse XMTC datasets by fine-tuning hyperparameters on the MIMIC-III-full and MIMIC-IV-full validation sets. Experiments are conducted on an NVIDIA QUADRO RTX 8000 GPU with 48 GB VRAM. We utilize the AWD-LSTM LM with an embedding size of 400, 3 LSTM layers with 1152 hidden activations, and the Adam Optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.99$, and weight decay of 0.01. During fine-tuning, we apply dropout rates and weight dropout, with a batch size of 384, BPTT of 80, 20 epochs, and a learning rate of $1e-5$. Classifier training also includes dropout rates and weight dropout, with a batch size of 16, BPTT of 72, and discriminative fine-tuning with gradual unfreezing over 115 epochs (on MIMIC-III-full), alongside scheduled weight decay and learning rate ranges.

A.7 Baselines for comparisons

ICD Baselines: This study compares PLANT with a range of ICD coding models developed over recent years, starting with CAML (Mullenbach et al., 2018), MSATT-KG (Xie et al., 2019), MUltiResCNN (Li and Yu, 2020), and HyperCore (Cao et al., 2020). Later models include LAAT/JointLAAT (Vu et al., 2021), ISD (Zhou et al., 2021), Effective-CAN (Liu et al., 2021), Hierarchical (Dai et al., 2022), and MSMN (Yuan et al., 2022). More recent approaches, such as DiscNet (Zhang et al., 2022), KEPTLongformer (Yang et al., 2022), PLM-ICD (Huang et al., 2022), AHDD (Zhang and Wang, 2024), CoRelation (Luo et al., 2024), Contrastive (Lu et al., 2023), KEMTL (Li et al., 2023), and MIMIC-IV-Benchmark (Nguyen et al., 2023), expand the scope. For few-shot learning, we also consider models like AGMHT (Song et al., 2021), RareCodes (Chen et al., 2023), GP (Yang et al., 2023), and KEPT (Yang et al., 2022).

XMTC Baselines: We also compare PLANT against XMTC models like: PECOS (Yu et al., 2022), ICXML (Zhu and Zamani, 2023), XRR (Xiong et al., 2023), RDE (Shi et al., 2024), MatchXML (Shi et al., 2024), DE (Gupta et al.), InceptionXML (Kharbanda et al., 2023), GANDALF (Kharbanda et al., 2024), CG (Chai et al., 2024).

A.8 Evaluation metrics

We focus on micro and macro F1 scores, AUC, and P@k to compare with prior ICD studies. Micro-averaging treats each (text, code) pair individually, while macro-averaging computes metrics per label, giving more weight to infrequent labels. Micro-R is the ratio of true positives to the sum of true positives and false negatives for each label, while Macro-R averages recall across all labels. P@k measures the proportion of the top k predicted labels that match the ground truth.

A.9 Additional Experiments

A.9.1 Performance of PLANT on MIMIC-III-top50

Model	AUC		F1		P@k		
	Macro	Micro	Macro	Micro	P@5	P@8	P@15
CAML/DR-CAML	88.4	91.6	57.6	63.3	61.8	-	-
MSATT-KG	91.4	93.6	63.8	68.4	64.4	-	-
MultiResCNN	89.9	92.8	60.6	67.0	64.1	-	-
HyperCore	89.5	92.9	60.9	66.3	63.2	-	-
LAAT/JointLAAT	92.5	94.6	66.6	71.6	67.5	54.7	35.7
ISD	93.5	94.9	67.9	71.7	68.2	-	-
Effective-CAN	92.0	94.5	66.8	71.7	66.4	-	-
MSMN	92.8	94.7	68.3	72.5	68.0	-	-
AHDD	92.8	94.7	68.5	72.8	67.8	-	-
CoRelation	93.3	95.1	69.3	73.1	68.3	55.6	-
MSMN (/w Contrastive)	-	-	69.1	72.5	68.3	-	-
KEMTL	94.8	95.5	69.5	72.9	**70.8**	-	-
PLANT (Ours)	**95.1***	**95.9***	**69.7**	**73.1***	**70.8**	**55.9***	**36.3***

Table 10: Results on the MIMIC-III-top50 test set.

A.9.2 Performance of PLANT on EURLEX-4K and WIKI10-31K

In evaluating the EurLex-4k and Wiki10-31k datasets, PLANT demonstrates superior performance compared to various SOTA XMTC models. On the EurLex-4k dataset, PLANT achieves a precision at 1 (P@1) of **89.92%**, outperforming the closest competitor, PECOS, by **1.51 percentage points**. For precision at 3 (P@3), it scores **80.42%**, surpassing XRR by **1.54 percentage points**, and achieves **69.38%** for precision at 5 (P@5), leading XRR by **0.86 percentage points**. In the Wiki10-31k dataset, PLANT scores **90.10%** for P@1, exceeding XRR by **0.56 percentage points**, and it achieves **86.67%** for P@3, outperforming XRR by **1.29 percentage points**. Although it scores **81.33%** for P@5, it remains highly competitive. The superior performance of PLANT in precision at 1, 3, and 5 (P@1, P@3, P@5) can be attributed to its innovative use of *L2R* activations as *planted attention weights*. These components enable PLANT to start from a highly informed state, allowing it to prioritize the most relevant tokens for each label from the very beginning of training.

- **For** P@1**:** PLANT 's static attention mechanism effectively focuses on the most critical token-to-label associations, making it highly precise in identifying the topmost relevant label for any given input. This leads to a more accurate prediction of the top label, resulting in its superior P@1 score.

- **For** P@3 **and** P@5**:** PLANT 's differentiable-planted attention weights further refine token relevance as the model trains, allowing it to capture more nuanced token-label interactions. This fine-tuned attention helps PLANT identify not only the most relevant label but also the next 3 or 5 most relevant labels with high accuracy, reflected in its high P@3 and P@5 scores.

Note: Since models like ICXML, DE, InceptionXML, GANDALF, and CG did not report results on the EURLEX-4K and WIKI10-31K datasets, we downloaded their code and trained their models from scratch using the hyperparameters specified in their respective papers.

A.9.3 How does PLANT handle overfitting?

The comparison with LAAT (Vu et al., 2021) was made to highlight PLANT 's ability to avoid overfitting, particularly in skewed label distribution scenarios where overfitting is a common issue. While LAAT may not represent the latest SOTA in ICD coding, we chose it for this comparison because of its similar architecture to PLANT, allowing us to isolate the impact of key components like the pretrained L2R activations. This setup enables us to demonstrate how PLANT remains stable, unlike LAAT, which begins to overfit after 40 epochs, as illustrated in Figure 5. PLANT maintains consistent performance through 60 epochs, preventing the divergence of train and test loss that occurs with LAAT, particularly in P@15. This comparison validates PLANT 's ability to remain robust even in extended training sessions. Our motivation for this comparison was to ensure that PLANT effectively mitigates overfitting and maintains generalization across epochs. We have taken significant care to incorporate industry-standard techniques for avoiding overfitting, such as regularization, dropout layers, early stopping, and learning rate scheduling. Regularization methods like L2 weight decay help prevent the model from becoming too complex by penalizing large weights, while dropout randomly ignores certain neurons during training to avoid co-adaptation of features. Additionally, early stopping halts training if performance on the validation set starts to degrade, ensuring the model does not overfit, and learning rate scheduling reduces the learning rate as training progresses to fine-tune model weights. In PLANT, we apply these techniques to ensure the model maintains a balance between underfitting and overfitting, as detailed in section A.6. This includes specific hyperparameter choices, dropout rates, and regularization strengths

that are optimized for our model's architecture, further enhancing its robustness. These strategies, widely adopted in the NLP industry, are crucial for maintaining model stability and generalizability in real-world applications, and their impact on PLANT 's performance is evident in our experiments.

A.10 Setup for Experiments and Ablation

We ran our model 5 times each with different random seeds for initialization and report mean scores. * indicates that the performance difference between PLANT and the next best is significant ($p < 0.01$, using the Approximate Randomization test). All scores in tables 2, 3, 4, 5, 6 and 10 are reported under the same experimental setup.

Except for the Gradual Unfreezing and Bidirectionality ablation, we selectively unfreeze the layers in decoder, keeping the encoder frozen—meaning no backpropagation was performed on their weights during training. This ensures that performance improvements are attributed directly to the decoder, our primary focus. Secondly, all reported performance metrics stem from the full test sets of both `MIMIC-III-full` and `MIMIC-IV-full` datasets. Thirdly, reported enhancements were statistically significant ($p < 0.01$, using the Approximate Randomization test).

A.11 Bidirectional Language Model Ablation

For the `MIMIC-III-full` and `MIMIC-IV-full` (Table 8), we pretrain both a forward and backward LM. We fine-tune an XMTC model for each LM independently and average the classifier predictions. On `MIMIC-III-full` P@15 increased from 60.61 to 61.67, and on `MIMIC-IV-full`, from 54.5 to 55.6.

A.12 Interpretability Case Study (Table 11)

We compare PLANT's interpretability against three baselines: MSATT-KG, CAML, and Text-CNN(Kim, 2014). While PLANT selects top 5 tokens per label based on attention values, baseline methods extract informative n-grams. MSATT-KG employs multi-scale and label-dependent attention, while CAML and Text-CNN use label-dependent attention and different phrase selection strategies. CAML uses a receptive field, and Text-CNN selects positions based on maximum channel values. In the interpretability case study, PLANT attends to tokens like 'intubation', 'fio2', and 'pc02'. 'fio2' represents Fraction of Inspired Oxygen, critical in determining oxygen concentration delivered to a patient. 'PCO2' signifies partial pressure of carbon dioxide, indicative of conditions like respiratory acidosis or alkalosis. In another example, informative tokens include 'gastrophageal', 'reflux', 'gerd', and 'prilosec', where 'gerd' denotes Gastroesophageal Reflux Disease and 'prilosec' is a proton pump inhibitor.

518.81: Acute respiratory failure
PLANT: ...patient had a gcs3t and required **intubation** ...**fio2** $\cdots$ temp po2 **pco2** ph ...
MSATT-KG: ... left **hemothorax**, **ETOH**, **depression**, stable discharge condition...
CAML: ...**small apical pneumothorax remained unchanged ... now tolerating a** ...
Text-CNN: ...**revealed a persistent left pleural effusion and due** to concern for loculated hemothorax...
530.81: Esophageal reflux
PLANT: ... **gastroesophageal reflux** ... home o2 **gerd** osteoporosis ... one puff hospital1 **prilosec** 20mg....
MSATT-KG: ... **tracheostomy & feeding gastrostomy** ... GERD, anxiety ...
CAML: ... **rib fx requiring tracheostomy & feeding gastrostomy,**, GERD, anxiety, cataracts...
Text-CNN: ... **right thoracotomy, decortication of lung, mobilization of liver** off of chest wall...

Table 11: Interpretability evaluation results for different models.

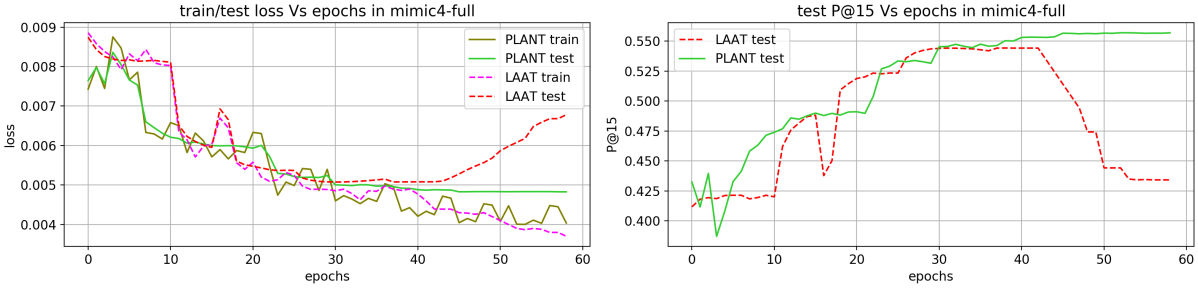


Figure 5: PLANT does not overfit on `MIMIC-IV-full`, LAAT (Vu et al., 2021) does.