InceptionXML: A Lightweight Framework with Synchronized Negative Sampling for Short Text Extreme Classification

Anonymous ACL submission

Abstract

Automatic annotation of short-text data to a large number of target labels, referred to as Short Text Extreme Classification, has found numerous applications including prediction of related searches and product recommendation tasks. In this paper, we propose a convolutional architecture INCEPTIONXML which is lightweight, yet powerful, and robust to the inherent lack of word-order in short-text queries encountered in search and recommendation tasks. We demonstrate the efficacy of applying convolutions by recasting the operation along the embedding dimension instead of the word dimension as applied in conventional CNNs for text classification. Towards scaling our model to datasets with millions of labels, we also propose INCEPTIONXML+ framework which improves upon the shortcomings of the recently proposed dynamic hard-negative mining technique for label shortlisting by synchronizing the label-shortlister and extreme classifier. INCEPTIONXML+ not only reduces the inference time to half but is also an order of magnitude smaller than previous state-of-the-art ASTEC in terms of model size. Through our proposed models, we outperform all existing approaches on popular benchmark datasets.

1 Introduction

Extreme Multi-label Classification (**XML**) involves classifying instances into a set of most relevant labels from an extremely large (on the order of millions) set of all possible labels. For scenarios when the input instances are short text queries, many successful applications of the XML framework have been found in ranking and recommendation tasks such as prediction of *Related Search* on search engines (Jain et al., 2019), suggestion of query phrases corresponding to short textual description of products on e-stores (Chang et al., 2020) and product-to-product recommendation (Dahiya et al., 2021a; Chang et al., 2021).

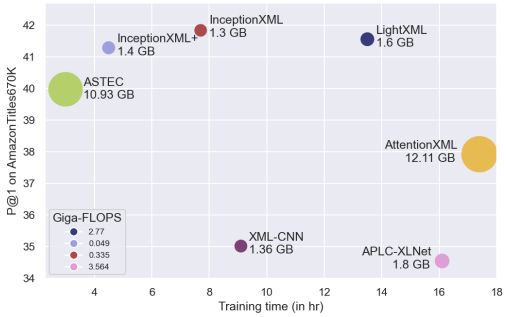

Figure 1: INCEPTIONXML(+) *(Ours)* hits the sweet spot in terms of performance on the P@1 metric, training time, model size and inference times.

Challenges in Short-Text XML: (i) Unlike regular documents, most short text queries are sparse and contain very few words and (ii) are typically plagued with noise and non-standard phrases which do not always observe the syntax of a written language. For instance, queries "*best wireless headphones 2022*" and "*2022 best headphones wireless*" should invoke similar search results on an e-commerce website (Tayal et al., 2020). Short text input data in search and recommendation, therefore, give rise to a significant amount of ambiguity (Wang and Wang, 2016). Furthermore, (iii) a large fraction of classes are tail labels, which are paired with a handful of positive samples (Jain et al., 2016). Taken together, the above characteristics, pose a challenge in learning rich feature representations for the task at hand.

Need of lightweight architectures in Short-Text XML: While large pre-trained language models are the default choice for most down-stream language tasks, we argue that (i) using such computationally intensive architectures for modeling short-text queries is rather overcompensating for the XML task at hand. Further, (ii) the real-world use cases of short-text extreme classification require very fast inference times. The deployment of large pre-trained

language models such as BERT, RoBERTa and XLNet as in LightXML (Jiang et al., 2021), APLC-XLNet (Ye et al., 2020) and X-Transformer (Chang et al., 2020) adds heavily to the already existing compute costs in XML tasks leading to slower training and inference times (Table: 2). Finally, (iii) extremely large number of possible labels leads to memory bottlenecks in XML tasks. As a result, these transformer-based methods become unscalable to millions of labels (Table: 1) while staying within reasonable hardware constraints.

InceptionXML: To address the above, we (i) develop INCEPTIONXML, a lightweight CNN-based encoder, which goes against the traditional paradigm (Kim, 2014; Liu et al., 2017) of convolving over the words dimension in favor of the embedding dimension, (ii) propose an embedding-enhancement module for learning a word-order agnostic representation, making our approach more robust to lack of structure in short-text queries, (iii) develop a very fast and computationally inexpensive INCEPTIONXML+ framework, which synchronizes the label-shortlisting and extreme tasks making it scalable to millions of labels.

Highlights: We (i) further the state-of-the-art on 23 out of 24 metrics across 4 popular benchmark datasets (ii) reduce the inference time to half of the previous fastest state-of-the-art, and (iii) require only $1/53$x FLOPS as compared to previous pre-trained transformer based approaches.

2 Related Work

Extreme Classification: The focus of a majority of initial works in this domain has been on designing one-vs-rest (Babbar and Schölkopf, 2017), tree-based (Prabhu et al., 2018; Chalkidis et al., 2019; Khandagale et al., 2020) or label embedding based (Bhatia et al., 2015) classifiers with fixed features in the form of bag-of-words representation. With advances in deep learning, jointly learning label and input text embeddings has also been developed (Tang et al., 2015; Wang et al., 2018). For XML tasks, recent techniques based on attention mechanism (You et al., 2019) and pre-trained transformer models (Chang et al., 2020; Ye et al., 2020; Jiang et al., 2021; Yu et al., 2020) have shown great promise. In the context of CNNs for text classification, while (Wang et al., 2017) extended (Kim, 2014) for short input sequences, (Liu et al., 2017) built upon the same for XML tasks.

Short-text Extreme Classification: In XML tasks where the inputs are short text queries, there has been a slew of recent works. Based on the availability of label meta-data, these works can be divided into two categories: (i) ones which make no assumptions regarding label text, i.e., labels are numeric identifiers, such as ASTEC (Dahiya et al., 2021b) and (ii) others which assume that the labels are endowed with clean label text which include DECAF (Mittal et al., 2021a), GALAXC, ECLARE (Mittal et al., 2021b), and SIAMESEXML (Dahiya et al., 2021a). Even though the additional label meta-data is useful, it is usually only known for only a small subset of all labels. Further, the former problem setup, which is the focus of this work, makes no assumption about label-text, and hence is a harder, more general and widely applicable.

We compare our model vis-à-vis the frugal ASTEC baseline, which uses a tfidf-weighted sum of word embeddings as inputs and comprises only of a single residual layer as its encoder. ASTEC further relies on the capabilites of ANNs (Malkov and Yashunin, 2020) for label-shortlisting while we create our scalable extension INCEPTIONXML+ through dynamic hard-negative mining of labels.

Drawbacks of conventional CNNs in short-text classification: Traditionally, in the usage of CNNs over words in text classification, the intent is to capture the occurrences of n-grams for representation learning (Kim, 2014; Liu et al., 2017). We argue that this formulation is unsuitable for short-text classification problems as (i) the implicit but incorrect assumption of proper *word-ordering* in short-text queries (Wang and Wang, 2016), and (ii) as explained next, the much *smaller sequence length* that restricts the effectiveness of convolution in CNNs over the inputs.

In the datasets derived from Wikipedia titles, 98% documents have 8 or less words, while 82% have 4 words or less (Table: 5 in Appendix). Moreover, 70% of the instances in AmazonTitles-670K consist of 8 words or less (Figure: 6). This makes the convolutional filters spanning over 4-8 words in Kim (2014); Liu et al. (2017); Wang et al. (2017) behave analogously to a weak fully connected layer with very few hidden units, and hence leading to feature maps with very few activations which are sub-optimal for representation learning. In context of the aforementioned problems, we hypothesize and empirically demonstrate the suitability of convolving over the embedding dimensions of the inputs instead of the words for short-text queries.

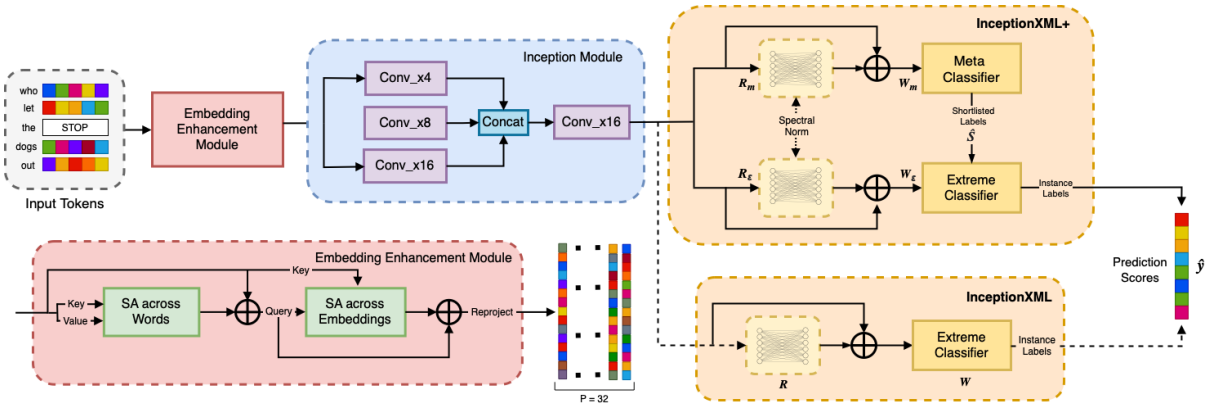

Figure 2: INCEPTIONXML(+) Framework. The convolution filters on the input data span only a subset of adjacent dimensions in the word embeddings while covering all the input tokens ('*who let the dogs out*'). The Embedding-enhancement module is shown in detail with its orthogonal self-attention layers followed by a projection layer.

3 Embedding Convolutions

By convolving over embeddings in a stacked setting, we enable the model to detect correlations or "coupled semantics" between different dimensions in the embedding space by processing a limited subset of semantics at a time. As compared to traditional convolutional operation, embedding convolutions create significantly larger and enriched activation maps for the same inputs, while requiring substantially lesser parameters by using smaller filters of size $\mathbb{R}^{S\times 16}$, where S is the maximum sequence length of the input. We show empirically that this modified approach works well for both *short* as well as *medium* queries of up to 32 words, significantly outperforming conventional CNN-based approaches (Liu et al., 2017; Kim, 2014) for short-text XML task.

As some readers might rightfully argue, pre-trained word embeddings are typically not trained with any incentive for localizing semantic information in the embedding dimension. To this end, we process the stacked word embeddings with self-attention based embedding enhancement module before applying embedding convolutions. This lets information flow across every pair of semantics irrespective of the spatial distance between them.

4 Proposed Model - InceptionXML

Problem Setup : Given a training set $\{x_i, y_i\}_{i=1}^{N}$, x_i represents an input short-text query, and the corresponding label set is represented by $y_i \in \{0,1\}^L$ where L denotes the total number of labels. It may be noted that even though $L \sim 10^6$, an instance is only annotated with a few positive labels (Table: 5). The goal is to learn a classifier which, for a novel test instance x', predicts the top-k labels towards better precision@k and propensity-scored precision@k (Bhatia et al., 2016) metrics. Towards this goal, the main body of our encoder consists of three modules that are applied sequentially on the word embeddings (Fig. 2). These are (i) an embedding enhancement module, (ii) embedding convolution layers and (iii) an extreme linear classifier.

4.1 Embedding Enhancement Module

This module takes stacked word embeddings lacking structure and context as input and makes it word order agnostic. Specifically, the module consists of two orthogonal attention layers (Doria, 2019) applied sequentially on the word and the embedding dimensions followed by a projection layer, effectively encoding global information both, on a word-level and on a semantic-level (Figure 4).

The sequential attention formulation in our embedding enhancement module is given by:

$$x_{sa} = \text{SA}(q = E(x),\ k = E(x),\ v = E(x))$$
$$x_{sa} = \text{SA}(q = x_{sa}^{T},\ k = E(x)^{T},\ v = E(x)^{T})^{T}$$

where $E(x)$ denotes the stacked word embeddings for a sample text input x such that $E(x) \in \mathbb{R}^{S\times d}$. Finally, each dimension of the intermediate embeddings x_{sa} is then projected to a p-dimensional space where $p = 32$ to obtain the final enhanced embeddings $x_{enh} \in \mathbb{R}^{p\times d}$. The information flow across the embeddings in this module followed by per-dimension projection makes x_{enh} independent of the word order in short-text queries and makes our model more robust to their lack of structure.

4.2 Embedding Convolution Layers

We employ three parallel branches of one-dimensional convolution layers V_i, $i \in [1, 2, 3]$ with filter sizes of w_i where $w_i \in [4, 8, 16]$ each with a stride of 4 along the embedding dimension and p output channels. Let h_{w_i} be the result of applying V_i over SA_{out}. We concatenate all resultant h_{w_i} row-wise before passing them to the next layer.

$$\begin{aligned} h_{w_i} &= V_i * x_{enh} \\ h_f &= V_f * [h_{w_1}, h_{w_2}, h_{w_3}] \end{aligned}$$

A final embedding convolutional layer V_f with kernel size of 16 and stride 4 is applied on the concatenated feature map, which is further flattened to form the final feature representation h_f. This formulation allows V_f to have an effective receptive field spanning $1/4^{th}$ of the enhanced embeddings, further obviating the locality constraints of CNNs as highlighted in section 3.

4.3 Extreme Linear Classifier

The first layer R transforms the feature map from the encoder with a skip-connection while keeping the dimensions same. The next linear layer W has one-vs-all classifiers for each label in the dataset which projects the features to the label space.

$$\hat{y} = \sigma(W \cdot (relu(R \cdot h_f) + h_f))$$

The model is trained end-to-end using binary cross entropy loss.

$$\text{BCE}(y, \hat{y}) = -\sum_{j \in L} (1 - y_j) \log(1 - \hat{y}_j) + y_j \log(\hat{y}_j)$$

5 InceptionXML+ Framework

INCEPTIONXML described previously scales to datasets with hundreds of thousands of labels. However, scaling up to millions of labels in its existing form is difficult as the loss computation in equation above involves calculation of loss over all L labels, a very large majority of which are negative labels for a given instance. Even with sufficient hardware resources, scaling up over the entire label space requires very large training times (Chang et al., 2020). We thus propose INCEPTIONXML+ framework, which improves existing hard-negative mining to enable scaling to output spaces in the order of millions along with an updated training schedule. Not only does the framework scale our encoder, but also significantly reduces the training time and computational cost (Table 6).

Hard Negative-Mining of Labels: While techniques have been studied for efficient hard-negative label mining under *fixed representation* of data points (Jain et al., 2019; Dahiya et al., 2021b), only recent algorithms (Jiang et al., 2021) have come up with *dynamic* hard negative-mining techniques. Following the approach popularized by these recent methods, our model makes predictions in two stages: (i) shortlisting top K label-clusters or "meta-labels" using a meta-classifier, and (ii) employing a computationally feasible number of one-vs-all classifiers corresponding to the labels included in the shortlisted clusters to get the final predicted labels and perform backpropagation.

Label Clustering To perform label clustering, we construct Hierarchical Label Tree (HLT) using the labels' Positive Instance Feature Aggregation (PIFA) representation over sparse BOW features of their training samples (Chang et al., 2020; Dahiya et al., 2021b). Specifically, we use balanced 2-means clustering to recursively partition the label set until we have a mapping C from L labels to L' label clusters where $L' \ll L$ (Table:5).

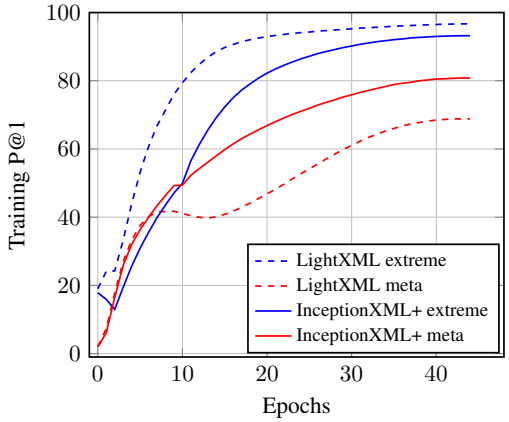

Figure 3: Progress of training (Precision@1) for the extreme and meta-classifier of LIGHTXML and INCEPTIONXML+ frameworks on AmazonTitles-670K

Drawbacks of LIGHTXML framework: When scaling our model using hard-negative mining as done in LIGHTXML (Jiang et al., 2021), we noticed that the performance of our encoder is bottlenecked by a poorly performing meta-classifier. From the training metrics (Fig: 3), we see a smooth increment in the P@1 values for the extreme classifier (dashed blue) while the meta-classifier is unable to catch-up (dashed red). This indicates that these two sub-tasks are not aligned well enough for the encoder to learn suitable common representations that work well simultaneously for both the

sub-tasks. Our observations also indicate the fact that the extreme task is easier to learn on shortlisted labels than the meta-task on label clusters, and the model tends to learn representations that benefit the extreme task at the expense of the meta-task.

Key Improvements Our changes to the hard-negative mining framework can be broadly grouped into two sets. Firstly, we propose architectural improvements meant to synchronize the two tasks in order to enable the encoder to learn better common representations. Secondly, we make modifications to the training loop in order to force the encoder to learn representations that improve the performance of the meta-classifier while remaining in sync with the extreme task. Next, we discuss these in detail.

5.1 Synchronized Architecture

To synchronize the training of extreme and meta-classifier tasks, we give them similar structures by adding a linear layer W_m with a residual connection R_m before the meta-classifier. Using the intermediate representation h_f from equation (1), this is given by :

$$\hat{y}_m = \sigma(W_m \cdot (relu(R_m \cdot h_f) + h_f))$$

We create a shortlist $\hat{\mathcal{S}}$ of all the labels in the top K label clusters as predicted by the meta-classifier using a label cluster to label mapping C^{-1}. Via the linear mapping W_e, extreme classifier then predicts the probability of the query belonging to only these shortlisted labels, instead of all L labels.

$$\hat{\mathcal{S}} = C^{-1}(\ top_K(\hat{y}_m, k)\ )$$
$$g_e = relu(R_e \cdot h_f) + h_f$$
$$\hat{y}_{e,l} = \sigma(W_{e,l} \cdot g_e),\ \forall l \in \hat{\mathcal{S}}$$

Architectural similarity of branches alone does not ensure strong common representation learning. To help the encoder learn suitable common representations, we further sync the two branches by (i) increasing the "extremeness" of the meta-task by enlarging the fan out of label clusters, and (ii) adding spectral norm to the penultimate linear layers of both heads to prevent the final features from drifting too far from each other (Dahiya et al., 2021b). Not only does this heavily improve (Table: 3) upon the original implementation of dynamic negative-hard mining as proposed in (Jiang et al., 2021), but also inherently combines the task of the two stages of the DeepXML pipeline (Dahiya et al., 2021b) into an end-to-end trainable model. Though substantial gains are observed from enlarging the fan out, this comes at a computational cost. Thus, in practice we aim to strike a balance (Table: 3) between number of clusters and model efficiency for non-trivial gains in accuracy.

5.2 Detached Training Schedule

To force the encoder to learn representations benefiting the meta-task, we detach i.e. stop the flow of gradients from the extreme classifier head to the encoder (Algorithm 1), for the initial 25% of the training loop. This results in shortlisting of harder negative labels for the extreme classifier to learn during training time and ensures higher recall during inference time (Table: 3). Detaching instead of simply removing the extreme classification head enables the module to continuously adapt to the changing encoder representations without allowing it to affect the training of the meta-classifier. This setting is possible because of the spectral norm applied to the weights of the penultimate layers in both the heads which ensures that the encoder learnt for the meta-task remains relevant for the extreme task when its gradients are re-attached.

Algorithm 1: Training algorithm for INCEPTIONXML+

```
for epoch in (1, epochs):
  for x, y in data:
    z = E(x)
    h = encoder(z)
    y_meta = meta_classifier(h)
    y_cluster = label_to_cluster(y)
    meta_loss = bce(y_meta, y_cluster)

    # shortlisting top K clusters
    top_k = get_top_K_clusters(y_meta, k)
    candidates = cluster_to_label(top_k)
    # add missing positive labels
    candidates = add_missing(candidates,y)

    # detached training
    if epoch <= epochs/4:
      h = h.detach()
    y_ext = ext_classifier(h, candidates)
    ext_loss = bce(y_ext, y, candidates)
    loss = meta_loss + ext_loss
    loss.backward()

    # gradient descent
    update(E, encoder, meta_classifier,
     ext_classifier)
```

Loss: The losses for the meta-classifier and the extreme classifier are given by:

$$\mathcal{L}_{meta} = \text{BCE}(y_m\ ,\ \hat{y}_m),$$
$$\mathcal{L}_{ext} = \text{BCE}(y_{e,l}\ ,\ \hat{y}_{e,l})\ \ \forall l \in \hat{\mathcal{S}}.$$

The final loss is the sum of the above losses i.e. $\mathcal{L} = \mathcal{L}_{meta} + \mathcal{L}_{ext}$. For prediction, the final ranking

is produced by only using the logits of the extreme classifier.

6 Experiments

Implementation Details: We initialize our embedding layer with 300-dimensional pre-trained GloVe embeddings (Pennington et al., 2014). Embeddings of words that do not exist in GloVe are initialized with a random vector sampled from the uniform distribution $\mathcal{U}(-0.25, 0.25)$. Following (Liu et al., 2017), we use a use white space separated pre-processing function for tokenization and remove the stop words and punctuation from the raw data using NLTK library. We train all our models on a single 32GB Nvidia V100 GPU. Further implementation details about batch size, learning rate, epochs etc. can be found in table 6 in the appendix.
Datasets: We evaluate the proposed INCEPTIONXML(+) frameworks on 4 publicly available benchmarks from the extreme classification repository (Bhatia et al., 2016). The details of the datasets are given in Table 5 (Appendix), the number of labels range from 350,000 (WikiSeeAlsoTitles-350K) to 2.8 Million (AmazonTitles-3M).

6.1 Main Results

INCEPTIONXML+ finds a sweet-spot (Fig. 1) between the two extreme ends of modern deep extreme classification pipelines - heavy transformer-based methods, and frugal architectures such as ASTEC. We show that replacing the pre-trained transformer encoder with our lightweight CNN-based encoder, combined with further improvements to the hard-negative mining pipeline leads to better prediction performance apart from faster training and the ability to scale to millions of labels.

As shown in Table 1, for most of the dataset-metric combinations, the proposed models, INCEPTIONXML(+), not only outperform the previous state-of-the-art ASTEC and but also its ensemble version ASTEC-3 with non-trivial gains. Notably, INCEPTIONXML gains an average of 3.9% and 6.9% over ASTEC on all three datasets except AmazonTitles-3M on the P@1 and PSP@1 metrics. Also, the following observations can be made :

- The proposed models achieves at least 10% relative improvement as compared to XML-CNN (Liu et al., 2017), which captures n-grams for representation learning showing the effectiveness of our approach as compared to conventional CNNs-based approaches.

Method	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5
	AmazonTitles-670K					
INCEPTIONXML+	41.28	37.04	33.92	27.02	30.05	32.72
INCEPTIONXML	**41.78**	**37.47**	**34.15**	**28.17**	**30.96**	**33.31**
ASTEC	39.97	35.73	32.59	27.59	29.79	31.71
ASTEC-3	40.63	36.22	33.00	28.07	30.17	32.07
LIGHTXML	41.57	37.19	33.90	25.23	28.79	31.92
APLC-XLNET	34.87	30.55	27.28	20.15	21.94	23.45
ATTENTIONXML	37.92	33.73	30.57	24.24	26.43	28.39
XML-CNN	35.02	31.37	28.45	21.99	24.93	26.84
DISMEC	38.12	34.03	31.15	22.26	25.45	28.67
PARABEL	38.00	33.54	30.10	23.10	25.57	27.61
BONSAI	38.46	33.91	30.53	23.62	26.19	28.41
MACH	34.92	31.18	28.56	20.56	23.14	25.79
	WikiSeeAlsoTitles-350K					
INCEPTIONXML+	20.77	14.61	11.44	10.26	12.41	14.15
INCEPTIONXML	**21.54**	**15.19**	**11.97**	**10.93**	**13.05**	**14.92**
ASTEC	20.42	14.44	11.39	9.83	12.05	13.94
ASTEC-3	20.61	14.58	11.49	9.91	12.16	14.04
LIGHTXML	21.25	14.36	11.11	9.60	11.48	13.05
APLC-XLNET	20.42	14.22	11.17	7.44	9.75	11.61
ATTENTIONXML	15.86	10.43	8.01	6.39	7.20	8.15
XML-CNN	17.75	12.34	9.73	8.24	9.72	11.15
DISMEC	16.61	11.57	9.14	7.48	9.19	10.74
PARABEL	17.24	11.61	8.92	7.56	8.83	9.96
BONSAI	17.95	12.27	9.56	8.16	9.68	11.07
MACH	14.79	9.57	7.13	6.45	7.02	7.54
	WikiTitles-500K					
INCEPTIONXML+	45.24	25.91	18.36	19.24	19.38	19.50
INCEPTIONXML	**47.28**	**27.14**	**19.39**	**20.79**	**21.01**	**21.17**
ASTEC	46.01	25.62	18.18	18.62	18.59	18.95
ASTEC-3	46.60	26.03	18.50	18.89	18.90	19.30
LIGHTXML	47.17	25.85	18.14	17.64	17.54	17.50
APLC-XLNET	43.56	23.01	16.58	14.73	13.19	13.47
ATTENTIONXML	42.89	22.71	15.89	15.12	14.32	14.22
XML-CNN	43.45	23.24	16.53	15.64	14.74	14.98
DISMEC	39.89	21.23	14.96	15.89	15.15	15.43
PARABEL	42.50	23.04	16.21	16.55	16.12	16.16
BONSAI	42.60	23.08	16.25	17.38	16.85	16.90
MACH	33.74	15.62	10.41	11.43	8.98	8.35
	AmazonTitles-3M					
INCEPTIONXML+	46.95	**45.28**	**43.45**	**16.02**	**18.94**	**21.03**
ASTEC	**47.64**	44.66	42.36	15.88	18.59	20.60
ATTENTIONXML	46.00	42.81	40.59	12.81	15.03	16.71
DISMEC	41.13	38.89	37.07	11.98	14.55	16.42
PARABEL	46.42	43.81	41.71	12.94	15.58	17.55
BONSAI	46.89	44.38	42.30	13.78	16.66	18.75
MACH	37.10	33.57	31.33	7.51	8.61	9.46

Table 1: Comparison of InceptionXML to state-of-the-art algorithms on benchmark datasets. The best-performing approach is in **bold** and the second best is underlined. The algorithms omitted in AmazonTitles-3M do not scale for this dataset on 1 Nvidia V100 GPU.

- Significant gains of up to 20% in are obtained compared to the transformer based APLC-XLNET (Ye et al., 2020). We also outperform LIGHTXML (Jiang et al., 2021) on all benchmarks despite having a comparatively light-weight architecture. Notably, none of these architectures scale to AmazonTitles-3M dataset, demonstrating the efficacy and scalability of the proposed light-weight encoder in INCEPTIONXML+ framework.
- Our models also significantly outperform non-deep learning approaches using bag-of-words

representations such as the label-tree based algorithms like BONSAI (Khandagale et al., 2020) and PARABEL (Prabhu et al., 2018), and DISMEC (Babbar and Schölkopf, 2017) which is an embarrassingly parallel implementation of LIBLINEAR (Fan et al., 2008).

- It maybe noted that INCEPTIONXML outperforms its scaled counterpart on all benchmarks, especially for the PSP metrics. While INCEPTIONXML always gets information about all negative labels instead of only hard-negatives during training, it also makes prediction over the entire label space. On the other hand, INCEPTIONXML+ has to rely on the meta-classifier for label shortlisting. As a 100% recall rate cannot be ensured for label-shortlisting, some positive label-clusters are occasionally missed leading to slightly reduced performance.

6.2 Discussion on Computational Cost

Training time: As shown in Table 2, training time of INCEPTIONXML+ ranges from 4.3 hours on WikiSeeAlsoTitles-350K & AmazonTitles-670K datasets to 27.2 hours on AmazonTitles-3M. We observe a ~44% decrement in training time by scaling our encoder in the INCEPTIONXML+ Framework as compared to the unscaled INCEPTIONXML. As expected, our models train much faster than transformer based approaches (LIGHTXML, APLC-XLNET) while being comparable with ASTEC.

Model Size: INCEPTIONXML is extremely lightweight in terms of model size containing only 400K parameters while INCEPTIONXML+ contains only 630K parameters, which is multiple orders of magnitude lesser compared to pretrained transformer based models with ~110 million parameters. Further, our models are approximately 8-10x smaller compared to ASTEC which needs to store ANNS graphs for label centroids and training data-points for performance leading to exceptionally large model size (Table 2).

Flops: To compute flops, we use the standard FVCORE library[1] from facebook. Notably, INCEPTIONXML+ performs favourably with LIGHTXML while requiring only 1/53x flops on average, and INCEPTIONXML significantly outperforms the same with 1/8x flops (Table: 2).

Inference Time: Inference time has been calculated considering a batch size of 1 on a single Nvidia 32GB V100 GPU. We note that our proposed INCEPTIONXML(+) architectures not only have the lowest inference times on all datasets, but also our framework reduces the inference time to half as compared to the previous fastest ASTEC. However, using transformer based models results in 3-5x slower inference as compared to INCEPTIONXML+.

To summarize, our models improve by approximately an order of magnitude on both model sizes and floating point operations compared to recent state-of-the-art approaches, and are economical in terms of training time with very low inference times, all while achieving state-of-the-art performance on all important metrics and benchmarks.

	Giga FLOPS	Training Time (hr)	Inference Time (msec)	Model Size (GB)
Method	**AmazonTitles-670K**			
INCEPTIONXML+	**0.049**	4.3	**4.67**	1.4
INCEPTIONXML	0.334	7.7	7.97	**1.3**
ASTEC	*	**3.0**	8.17	10.93
LIGHTXML	2.775	13.5	13.11	1.6
APLC-XLNET	3.564	16.1	24.66	1.8
	WikiSeeAlsoTitles-350K			
INCEPTIONXML+	**0.027**	4.3	**4.36**	0.80
INCEPTIONXML	0.176	6.3	5.60	**0.73**
ASTEC	*	**2.9**	9.70	7.41
LIGHTXML	1.214	14.5	10.35	1.0
APLC-XLNET	2.248	16.0	18.39	1.5
	WikiTitles-500K			
INCEPTIONXML+	**0.029**	10.3	**4.37**	1.2
INCEPTIONXML	0.247	20	6.45	**1.1**
ASTEC	*	**7.3**	9.97	15.15
LIGHTXML	1.742	22.4	11.33	1.5
APLC-XLNET	2.51	25.1	20.25	1.9

Table 2: Comparison of algorithms in terms of Giga-Flops, Training/Inference Time and Model Size. * Due to the external ANNS module used in ASTEC for label shortlisting, it is not possible to compute its flops.

6.3 Ablation Results

Self-Attention Layers: The sequentially applied self-attention layers improve INCEPTIONXML's performance by only 1% at max on the performance metrics as shown in Fig. 4. This further demonstrates the superior representation learning capability of our encoder for short-text queries as even without the self-attention layers, our model outperforms the ensemble model ASTEC-3 and the transformer model LIGHTXML.

InceptionXML+: Table 3 shows a comparison of the proposed INCEPTIONXML+ pipeline vis-à-vis LIGHTXML for AmazonTitles-670K dataset. It is clear that the INCEPTIONXML+ framework sig-

[1] https://github.com/facebookresearch/fvcore/blob/main/docs/flop_count.md

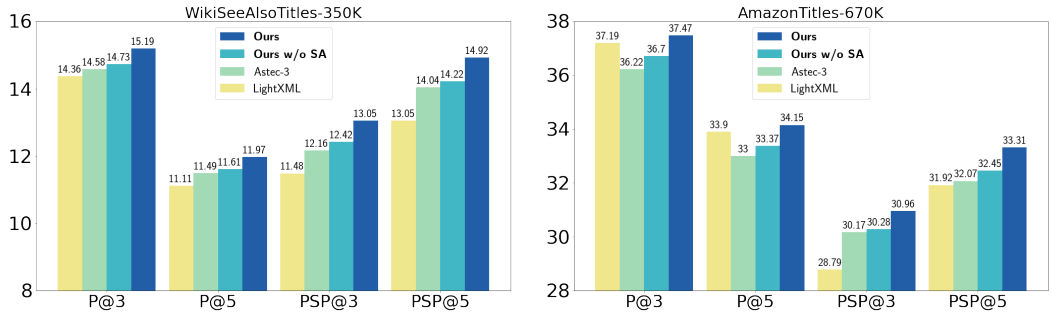

Figure 4: Performance with and w/o the self-attention layers on WikiSeeAlsoTitles-350K & AmazonTitles-670K

nificantly improves upon the hard-negative mining technique as proposed in LIGHTXML in terms of performance in both P@K and PSP@K metrics. Note that we keep the shortlisted labels consistent by doubling the number of shortlisted meta-labels as the fan-out doubles. It may be also be noted that as the fan-out increases, our detached training method improves the results more prominently. This can be attributed to the increased "extremeness" of the meta-task which ensures that the representations learnt by the encoder for the meta-task become increasingly more relevant to the extreme-task when the gradients of the extreme classifier are re-attached during training.

(L', Top_K)	Model	P@1	P@5	PSP@1	PSP5
8K, 100	Ours	40.26	32.75	26.05	31.21
	Ours w/o Detaching	40.13	32.68	25.75	31.07
	in LightXML Framework	39.40	32.36	25.14	30.38
16K, 200	Ours	40.67	33.27	26.34	31.81
	Ours w/o Detaching	40.51	32.95	26.03	31.36
	in LightXML Framework	39.47	32.43	24.89	30.67
32K, 400	Ours	41.01	33.65	26.75	32.32
	Ours w/o Detaching	40.24	33.09	26.07	31.67
	in LightXML Framework	39.58	34.81	24.45	30.73
65K, 800	Ours	**41.28**	**33.92**	27.02	**32.72**
	Ours w/o Detaching	40.47	33.23	26.40	31.97
	in LightXML Framework	39.27	32.77	23.54	30.54
_, _	in DeepXML Pipeline	38.53	32.21	**27.80**	31.62

Table 3: Impact of increasing fan-out of label clusters (L') on InceptionXML+ Framework (*Ours*) and LightXML Framework over AmazonTitles-670K.

Robustness to Lack of Word-order: For testing the robustness of our method to the order of words in input data, we train the InceptionXML+ on the original training data for AmazonTitles-670K, but randomly permute the words in test set, and evaluate the performance. This is repeated 10 times with different test set permutations (Table 4). We witness only a minor dip in performance across the metrics still outperforming ASTEC-3 and demonstrating the robustness of our encoder to lack of structure in short-text queries.

Test data	P@3	P@5
Original AmazonTitles-670K	37.04	33.92
Permuted Word-order	36.01 ± 0.05	32.86 ± 0.03
Original WikiSeeAlsoTitles-350K	14.61	11.44
Permuted Word-order	14.45 ± 0.03	11.32 ± 0.02

Table 4: Comparison of results with original test data and that obtained by permuting the word order in the test set for INCEPTIONXML+

InceptionXML in DeepXML Framework: We integrate our encoder with the DeepXML (Dahiya et al., 2021b) pipeline as used by ASTEC and find it inflexible to improve upon due to the requirement of fixed representations for their label shortlisting strategy. Moreover, when using our encoder as a drop-in replacement, we find our encoder's performance degrades in terms of precision in the DeepXML Framework as compared to the performance in the vanilla LIGHTXML Framework (Table 3: last row). This indicates the overall advantage of using dynamic hard-negative mining as compared to techniques requiring fixed representations.

7 Conclusion

In this work, we develop a lightweight CNN-based encoder for the task of short-text extreme classification. Augmented with a self-attention based word-order agnostic module, the proposed encoder betters state-of-the-art performance on all popular benchmark datasets. By synchronizing the training of extreme and meta-classifiers, we make improvements to the label hard-negative mining pipeline and develop a framework INCEPTIONXML+ that scales our encoder to dataset million of labels. Importantly, these capabilities are achieved while being computationally inexpensive in training, inference, and model size.

References

R. Babbar and B. Schölkopf. 2017. DiSMEC: Distributed Sparse Machines for Extreme Multi-label Classification. In *WSDM*.

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

K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain. 2015. Sparse Local Embeddings for Extreme Multi-label Classification. In *NIPS*.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, and Ion Androutsopoulos. 2019. Large-scale multi-label text classification on eu legislation. *arXiv preprint arXiv:1906.02192*.

W-C. Chang, H.-F. Yu, K. Zhong, Y. Yang, and I. Dhillon. 2020. Taming Pretrained Transformers for Extreme Multi-label Text Classification. In *KDD*.

Wei-Cheng Chang, Daniel Jiang, Hsiang-Fu Yu, Choon-Hui Teo, Jiong Zhang, Kai Zhong, Kedarnath Kolluri, Qie Hu, Nikhil Shandilya, Vyacheslav Ievgrafov, et al. 2021. Extreme multi-label learning for semantic matching in product search. *arXiv preprint arXiv:2106.12657*.

K. Dahiya, A. Agarwal, D. Saini, K. Gururaj, J. Jiao, A. Singh, S. Agarwal, P. Kar, and M. Varma. 2021a. Siamesexml: Siamese networks meet extreme classifiers with 100m labels. In *Proceedings of the International Conference on Machine Learning*.

K. Dahiya, D. Saini, A. Mittal, A. Shaw, K. Dave, A. Soni, H. Jain, S. Agarwal, and M. Varma. 2021b. DeepXML: A Deep Extreme Multi-Label Learning Framework Applied to Short Text Documents. In *WSDM*.

Sebastian Doria. 2019. Simple self-attention https://github.com/sdoria/simpleselfattention.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *the Journal of machine Learning research*, 9:1871–1874.

H. Jain, V. Balasubramanian, B. Chunduri, and M. Varma. 2019. Slice: Scalable Linear Extreme Classifiers trained on 100 Million Labels for Related Searches. In *WSDM*.

H. Jain, Y. Prabhu, and M. Varma. 2016. Extreme Multi-label Loss Functions for Recommendation, Tagging, Ranking and Other Missing Label Applications. In *KDD*.

Ting Jiang, Deqing Wang, Leilei Sun, Huayi Yang, Zhengyang Zhao, and Fuzhen Zhuang. 2021. Lightxml: Transformer with dynamic negative sampling for high-performance extreme multi-label text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7987–7994.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

S. Khandagale, H. Xiao, and R. Babbar. 2020. Bonsai: diverse and shallow trees for extreme multi-label classification. *Machine Learning*, 109(11):2099–2119.

Y. Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *EMNLP*.

J. Liu, W. Chang, Y. Wu, and Y. Yang. 2017. Deep Learning for Extreme Multi-label Text Classification. In *SIGIR*.

Yu A. Malkov and D. A. Yashunin. 2020. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42:824–836.

A. Mittal, K. Dahiya, S. Agrawal, D. Saini, S. Agarwal, P. Kar, and M. Varma. 2021a. Decaf: Deep extreme classification with label features. In *Proceedings of the ACM International Conference on Web Search and Data Mining*.

A. Mittal, N. Sachdeva, S. Agrawal, S. Agarwal, P. Kar, and M. Varma. 2021b. Eclare: Extreme classification with label graph correlations. In *Proceedings of The ACM International World Wide Web Conference*.

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

Y. Prabhu, A. Kag, S. Harsola, R. Agrawal, and M. Varma. 2018. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *WWW*.

Jian Tang, Meng Qu, and Qiaozhu Mei. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1165–1174.

Kshitij Tayal, Nikhil Rao, Saurabh Agarwal, Xiaowei Jia, Karthik Subbian, and Vipin Kumar. 2020. Regularized graph convolutional networks for short text classification. In *Proceedings of the 28th International Conference on Computational Linguistics: Industry Track*, pages 236–242.

Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018. Joint embedding of words and labels for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2321–2331.

Jin Wang, Zhongyuan Wang, Dawei Zhang, and Jun Yan. 2017. Combining knowledge with deep convolutional neural networks for short text classification. In *IJCAI*.

Zhongyuan Wang and Haixun Wang. 2016. Understanding short texts. In *the Association for Computational Linguistics (ACL) (Tutorial)*.

H. Ye, Z. Chen, D.-H. Wang, and Davison B. D. 2020. Pretrained Generalized Autoregressive Model with Adaptive Probabilistic Label Clusters for Extreme Multi-label Text Classification. In *ICML*.

R. You, Z. Zhang, Z. Wang, S. Dai, H. Mamitsuka, and S. Zhu. 2019. Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. In *Neurips*.

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

A Appendix

A.1 Evaluation Metrics

As stated earlier, the main application of short-text XML framework is in recommendation systems and web-advertising, where the objective of an algorithm is to correctly recommend/advertise among the top-k slots. Thus, for evaluation of the methods, we use precision at k (denoted by $P@k$), and its propensity scored variant (denoted by $PSP@k$) (Jain et al., 2016). These are standard and widely used metrics by the XML community (Bhatia et al., 2016).

For each test sample with observed ground truth label vector $y \in \{0,1\}^L$ and predicted vector $\hat{y} \in \mathbb{R}^L$, $P@k$ is given by :

$$P@k(y,\hat{y}) := \frac{1}{k} \sum_{\ell \in top@k(\hat{y})} y_\ell$$

where $top@k(\hat{y})$ returns the k largest indices of $\hat{y}$.

Since $P@k$ treats all the labels equally, it doesn't reveal the performance of the model on tail labels. However, because of the long-tailed distribution in extreme classification datasets, one of the main challenges is to predict tail labels correctly, which are more valuable and informative compared to head classes, and it is essential to measure the performance of the model specifically on tail labels. By alluding to the phenomenon of missing labels in the extreme classification setting and its relation to tail-labels, $PSP@k$ was introduced in Jain et al. (2016) as an unbiased variant of original precision at k under no missing labels. This is widely used by the community to compare the relative performance of algorithms on tail-labels, and is also another metric used in our relative comparisons among various extreme classification algorithms in Tables 1 and 3 for main results and ablation tests respectively.

A.2 Vocabulary & Word Embedding

As opposed to taking their TF-IDF weighted linear combination as used in some recent works (Dahiya et al., 2021b,a; Mittal et al., 2021a) or the more conventional bag-of-words representations approaches like (Babbar and Schölkopf, 2017; Prabhu et al., 2018), we use the approach of stacking Glove embeddings (Pennington et al., 2014) as done in (Kim, 2014; Liu et al., 2017; Wang et al., 2017). For a fair comparison, we use exact same size of vocabulary space as (Dahiya et al., 2021b) for all benchmark datasets. As state before, we use wide-space tokenizer and find empirically that our model works better without using sub-word tokenizers like word-piece or sub-word based embeddings like fastText (Joulin et al., 2016).

A.3 Impact of Permuting Embedding Dimensions:

To show that INCEPTIONXML is independent of the order of embedding dimensions, we randomly permute the dimensions of the input word embeddings before start of the training, train with this fixed permuted order and evaluate in the standard manner. This is repeated 10 times with different permutations before training. Only *slight* variation in performance metrics can be observed in figure 5 with respect to the median of each boxplot which implies that the order of embedding dimensions has *little or no* impact over the results of our model.

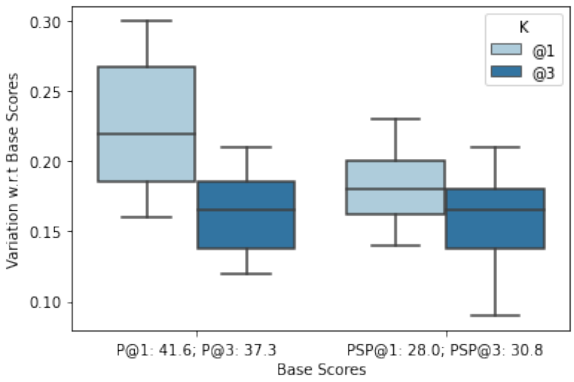

Figure 5: Variation in scores after shuffling embedding dimensions randomly before start of training for AmazonTitles-670K dataset. The boxplot only shows a variation in the performance metrics from the 10 runs. Different scores and statistics can be obtained by adding the values in the y-axis to the base scores on the x-axis.

A.4 Dataset Details

The key statistics of the datasets used in our evaluation are given in Table 5. These are open benchmark datasets taken from the Extreme Classification repository [2]. Figure 6 details the distribution of sequence lengths in AmazonTitles-670K dataset. Also, through the last two columns in Table 5 we confirm the short-text nature of these datasets.

[2] http://manikvarma.org/downloads/XC/XMLRepository.html

Datasets	# Features	# Labels	# Training	# Test	APpL	ALpP	#W≤ 4	#W≤ 8
WikiSeeAlsoTitles-350K	91,414	352,072	629,418	162,491	5.24	2.33	82%	98%
WikiTitles-500K	185,479	501,070	1,699,722	722,678	23.62	4.89	83%	98%
AmazonTitles-670K	66,666	670,091	485,176	150,875	5.11	5.39	40%	70%
AmazonTitles-3M	165,431	2,812,281	1,712,536	739,665	31.55	36.18	15%	52%

Table 5: Dataset Statistics. APpL denotes the average data points per label, ALpP the average number of labels per point. #W is the number of words in the training samples.

Model	# epochs	Batch Size	lr_{max}	L'	Top_K	ALpC
	AmazonTitles-670K					
INCEPTIONXML	42	128	0.005	-	-	-
INCEPTIONXML+	35	256	0.008	65536	800	11
	WikiSeeAlsoTitles-350K					
INCEPTIONXML	42	128	0.005	-	-	-
INCEPTIONXML+	30	256	0.008	32768	800	10
	WikiTitles-500K					
INCEPTIONXML	42	128	0.005	-	-	-
INCEPTIONXML+	27	256	0.008	32768	800	11
	AmazonTitles-3M					
INCEPTIONXML	-	-	-	-	-	-
INCEPTIONXML+	35	128	0.008	131072	800	22

Table 6: Hyperparameters of INCEPTIONXML(+) architectures. For INCEPTIONXML+, L' and Top_K denote the number of label-clusters and the number of clusters shortlisted per dataset while ALpC denotes the average labels per cluster.

A.5 Hyperparameters

We present the details of our hyperparameters like learning rate, batch size and number of epochs in table 6 along with the details of the label-clusters as used in INCEPTIONXML+. Note that we train our models using a cyclic learning rate and lr_{max} denotes the maximum learning rate in the cycle.

B Responsible NLP Research Checklist

B.1 Limitations

- Given that the convolution operation spans over the entire document length, the proposed method is mostly suited for short and medium length text sequences.

- Our method is agnostic to the presence of label texts, which despite constraining the problem to a much smaller subset, have been shown to help in achieving better prediction performance.

B.2 Potential Risks

We do not foresee any potential risks of our methods. Rather, it should be seen to be as energy-efficient alternatives to large-transformer models for the core textual and language problems encountered in search and recommendation.

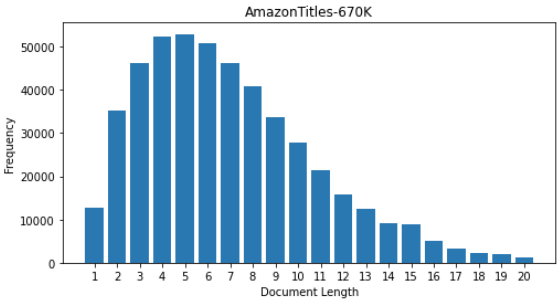

Figure 6: Sequence lengths of the input instance plotted against corresponding frequency for AmazonTitles-670K dataset. For this dataset, 70% of training instances have ≤ 8 words, and 30% have ≤ 4 words.