TAILMIX: OVERCOMING THE LABEL SPARSITY FOR EXTREME MULTI-LABEL CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Extreme multi-label classification (XMC) aims at finding the most relevant labels from a huge label set at the industrial scale. The XMC problem inherently poses two challenges: *data scalability* and *label sparsity*. This work introduces a new augmentation method, namely *TailMix*, to address the label sparsity issue, *i.e.*, the long-tail labels in XMC have few positive instances. TailMix utilizes the context vector generated from the label attention layer in a label-wise manner instead of using the existing Mixup methods in a sample-wise manner. In this process, TailMix selectively chooses two context vectors and augments the most plausible positive instances to improve the accuracy for long-tail labels. Despite the simplicity of TailMix, extensive experimental results show that TailMix consistently improves the baseline models without TailMix and other Mixup-based methods on three benchmark datasets. Notably, TailMix is effective for improving the performance for long-tail labels on *PSP@k* and *PSN@k*, which are the common metrics that reflect the propensity of labels.

1 INTRODUCTION

Extreme multi-label classification (XMC) aims at finding the most relevant multiple labels from an enormously large label set. Essentially, XMC deals with a text classification problem at the industrial scale, where the number of labels can be the order of millions or more. It is closely related to various real-world applications, *e.g.*, finding a few relevant products from the catalog on the online store, categorizing an article into tens of thousands of topic categories in Wikipedia, and annotating few keywords for items to millions of advertisement keywords in E-commerce.

Despite the increasing popularity, the XMC problem faces two challenging issues: *data scalability* and *label sparsity*. Because the number of instances and labels are vast, it is difficult to handle expensive computational cost in the limited resource. Moreover, the enormous label space is aggravated by the label sparsity. As reported in (Chang et al., 2020), 98% of the labels in the Wiki-500k dataset are long-tail labels with less than 100 positive instances, indicating the extreme label imbalance problem.

Many existing studies have been proposed for addressing these issues. Existing work can be categorized into three directions: *sparse linear models*, *partition-based models*, and *embedding-based models*. Among them, the partition-based approach is commonly used in many existing models, *e.g.*, Parabel (Prabhu et al., 2018b), eXtremeText (Wydmuch et al., 2018), AttentionXML (You et al., 2019), XR-Linear (Yu et al., 2020) and X-Transformer (Chang et al., 2020), because it is effective for reducing the output space of labels. Specifically, it divides the label space into clusters in which the number of labels is much small. Based on the label clusters, each instance is first mapped to a few label clusters via a matching algorithm, and then the subset of labels within the cluster is only used for classification. The partitioning-based approach is computationally effective by reducing unnecessary costs for searching enormous labels. However, because it neglects to consider the imbalance problem for long-tail labels, it still shows low accuracies for long-tail labels.

In this work, we mainly tackle the label sparsity problem using a data augmentation technique. Specifically, we are inspired by Mixup (Zhang et al., 2018), which is known as a successful augmentation strategy to improve the generalization and robustness of the baseline model. Although several Mixup methods, *e.g.*, wordMix (Guo et al., 2019) and SSMix (Yoon et al., 2021), are proposed for the text classification problem, they have not yet been carefully considered the label sparsity of XMC.

In contrast, we propose a novel Mixup-based method for overcoming long-tailed sparsity, namely *TailMix*. Our TailMix generates new instances at the label attention layer for tail labels by carefully selecting context vectors based on the label sparsity and semantics. For that, we utilize a label proximity graph to find neighbors for the long-tail label and compare the similarity at the label attention context vector to find the most relevant neighbors. In this way, we can reduce the cost of finding semantically similar instances for long-tail labels and generate a context-aware representation for each label.

In summary, the key contributions of this work are as follows.

- We propose TailMix to address the label sparsity by selectively augmenting the instances with long-tailed labels.
- Unlike previous Mixup methods using a sample-wise combination, TailMix is designed to combine label-wise context vectors induced from the models with label-wise attention layer, which is beneficial for handling the selective augmentation for long-tail labels.
- We show that the representative partition-based model, AttentionXML (You et al., 2019) with different encoder architectures, *i.e.*, Bi-LSTM and RoBERTa, can be improved by equipping TailMix on three benchmark datasets.

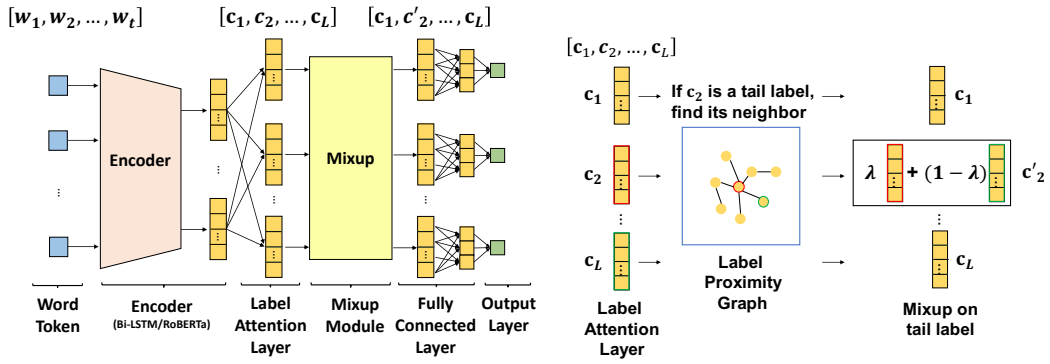
2 Related work

2.1 Extreme multi-label classification (XMC)

Sparse linear models. Because linear one-verse-reset (OVR) models learn a classifier for each label separately, the computational cost increases linearly by the number of labels. To resolve the scalability issue, DisMEC (Babbar & Schölkopf, 2017), ProXML (Babbar & Schölkopf, 2019), PDSparse (Yen et al., 2016), and PPDSparse (Yen et al., 2017) enforce sparsity using L1 penalty term to reduce the number of model parameters, DisMEC and PPDSparse utilize parallelism to achieve scalability. Besides, Parabel (Prabhu et al., 2018b) and SLICE (Jain et al., 2019) employ negative sampling to balance the number of positive and negative samples. In general, the OVR methods can be used as the building block for other XMC approaches.

Partition-based models. Because the number of output labels is enormous, the sparse linear models can be combined with different partitioning techniques. A hierarchical tree of labels is built by a recursive partitioning strategy to reduce training and prediction complexity. Parabel (Prabhu et al., 2018b) uses a balanced 2-means label tree using label features. FastXML (Prabhu & Varma, 2014) learns a binary classifier for each level by recursively partitioning a parent's instance feature space based on nDCG, and PfastreXML (Jain et al., 2016) is optimized by propensity scored nDCG with FastXML. SwiftXML (Prabhu et al., 2018a) splits a tree into two child nodes, where two hyperplanes for instances and label features are stored separately. CRAFTML (Siblini et al., 2018) builds a random forest exploiting tree randomization by projecting instance and label spaces using random projection matrices. Recently, AttentionXML (You et al., 2019) utilizes Bi-LSTM and attention mechanism and builds probabilistic label trees (PLTs) for feature vectors. To improve AttentionXML, X-Transformer (Chang et al., 2020) utilizes the transformer with three steps: 1) semantic label indexing, 2) deep neural matching, and 3) ensemble ranking.

Embedding-based models. These models project high-dimensional label space into low-dimensional space and use an approximate nearest neighbor (ANN) search to reduce the label space size. SLEEC (Bhatia et al., 2015) learns embedding vectors to capture non-linear label correlations by preserving pair-wise distance between two label vectors. During prediction, ANN search is used in low-dimensional embedding space. AnnexML (Tagami, 2017) constructs a k-nearest neighbor graph ($kNNG$) in the embedding space to retain a graph structure in label vector space. DEFRAG (Jalan & Kar, 2019) agglomerates sparse input features to reduce dimensionality to improve efficiency. ExMLDS (Gupta et al., 2019) learns embedding space using skip-gram negative sampling (SGNS), inspired by distributional semantics.



(a) Overall model architecture

(b) TailMix module

Figure 1: TailMix: (a) The operation is performed on the label attention layer. (b) The context vectors $\mathbf{c}_l$ corresponding to ground-truth labels are mixed with their neighbor context vector which is sampled based on co-occurrence and similarity.

2.2 Mixup as Data Augmentation

Mixup (Zhang et al., 2018) was originally introduced in computer vision, which interpolates two inputs and targets linearly. Specifically, the synthesized sample is generated from two samples $(\mathbf{x}_i, \mathbf{y}_i)$ and $(\mathbf{x}_j, \mathbf{y}_j)$ as follows:

$$\begin{aligned} \tilde{\mathbf{x}} &= \lambda \mathbf{x}_i + (1-\lambda)\mathbf{x}_j, \\ \tilde{\mathbf{y}} &= \lambda \mathbf{y}_i + (1-\lambda)\mathbf{y}_j. \end{aligned} \tag{1}$$

Here, $\mathbf{x}_i$ and $\mathbf{x}_j$ are input vectors, and $\mathbf{y}_i$ and $\mathbf{y}_j$ are their corresponding binary label vectors. λ is a mixing ratio sampled from the beta distribution parameterized with hyperparameter α.

For NLP tasks, Mixup is operated on the word embedding layer (Yoon et al., 2021; Guo et al., 2019) or any hidden layers (Sun et al., 2020; Guo et al., 2019). Let $\mathbf{E}_i = [\mathbf{e}_{i1}, \mathbf{e}_{i2}, ..., \mathbf{e}_{iT}]$ is the word embedding matrix, where T is a sequence length. Then, the Mixup is operated on the word embedding layer as follows:

$$\begin{aligned} \tilde{\mathbf{E}} &= \lambda \mathbf{E}_i + (1-\lambda)\mathbf{E}_j \\ &= \lambda[\mathbf{e}_{i1}, \mathbf{e}_{i2}, \ldots, \mathbf{e}_{iT}] + (1-\lambda)[\mathbf{e}_{j1}, \mathbf{e}_{j2}, ..., \mathbf{e}_{jT}] \end{aligned} \tag{2}$$

where two word-level embedding vectors $\mathbf{e}_{ik}$ and $\mathbf{e}_{jk}$ $(1 \leq k \leq T)$ are combined with the ratio λ and $(1-\lambda)$. That is, two embedding vectors are linearly interpolated at the same position. Although it can generate a new sample, the Mixup result does not reflect the semantic information for each target label, especially for tail labels.

3 TailMix: Proposed Method

3.1 Model Architecture

The overall model architecture using TailMix is based on AttentionXML (You et al., 2019) and LaRoBERTa (Zhang et al., 2020). AttentionXML is one of the representative partition-based models using a shallow and wide probabilistic label tree (PLT) and multi-label attention mechanism, and LaRoBERTa is an extension of RoBERTa (Liu et al., 2019) with a label-wise attention layer following after its encoder. The process of our model consists of three steps. We first obtain label-wise context vectors by passing through the encoder and the label-wise attention layer. Then, the context vectors are independently used for predicting the corresponding labels. In this process, we conduct our proposed augmentation method to synthesize new instances. Lastly, our model is trained with Mixup-based augmented instances.

Figure 1a depicts the model architecture of our model using TailMix. Specifically, the model consists of six layers: word representation layer, feature encoder layer, multi-label attention layer, Mixup

layer, fully connected layer, and output layer. Firstly, our model takes raw tokenized text with length T as input. Each token is represented by a dense vector using a word embedding, *i.e.*, $[\mathbf{h}_1, \ldots, \mathbf{h}_T]$. In this work, we utilize two encoders, Bi-LSTM (You et al., 2019) and RoBERTa (Liu et al., 2019), where Bi-LSTM is used as the encoder in AttentionXML and RoBERTa is recently used as a popular pre-trained language model.

An attention mechanism is used to represent different contexts for a given instance. Specifically, the label-wise attention layer computes a linear combination of hidden vectors from the encoder for each label. The output vector is capable of capturing the salient parts of the input text for each label. The context vector is obtained as follows:

$$\alpha_{ij} = \frac{\exp(\mathbf{h}_i^\top \mathbf{w}_j)}{\sum_{k=1}^{T} \exp(\mathbf{h}_k^\top \mathbf{w}_j)}, \quad \mathbf{c}_j = \sum_{k=1}^{T} \alpha_{kj} \mathbf{h}_k, \tag{3}$$

where $\mathbf{w}_j$ is the weight vector for the j-th label and α_{ij} is the normalized coefficient of $\mathbf{h}_i$. Finally, $\mathbf{c}_j$ is computed by a weighted sum of hidden vectors $[\mathbf{h}_1, \ldots, \mathbf{h}_T]$ with length T.

As the core part, we perform the Mixup layer, where context vectors and the corresponding labels are used as input and output. In the next section, we explain how to generate a new instance using context vectors in detail. Lastly, one or two fully connected layers and one output layer are used for label predictions. For each label, our model is trained by the binary cross-entropy loss function as used in the existing study (You et al., 2019).

3.2 Long-tail label mixup strategy

We present a novel data augmentation strategy, namely *TailMix*, which synthesizes a new training sample by applying Mixup for context vectors of long-tail labels. TailMix is different from the existing Mixup methods from two perspectives.

Firstly, we generate the Mixup-based samples in a label-wise manner instead of generating a new sample in a sample-wise manner. By using the multi-label attention layer, we can generate context vectors for each label. Since the context vector can be interpreted as different representations for a given sample, we can effectively utilize context vectors instead of using input samples to choose samples with similar semantics. For that, SSMix (Yoon et al., 2021) combines two samples by preserving the salient part of samples so that it can effectively capture the semantics of two input samples. However, it requires an expensive computational cost to check the saliency for each sample. In contrast, we reduce the computational cost by reusing the context vectors, which are the byproducts of model training.

Secondly, we selectively synthesize new samples using context vectors for long-tail labels while existing Mixup methods combine two random samples and the corresponding labels. For that, we introduce probabilistic sampling using inverse propensity scores (IPS) to eliminate the bias of top-head labels. Besides, we choose semantically similar two context vectors using similarity measures such as Euclidean distance. We empirically observe that Mixup between similar contexts is more effective than Mixup between randomly picked samples. Motivated the partition-based method for reducing the cost for calculating similarity, we introduce the label proximity graph to narrow down the candidates for Mixup by co-occurrence.

The overall process of TailMix is depicted in Figure 1b. It consists of two steps: *target context vector selection* and *similar semantic context vector selection*. Given a sample, we perform a convex combination of two context vectors and the corresponding labels as follows:

$$\begin{aligned} \tilde{\mathbf{c}} &= \lambda \mathbf{c}_i + (1-\lambda)\mathbf{c}_j, \quad \text{where } \; i, j \in [1, \ldots, L] \\ \tilde{\mathbf{y}} &= \lambda \mathbf{y}_i + (1-\lambda)\mathbf{y}_j, \quad \text{where } \; i, j \in [1, \ldots, L] \end{aligned} \tag{4}$$

Here, L is the number of labels in the dataset. $\mathbf{c}_i$ and $\mathbf{c}_j$ are context vectors for a given sample. Besides, $\mathbf{y}_i$ and $\mathbf{y}_j$ are the corresponding one-hot label vectors for $\mathbf{c}_i$ and $\mathbf{c}_j$. In this process, we selectively choose a target vector $\mathbf{c}_i$ and $\mathbf{c}_j$ for alleviating the label sparsity.

Target context vector selection. We first introduce the inverse propensity score (IPS) (Jain et al., 2016) for probabilistic sampling. Since the IPS is inversely proportional to raw frequency for labels,

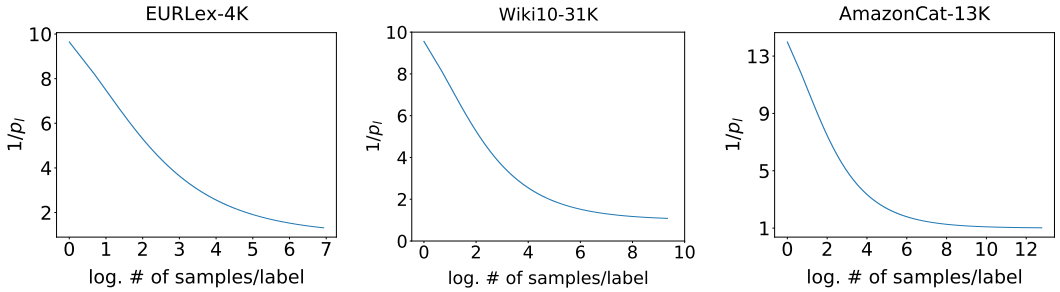


Figure 2: Distributions of the inverse propensity score $1/p_l$ on three benchmark datasets.

it is useful for measuring the bias of labels. Specifically, IPS, s_l for label l is computed as follows:

$$s_l = \left(\frac{1}{1 + Ce^{-A \log (N_l + B)}} \right)^{-1}, \quad \text{where } C = (\log N - 1)(B + 1)^A, \tag{5}$$

where A and B are empirically observed values from real-world datasets ($A = 0.55$, $B = 1.5$), N_l is the number of positive instances with label l, and N is the number of training samples.

We then convert IPS as the probability $prob_{il} = \text{softmax}(\{s_l\}_{l \in Y_i})$ using the softmax function where Y_i is a set of labels for i-th instance. Figure 2 shows the IPS of labels on three benchmark datasets, *i.e.*, EURLex-4K, Wiki10-31K, and AmazonCat-13K. It is found that long-tail labels have a higher probability than top-head labels. To eliminate the label bias, we sample the label in a probabilistic manner; if the label is sparser, it is more likely to be selected as the target context vector.

Similar semantic context vector selection. Once the target context vector is chosen, we then determine another context vector that shares high semantic similarity. Since the random selection hinders the improvement gains for long-tail labels, we synthesize a new sample with high similarity for the target context vector. For that, we choose another context vector that has the smallest Euclidean distance from the target context vector.

However, computing the similarity for all L labels in XMC is expensive since it has extremely large labels. For that, we utilize a *label proximity graph* to narrow down the candidates for Mixup. We use context vectors of the selected candidates and then pick samples based on the semantic similarity on the target context vector. A proximity label graph is built with L nodes and the edges between them. Each node represents a distinct label within a dataset, and each edge represents the co-occurrence of labels within an instance. For each sample x, it is assigned a label $y_n \in \{0, 1\}$. If y_i and y_j are both assigned 1 within a sample, the edge e_{ij} between node i and j becomes 1, otherwise 0. That is, the nodes, *i.e.*, labels, are considered as the neighbors if two labels are shared by some instances. The adjacency matrix $\mathbf{A} \in \mathbb{R}^{L \times L}$ represents the proximity label graph.

$$\mathbf{A} = \text{Binary}(\mathbf{Y}^{\mathrm{T}} \times \mathbf{Y}), \tag{6}$$

where $\mathbf{Y} \in \mathbb{R}^{N \times L}$ is target label matrix, N and L are number of samples and labels respectively. $\text{Binary}(\cdot)$ is the operation to binarize the given input.

Among the labels that are adjacent to the target, we then calculate Euclidean distance on the context vector and select the M closest samples. Those M samples are finally used to create M synthetic samples with Mixup by interpolating between context vectors $\mathbf{c}$ and corresponding labels y. As a result, we can generate new samples for tail labels that are semantically similar to the original tail label, providing different but similar data samples for the models and helping the model predict tail labels better.dataset

4 Experiments

4.1 Experimental setup

Datasets. We use three XMC benchmark datasets: Eurlex-4K (Mencía & Fürnkranz, 2008), Wiki10-31K (Zubiaga, 2012), and AmazonCat-13K (McAuley & Leskovec, 2013), where the suffixes refer

Dataset	N_{train}	N_{test}	L	L_{lps}	L_{spl}	$\|D\|$
EURLex-4K	15,449	3,865	3,956	5.3	20.79	1248.58
Wiki10-31K	14,146	6,616	30,938	18.64	8.52	2484.3
AmazonCat-13K	1,186,239	306,782	13,330	5.04	448.57	246.61

Table 1: Detailed data statistics on three benchmark datasets. N_{train} and N_{test} refer to the number of examples in the training and test sets. L is the number of distinct labels within the datasets. L_{lps} and L_{spl} refers to the average number of labels per sample and the average number of samples per label respectively. $|D|$ refers to the average number of tokens within the datasets.

to the number of labels for each dataset. For data pre-processing, we follow the instructions in the public source code of AttentionXML (You et al., 2019). Table 1 reports detailed statistics of each dataset.

Baseline models. We adopt two baseline models in our experiments: AttentionXML (You et al., 2019) and LaRoBERTa (Zhang et al., 2020). AttentionXML is a Bi-LSTM based model, followed by a label-wise attention layer, and its embedding layer use pre-trained GloVe (Pennington et al., 2014) vectors. Likewise, LaRoBERTa is a transformer-based model with label-wise attention. We use pre-trained RoBERTa (Liu et al., 2019) from HuggingFace (Wolf et al., 2019). We train AttentionXML and LaRoBERTa using TailMix and measure the performance using an ensemble on the predictions of models using TailMix.

Evaluation metrics. We choose $P@k$ (precision at k) as our evaluation metric, which is widely used in XMC task. For evaluating tail label performance, we use $PSP@k$ (propensity-scored precision) and $PSN@k$ (propensity-scored nDCG@k).

$$P@k = \frac{1}{k} \sum_{l \in rank_k(\hat{\mathbf{y}})} \mathbf{y}_l, PSP@k = \frac{1}{k} \sum_{l \in rank_k(\hat{\mathbf{y}})} \frac{\mathbf{y}_l}{p_l} \tag{7}$$

$$PSDCG@k = \sum_{1 \in rank_k(\hat{y})} \frac{\mathbf{y}_l}{p_l \log(l+1)}, PSN@k = \frac{PSDCG@k}{\sum_{l=1}^{k} \frac{1}{\log(l+1)}}, \tag{8}$$

where p_l is propensity score shown in Eq. 5, $\mathbf{y} \in \{0,1\}^L$ is the ground-truth binary vector, and $rank_k(\hat{\mathbf{y}})$ is a function which returns the k largest index of prediction $\hat{\mathbf{y}}$ ranked in descending order. The performance on tail labels can be examined using propensity-scored metrics.

Comparing XMC methods. We compare our proposed method with other competitive XMC methods, including embedding-based (AnnexML, SLEEC), partition-based (Parabel, Bonsai), sparse linear (XT) and deep neural networks (XML-CNN, AttentionXML, LaRoBERTa).

Comparing Mixup variants. We also compare our proposed method to other Mixup methods. WordMix (Guo et al., 2019) performs interpolation on random word embeddings without considering any word orders or their semantics. HiddenMix (Verma et al., 2019) performs Mixup on a randomly picked hidden layer from the model. TMix (Chen et al., 2020) considers the difference in interpretation power of BERT layers and evaluates the effect of choosing specific layers for Mixup. SSMix (Yoon et al., 2021) considers the order of word sequences on the word embedding layer and synthesizes the sample using saliency to keep the token information related to the target labels.

Hyperparameter. For all datasets, we follow the hyperparameter settings of AttentionXML (You et al., 2019). Specifically, we set the drop rate of the embedding layer as 0.5, the size of the Bi-LSTM layer as 256, and the size of two fully connected layers as 256 and 128. Then, the model is optimized by AdamW (Loshchilov & Hutter, 2019) with the learning rate of 1e-3 with weight decay of 1e-2. In LaRoBERTa (Zhang et al., 2020), we follow the same parameters for fully connected layers, and the learning rate is set to 1e-5 with a weight decay of 1e-2. For TailMix, we set α varying from 0.2 to 0.6 and report the best model for each dataset.

4.2 Experimental results

Table 2 compares the proposed TailMix with other competitive methods by $P@k$, $PSP@k$, and $PSN@k$ over three benchmark datasets. Following previous works on XMC, we focus on top

EURLex-4K									
Method	P			PSP			PSN		
	@1	@3	@5	@1	@3	@5	@1	@3	@5
AnnexML*	79.26	64.30	52.33	34.25	39.83	42.76	34.25	38.35	40.30
SLEEC*	63.40	50.35	41.28	24.10	27.20	29.09	24.10	26.37	27.62
XT*	78.97	65.64	54.44	33.52	40.35	44.02	33.52	38.50	41.09
Parabel*	82.25	68.71	57.53	36.44	44.08	48.46	36.44	41.99	44.91
Bonsai*	82.96	69.76	58.31	37.08	45.13	49.57	37.08	42.94	46.10
XML-CNN†	75.32	60.14	49.21	32.41	36.95	39.45	-	-	-
AttentionXML	83.88	71.77	60.50	42.88	49.90	52.78	42.88	47.97	49.97
LaRoBERTa	83.18	70.42	58.50	38.52	46.76	50.61	38.52	44.51	47.13
AttentionXML+TailMix	84.11	71.93	60.48	**43.64**	51.11	53.69	**43.64**	49.11	50.94
AttentionXML+TailMix ‡	**85.80**	**73.70**	**61.99**	43.63	**51.65**	**54.59**	43.63	**49.48**	**51.53**
LaRoBERTa+TailMix	84.14	71.54	59.27	42.69	49.21	51.33	42.69	47.46	49.10
LaRoBERTa+TailMix ‡	85.02	72.40	60.18	42.41	49.41	52.38	42.41	47.50	49.55

Wiki10-31K									
Method	P			PSP			PSN		
	@1	@3	@5	@1	@3	@5	@1	@3	@5
AnnexML*	**86.49**	74.27	64.20	11.90	12.76	13.58	11.90	12.53	13.10
SLEEC*	85.88	72.98	62.70	11.14	11.86	12.40	11.14	11.68	12.06
XT*	86.15	75.18	65.41	11.87	13.08	13.89	11.87	12.78	13.36
Parabel*	84.17	72.46	63.37	11.68	12.73	13.69	11.68	12.47	13.14
Bonsai*	84.69	73.69	64.39	11.78	13.27	14.28	11.78	12.89	13.61
XML-CNN†	81.42	66.23	56.11	9.39	10.00	10.20	-	-	-
AttentionXML	84.95	77.13	67.53	13.00	15.35	16.45	13.00	14.77	15.59
LaRoBERTa	80.00	67.18	57.38	10.77	11.99	12.15	10.77	11.71	11.87
AttentionXML+TailMix	84.05	76.87	67.91	**13.47**	**17.09**	**18.44**	**13.47**	**16.22**	**17.26**
AttentionXML+TailMix ‡	85.17	**78.12**	**68.66**	13.00	16.03	17.43	13.00	15.29	16.33
LaRoBERTa+TailMix	82.91	72.82	62.73	11.35	13.39	13.81	11.35	12.91	13.28
LaRoBERTa+TailMix ‡	81.91	71.48	62.06	10.29	12.45	12.98	10.29	11.97	12.41

AmazonCat-13K									
Method	P			PSP			PSN		
	@1	@3	@5	@1	@3	@5	@1	@3	@5
AnnexML*	93.54	78.37	63.30	49.04	61.13	69.64	49.04	58.83	65.47
SLEEC*	90.53	76.33	61.52	46.75	58.46	65.96	46.75	55.19	60.08
XT*	92.59	78.24	63.58	49.61	62.22	70.24	49.61	59.71	66.04
Parabel*	93.03	79.16	64.52	50.93	64.00	72.08	50.93	60.37	65.68
Bonsai*	92.98	79.13	64.46	51.30	64.60	72.48	-	-	-
XML-CNN†	93.26	77.06	61.40	52.42	62.83	67.10	-	-	-
AttentionXML	95.22	81.10	65.86	51.29	65.21	72.88	51.29	61.37	66.47
LaRoBERTa	91.11	75.05	60.50	52.40	61.57	65.59	52.40	59.09	61.81
AttentionXML+TailMix	95.23	80.95	65.70	51.48	65.44	72.62	51.48	61.62	66.42
AttentionXML+TailMix ‡	**95.53**	**81.56**	**66.28**	51.34	**65.63**	**73.36**	51.34	**61.70**	**66.85**
LaRoBERTa+TailMix	91.44	75.62	60.86	**54.98**	63.60	67.12	**54.98**	61.28	63.68
LaRoBERTa+TailMix ‡	91.61	75.67	60.89	53.66	62.86	66.87	53.66	60.37	63.09

Table 2: Performance comparison TailMix between other competitive XMC methods. *, † indicate the results reported in XMC repository (Bhatia et al., 2016) and its publication, respectively. ‡ is marked as ensemble of baseline and a model trained with TailMix.

predictions by showing P (precision), PSP (propensity-scored precision) and PSN (propensity-scored NDCG) by varying k at 1, 3, 5.

For PSP and PSN, our method (AttentionXML + TailMix and LaRoBERTa + TailMix) outperforms all other methods at all k on EURLex-4K and Wiki10-31K datasets. On these datasets, applying TailMix to AttentionXML or RoBERTa always improves the performance of tail labels. However, it does not show consistent results in AmazonCat-13K. We conjecture that this is because

EURLex-4K

	Method	P @1	P @3	P @5	PSP @1	PSP @3	PSP @5	PSN @1	PSN @3	PSN @5	Time (h)
AttentionXML	wordMix	84.68	71.49	59.59	43.02	48.95	51.62	43.02	47.32	49.14	1.91
	hiddenMix	**85.25**	72.15	60.30	**44.41**	49.83	52.43	**44.41**	48.37	50.11	1.40
	SSMix	85.07	**73.01**	**60.88**	43.99	51.09	53.56	43.99	**49.12**	50.88	4.58
	TailMix	84.11	71.93	60.48	43.64	**51.11**	**53.69**	43.64	49.11	**50.94**	1.59
LaRoBERTa	wordMix	82.61	70.89	58.94	37.79	47.07	50.98	37.79	44.52	47.22	19.13
	TMix	80.80	69.40	57.60	36.78	45.52	49.12	36.78	43.14	45.64	16.67
	SSMix	82.69	70.15	58.64	38.04	45.88	50.06	38.04	43.73	46.54	22.64
	TailMix	**84.14**	**71.54**	**59.27**	**42.69**	**49.21**	**51.53**	**42.69**	**47.46**	**49.10**	1.80

Wiki10-31K

	Method	P @1	P @3	P @5	PSP @1	PSP @3	PSP @5	PSN @1	PSN @3	PSN @5	Time (h)
AttentionXML	wordMix	84.84	76.90	67.76	**14.46**	16.32	17.35	**14.46**	15.86	16.62	1.02
	hiddenMix	**85.29**	77.04	67.79	14.04	15.96	17.02	14.04	15.49	16.26	2.35
	SSMix	85.14	**77.31**	67.76	13.30	15.74	16.91	13.30	15.14	16.01	7.86
	TailMix	84.05	76.87	**67.91**	13.47	**17.09**	**18.44**	13.47	**16.22**	**17.26**	1.98
LaRoBERTa	wordMix	81.50	62.56	53.19	9.54	9.70	9.30	9.54	9.28	9.33	17.22
	TMix	80.76	70.09	60.50	9.08	11.22	12.02	9.08	10.73	11.35	17.59
	SSMix	80.77	68.88	57.91	9.09	10.73	10.86	9.09	10.39	10.54	20.80
	TailMix	**82.91**	**72.82**	**62.73**	**11.35**	**13.39**	**13.81**	**11.35**	**12.91**	**13.28**	2.52

AmazonCat-13K

	Method	P @1	P @3	P @5	PSP @1	PSP @3	PSP @5	PSN @1	PSN @3	PSN @5	Time (h)
AttentionXML	wordMix	94.48	79.81	64.59	51.31	64.38	71.00	51.31	60.70	65.22	13.64
	hiddenMix	95.00	80.93	65.62	51.19	63.97	71.58	51.19	60.45	65.46	30.16
	SSMix	**95.41**	**81.51**	**66.34**	50.96	65.38	**73.72**	50.96	61.39	**66.91**	100.42
	TailMix	95.23	80.95	65.70	**51.48**	**65.44**	72.62	**51.48**	**61.62**	66.42	49.09
LaRoBERTa	wordMix	91.03	75.17	60.43	53.91	62.72	66.35	53.91	60.35	62.82	80.75
	TMix	91.32	75.30	60.81	**55.11**	**65.65**	**67.17**	**55.11**	**61.35**	**63.76**	166.37
	SSMix	91.23	75.53	60.83	54.44	63.34	67.05	54.44	60.94	63.47	270.28
	TailMix	**91.44**	**75.62**	**60.86**	54.98	63.60	67.12	54.98	61.28	63.68	117.48

Table 3: Performance comparison on EURLex-4K, Wiki10-31K, and AmazonCat-13K dataset with different Mixup methods. Each Mixup methods are applied on two different model architectures: AttentionXML (You et al., 2019) and LaRoBERTa (Zhang et al., 2020).

AmazonCat-13K dataset has a high L_{spl}, average number of samples per label. The average number of samples per label of AmazonCat-13K is 448.57 (20.79 for Eurlex-4K and 8.52 for Wiki10-31K), and the gains from applying TailMix could have been reduced since high L_{spl} could mean that even tail labels have enough number of samples. Even for P, except for P@1 on Wiki10-31K, our suggested method outperforms all other methods. Although we focus on augmenting the tail labels, not only it boosts the performance on tail labels but also boosts performance on head labels. Specifically, when compared with original AttentionXML, our suggested method shows improvement of 3.42%, 3.12% for PSP@5 and PSN@5 on EURLex-4K and shows 12.10% and 10.71% improvement on Wiki10-31K.

Table 3 shows the performance results of various Mixup methods applied on AttentionXML and LaRoBERTa. We compare our method with wordMix (Guo et al., 2019), hiddenMix (Verma et al., 2019), TMix (Chen et al., 2020), and SSMix (Yoon et al., 2021). Note that the outcomes for each model are based on a single model trained using only different Mixup methods. The results on Table 3 exhibit similar trends from Table 2. TailMix shows the best or the second-best performance

EURLex-4K											
Module		P			PSP			PSN			Time
Sim	LPG	@1	@3	@5	@1	@3	@5	@1	@3	@5	(h)
$-$	$-$	**85.51**	72.54	60.41	42.26	49.37	52.26	42.26	47.45	49.45	0.92
$-$	$+$	85.23	**72.71**	**60.65**	41.79	49.53	52.68	41.79	47.40	49.58	1.38
$+$	$-$	84.71	72.04	60.50	41.11	48.76	52.49	41.11	46.70	49.24	3.03
TailMix (Ours)		84.11	71.93	60.48	**43.64**	**51.11**	**53.69**	**43.64**	**49.11**	**50.94**	1.59

Wiki10-31K											
Module		P			PSP			PSN			Time
Sim	LPG	@1	@3	@5	@1	@3	@5	@1	@3	@5	(h)
$-$	$-$	84.16	75.46	65.98	12.04	14.97	16.17	12.04	14.26	15.18	0.91
$-$	$+$	84.01	75.83	66.34	12.41	15.28	16.51	12.41	14.49	15.51	1.01
$+$	$-$	**84.58**	76.31	66.96	12.62	15.83	17.06	12.62	15.06	16.00	3.06
TailMix (Ours)		84.05	**76.87**	**67.91**	**13.47**	**17.09**	**18.44**	**13.47**	**16.22**	**17.26**	1.98

Table 4: Results of different sample selection strategies running on AttentionXML. Sim, LPG refer to the "similarity" and "label proximity graph" module, respectively. "$-$" means that the module is omitted, and "$+$" refers that the following module was used in the model.

among various Mixup methods in most experiments on PSP and PSN, and improvements were both found on EURLex-4K and Wiki10-31K with LaRoBERTa.

The first two methods for each AttentionXML and LaRoBERTa (wordMix (Guo et al., 2019), hiddenMix (Verma et al., 2019), TMix (Chen et al., 2020)) are methods which does not consider the semantics of input while conducting Mixup, therefore requires short training time. Both SSMix and TailMix consider the semantics for Mixup. However, TailMix shows comparable or shorter training time when compared with the methods that do not use semantic information for Mixup. For EURLex-4K and Wiki10-31K, TailMix is 10.63 and 6.83 times faster than wordMix.

To examine the effectiveness of individual modules in our suggested method, we also conduct an ablation study on the Eurlex-4K and Wiki10-31K datasets. We compare the results on TailMix with $(+)$ and without $(-)$ certain modules: similarity-based selection (Sim) and co-occurrence-based selection (LPG). Here, we used the AttentionXML model and did not employ the ensemble method on evaluation.

Table 4 shows that both $PSP@k$ and $PSN@k$ decrease when any one of the modules are not used. When both similarity and co-occurrence are not used, and all labels are randomly chosen, it has the shortest training time but shows worse performance when compared to our suggested method. When no similarity measure is used but randomly chosen from co-occurring labels, we observe -10.47% and -10.13% drop in performance with $PSP@5$ and $PSN@5$ on Wiki10-31K, and -1.88% and -2.67% drop on EURLex-4K dataset. On $(+Sim, -LPG)$, the performance on PSP and PSN decreases, and the training time also substantially increases when compared to the case when both modules are used since it needs to calculate similarity on all L labels. We can observe that narrowing down the candidates for Mixup by co-occurrence not only reduces the training time but also gives performance gain to the model.

5 Conclusion

In this work, we have proposed a novel Mixup method for mitigating the long-tail label problem in XMC, namely TailMix. Unlike previous work, our method operates the interpolation in a label-wise manner. Besides, we combine two context vectors to synthesize a new sample in which the inverse propensity scores of labels and the label proximity graph are used for selectively choosing two context vectors. Experimental results show that two baseline models using TailMix consistently improve long-tail labels' performance over three benchmark datasets.

REFERENCES

Rohit Babbar and Bernhard Schölkopf. DiSMEC: Distributed sparse machines for extreme multi-label classification. In *WSDM*, 2017.

Rohit Babbar and Bernhard Schölkopf. Data scarcity, robustness and extreme multi-label classification. *Mach. Learn.*, 108(8-9):1329–1351, 2019.

K. Bhatia, K. Dahiya, H. Jain, P. Kar, A. Mittal, Y. Prabhu, and M. Varma. The extreme classification repository: Multi-label datasets and code, 2016.

Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. Sparse local embeddings for extreme multi-label classification. In *NeurIPS*, 2015.

Wei-Cheng Chang, Hsiang-Fu Yu, Kai Zhong, Yiming Yang, and Inderjit S. Dhillon. Taming pretrained transformers for extreme multi-label text classification. In *SIGKDD*, 2020.

Jiaao Chen, Zichao Yang, and Diyi Yang. MixText: Linguistically-informed interpolation of hidden space for semi-supervised text classification. In *ACL*, 2020.

Hongyu Guo, Yongyi Mao, and Richong Zhang. Augmenting data with mixup for sentence classification: An empirical study. *arXiv preprint arXiv:1905.08941*, 2019.

Vivek Gupta, Rahul Wadbude, Nagarajan Natarajan, Harish Karnick, Prateek Jain, and Piyush Rai. Distributional semantics meets multi-label learning. In *AAAI*, 2019.

Himanshu Jain, Yashoteja Prabhu, and Manik Varma. Extreme multi-label loss functions for recommendation, tagging, ranking amp; other missing label applications. In *SIGKDD*, 2016.

Himanshu Jain, Venkatesh Balasubramanian, Bhanu Chunduri, and Manik Varma. Slice: Scalable linear extreme classifiers trained on 100 million labels for related searches. In *WSDM*, 2019.

Ankit Jalan and Purushottam Kar. Accelerating extreme classification via adaptive feature agglomeration. In *IJCAI*, pp. 2600–2606. ijcai.org, 2019.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

Julian J. McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*, 2013.

Eneldo Loza Mencía and Johannes Fürnkranz. Efficient pairwise multilabel classification for large-scale problems in the legal domain. In *ECML/PKDD*, 2008.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global vectors for word representation. In *EMNLP*, 2014.

Yashoteja Prabhu and Manik Varma. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *SIGKDD*, 2014.

Yashoteja Prabhu, Shilpa Gopinath, Kunal Dahiya, Anil Kag, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. Extreme multi-label learning with label features for warm-start tagging, ranking recommendation. In *WSDM*, 2018a.

Yashoteja Prabhu, Anil Kag, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *WWW*, 2018b.

Wissam Siblini, Frank Meyer, and Pascale Kuntz. CRAFTML, an efficient clustering-based random forest for extreme multi-label learning. In *ICML*, 2018.

Lichao Sun, Congying Xia, Wenpeng Yin, Tingting Liang, Philip S. Yu, and Lifang He. Mixup-transformer: Dynamic data augmentation for nlp tasks. In *COLING*, 2020.

Yukihiro Tagami. Annexml: Approximate nearest neighbor search for extreme multi-label classification. In *SIGKDD*, 2017.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *ICML*, 2019.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Marek Wydmuch, Kalina Jasinska, Mikhail Kuznetsov, Róbert Busa-Fekete, and Krzysztof Dembczynski. A no-regret generalization of hierarchical softmax to extreme multi-label classification. In *NeurIPS*, 2018.

Ian En-Hsu Yen, Xiangru Huang, Pradeep Ravikumar, Kai Zhong, and Inderjit S. Dhillon. Pd-sparse : A primal and dual sparse approach to extreme multiclass and multilabel classification. In *ICML*, 2016.

Ian En-Hsu Yen, Xiangru Huang, Wei Dai, Pradeep Ravikumar, Inderjit S. Dhillon, and Eric P. Xing. PPDsparse: A parallel primal-dual sparse method for extreme classification. In *SIGKDD*, 2017.

Soyoung Yoon, Gyuwan Kim, and Kyumin Park. Ssmix: Saliency-based span mixup for text classification. In *ACL/IJCNLP*, 2021.

Ronghui You, Zihan Zhang, Ziye Wang, Suyang Dai, Hiroshi Mamitsuka, and Shanfeng Zhu. AttentionXML: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. In *NeurIPS*, 2019.

Hsiang-Fu Yu, Kai Zhong, and Inderjit S. Dhillon. PECOS: prediction for enormous and correlated output spaces. *arXiv preprint arXiv:2010.05878*, 2020.

Danqing Zhang, Tao Li, Haiyang Zhang, and Bing Yin. On data augmentation for extreme multi-label classification. *arXiv preprint arXiv:2009.10778*, 2020.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

Arkaitz Zubiaga. Enhancing navigation on wikipedia with social tags. *arXiv preprint arXiv:1202.5469*, 2012.

A ALGORITHM

Algorithm 1 presents the procedure of training model with TailMix.

Algorithm 1: Training of TailMix

Input : Training batch $(\mathbf{X^B}, \mathbf{Y^B})$; Maximum # of targets M; Label Proximity Graph $\mathbf{A}$; Context encoder model $f(\cdot;\theta)$; Classifier (FC layers) $g(\cdot;\omega)$

$\mathbf{C} \leftarrow f(\mathbf{X^B};\theta)$; // Get context vectors for each label

for $i \leftarrow 1$ **to** $|\mathbf{X^B}|$ **do**

  $L_i \leftarrow Nonzero(\mathbf{Y}_i^{\mathbf{B}})$; // Get ground-truth labels

  $\overline{L_i} \leftarrow Choose(L_i, \{s_l\}_{l \in Y_i}, \min(|L_i|, M))$; // Select tail labels using IPS

  for $l \in \overline{L_i}$ **do**

    $T \leftarrow Nonzero(\mathbf{A}_l)$; // Get neighbor label index set

    $dist \leftarrow Euclidean(\mathbf{C}_{il}, \{\mathbf{C}_{ij}\}_{j \in T})$; // Calculate distance on neighbors

    $t \leftarrow Argmin(dist)$; // Get the most similar context vector

    $\lambda_{il} \leftarrow Beta(\alpha, \alpha)$; // Sample the mixing ratio

    $\mathbf{C}_{il} \leftarrow \lambda_{il}\mathbf{C}_{il} + (1 - \lambda_{il})\mathbf{C}_{it}$; // Perform Mixup on context vectors

    $\mathbf{Y}_{il}^B \leftarrow \lambda_{il}$; // Perform Mixup on the first label

    $\mathbf{Y}_{it}^B \leftarrow 1 - \lambda_{il}$; // Perform Mixup on the second label

$\hat{\mathbf{Y}}^B \leftarrow g(\mathbf{C};\omega)$; // Predict using the synthesized context vector

$loss \leftarrow BCE(\mathbf{Y^B}, \hat{\mathbf{Y}}^B)$; // Calculate BCE Loss

$AdamW(loss;\theta,\omega)$; // Update the model