Code and Data Release

Due to large size of the datafiles, the MM-AmazonTitles-300K dataset and the MUFIN model trained on this dataset has been released at the following URL

http://manikvarma.org/pubs/mittal22-supp.pdf

The MUFIN code has been submitted alongside this paper as supplementary material on the submission website. However, it is available on the above mentioned URL as well.

Discussion on Ethical Considerations, Limitations and Future Work

Ethical Considerations: The MM-AmazonTitles-300K dataset was curated from a publicly available crawl [29] and utilizes no personally identifiable or human subject data. The MUFIN approach is itself applied to tasks such as product-to-product recommendation, bid query prediction, and outfit completion that seek to improve user experience when browsing for products. We are unaware of any direct applications of the MUFIN approach that can have negative societal impact.

Limitations and Future Work: We identify two potential avenues for further improving MUFIN's performance. Firstly, Fig. 4 indicates that the MUFIN-$(\alpha = 1)$ is better at predicting tail/rare labels correctly whereas MUFIN performs better on head/popular labels. Although MUFIN outperforms MUFIN-$(\alpha = 1)$ overall (see Tab. 4), this indicates towards a possibility for a third variant that scores rare and popular labels differently to get the best of both worlds. Secondly, it is common for products on e-commerce and other portals to be endowed with taxonomies that can give direct information about related products. The use of such tree/graph metadata has been shown in XC literature [27] to positively impact performance. Incorporating such relational metadata at the scale of millions of products is an interesting direction.

A. MUFIN Pseudocode and Outline

Fig. 7 below presents MUFIN's prediction pipeline. The ViT+SentenceBERT encoder $\mathcal{E}$ trained in a Siamese fashion embeds a test point X as a vector $\hat{\boldsymbol{x}}^1$ which is used to shortlist $O(\log L)$ labels using augmented retrieval. For each shortlisted label e.g. label number 42 in the example, a label-adapted representation $\hat{\boldsymbol{x}}^{2,42}$ is generated by applying cross-attention between test point representation $\hat{\mathbf{X}}^1$ and label representation $\hat{\mathbf{Z}}^1_{42}$. The dot product between $\hat{\boldsymbol{x}}^{2,42}$ and the classifier for this label $\boldsymbol{w}_{42}$ gives final score of label 42 for this test point.

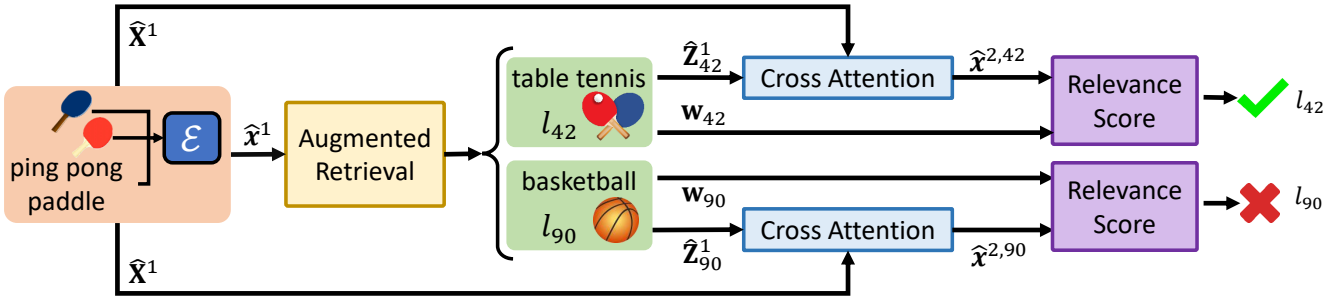


Figure 7. Scalable prediction pipeline deployed by MUFIN

Prediction Time Complexity for MUFIN. For sake of simplicity, let us assume that all labels and datapoints contain the same number m of descriptors. Computing the bag of pre-embeddings $\hat{\mathbf{X}}^0_t$ takes $\mathcal{O}\left(m \cdot (\text{ENC} + D)\right)$ time assuming the encoders $\mathcal{E}_V, \mathcal{E}_T$ take ENC time to encode a descriptor into a D-dimensional vector. Passing $\hat{\mathbf{X}}^0_t$ through the self-attention block $\mathcal{A}_S$ to obtain $\hat{\mathbf{X}}^1_t$ takes $\mathcal{O}\left(mD^2 + m^2D\right)$ time after which computing the vector representation $\hat{\boldsymbol{x}}^1_t$ takes an additional $\mathcal{O}\left(mD\right)$ time. Querying the ANNS structure NN^x to obtain the shortlist R_t of $\mathcal{O}\left(\log L\right)$ labels takes at most $\mathcal{O}\left(D \log L\right)$ time [24]. Applying the cross-attention block $\mathcal{A}_C$ with respect to each of these shortlisted labels takes a total of $\mathcal{O}\left((mD^2 + m^2D) \log L\right)$ time. Vectorizing these outputs to obtain $\hat{\boldsymbol{x}}^{2,l}_t$ and applying the label classifiers $\boldsymbol{w}_l$ takes at most $\mathcal{O}\left(mD \log L\right)$ time. This brings the total time complexity of the prediction pipeline per test datapoint to be at most $\mathcal{O}\left(m \cdot \text{ENC} + (m + D) \cdot mD \log L\right)$. In practice, MUFIN offered predictions within a 3-4 milliseconds even on tasks such as A2Q-4M with several millions of labels.

B. Datasets

(Discussion continued from the **Datasets** subsection in Sec. 4)

MM-AmazonTitles-300K.: This dataset was curated from an existing Amazon click dump [29]. Given a product as a datapoint, the aim is to recommend the subset of the most relevant (e.g. frequently bought-together) products from a catalog of over 300K unique products. Each product is represented using multiple descriptors including a product title and up to 15 product images. The dataset released in the supplementary [link] consists of multiple product entries in JSON format. Each product is associated with multiple tags described below:

1. "ASIN": this acts as a unique identifier (UID) for the product
2. "title": this represents a textual title for the product
3. "images": this presents a list of URLs pointing to multiple (up to 15) images of the product
4. "also_buy": this presents a list of UIDs of products that were frequently bought together with the product

Products having no images as well as no title were not included in the dataset. To generate the train test split, guidelines from [2] were closely followed.

C. Baselines and Related work

(Discussion continued from the **Baselines** subsection in Sec. 4)

Baselines for the Polyvore-Disjoint dataset: Performance numbers for all baselines in Tab. 3 were taken directly from published results [10, 18, 21, 32]. For sake of completion, each baseline method for this dataset is described in below in brief.

- **ADDE-O [10]**: Learns attribute-driven disentangled representations.
- **CSA-NET [21]**: Learns a category-based subspace attention network for scalable indexing and retrieval. This work introduced the notion of *outfit ranking loss* that considers the item-relationship of an entire outfit.
- **Type-aware [35]**: Learns an image embedding that respects item type, and jointly learns notions of item similarity and compatibility in an end-to-end manner. This method uses both visual and textual descriptors to represent an outfit.
- **S-VAL [18]**: Learns outfit representation by self-supervision. The self-supervision tasks used in the paper were histogram prediction and learning to discriminate shapeless patches and textures from different images.
- **SCE-Net [34]**: Learns model parameters by optimizing a combination of loss functions. For Polyvore-Disjoint, SCE-Net uses two objective loss function, namely VSE and Sim. The VSE loss requires that image embeddings and textual embeddings of the same outfit be embedded together. The Sim loss encourages images and descriptions of similar products to be embedded close to each other.
- **SSVR-Net [32]**: Learns model parameters using semi-supervised learning. This work uses a Siamese architecture trained using triplet loss that takes an input image, a positive instance (an affine transformation of the image) and a negative instance (an image with color transformations such as random gray scaling, jittering *etc*.).

Baselines for the MM-AmazonTitles-300K dataset: Baselines for this dataset are divided in two sets.

1. **Textual Methods**: Methods in the first set correspond to state-of-the-art extreme classification techniques including, AttentionXML, SiameseXML, ECLARE, Bonsai, MACH and XT. These methods are designed to be text-based and as such use only textual descriptors of a product. Consequently, each product was represented for these methods using its product title alone. Hyperparameters for each method were used as suggested by the respective papers. With the exception of the MACH method, all these methods enjoy an $\mathcal{O}(\log L)$ prediction time complexity similar to MUFIN.
2. **Visual + Textual Methods**: These methods include CLIP [30] and VisualBert [19] that use both visual and textual descriptors of a product.

As before, each baseline method for this dataset is described in below in brief. The details of how CLIP and VisualBERT were augmented and fine-tuned are discussed thereafter.

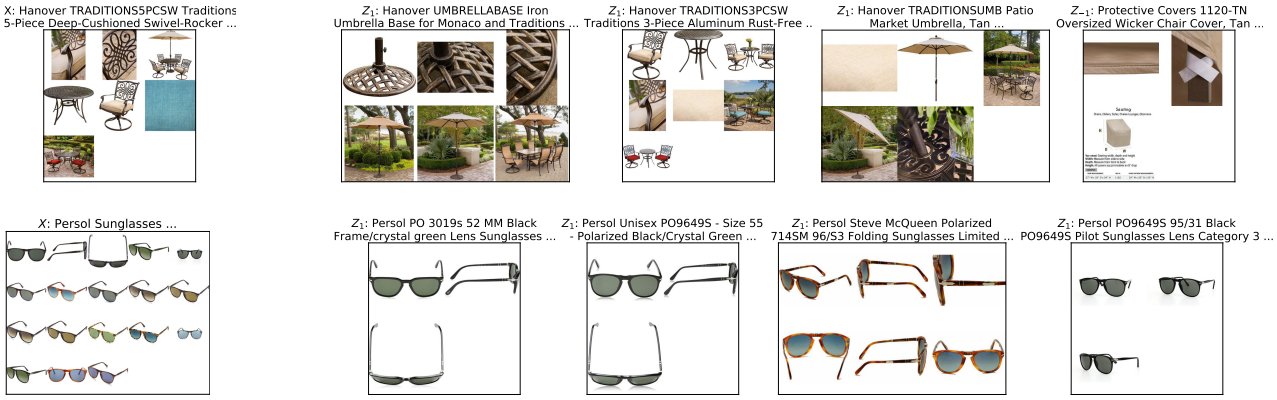
- **SiameseXML [5]**: Melds Siamese networks with one-vs-all classifiers. SiameseXML retrieves label shortlists using multiple ANNS structures unlike MUFIN that uses a single ANNS structure. The shortlisted labels are then ranked according to scores obtained from label-wise one-vs-all classifiers. Add SiameseXML implementation details
- **ECLARE [27]**: Exploits label graphs to obtain superior label representations with an aim to improve performance on rare labels. Since the MM-AmazonTitles-300K dataset does not provide a label graph natively, a label graph was mined by performing random walks using the label vectors as suggested in [27].
- **AttentionXML [39]**: Learns to partition labels using a shallow and wide PLT (depth between 2-3). A context vector is learnt per label that is used to generate label-specific datapoint representations. We note that the cross-attention block $\mathcal{A}_C$ used by MUFIN similarly produces label-adapted datapoint representations.
- **Bonsai [16]**: Implements a scalable tree-based classifier by learning a label hierarchy over the labels by representing each label using its bag-of-words (BoW) centroid vectors.
- **MACH [25]**: Learns an ensemble of 32 learners where each learner randomly partitions labels into several hash bins. Models are learnt to predict the hash bits corresponding to each learner. At prediction time, a majority vote is taken over all the learners to boost confidence. The method offers a prediction time of $\mathcal{O}(L)$.
- **XT [37]**: Generalizes the hierarchical softmax approach popular for multi-class problems to multi-label problems using a probabilistic label tree. Recall from the discussion in Sec. 1 that in multi-class classification, the objective is to predict a single mutually exclusive label for a given datapoint whereas in multi-label classification, the goal is to annotate datapoints with the most relevant *subset* (one or more) of labels.
- **VisualBert [19]**: Uses pre-trained Resnet-101 embeddings to represent images. Given a data point with multiple visual and textual descriptors, the visual descriptors are encoded and fed as tokens into a BERT (large) architecture alongside textual tokens from the textual descriptors. In this sense, VisualBert can be seen as utilizing early fusion.
- **CLIP [30]**: Uses a ViT architecture to encode images and a BERT architecture to encode text. The method pre-trains the architectures to encode relevant image-text pairs together. Subsequently, a late fusion architecture is learnt that fuses an image embedding and a text embedding into a joint representation over which a scoring model can be learnt.

C.1. Adapting CLIP and VisualBERT to MM-AmazonTitles-300K and Fine-tuning Details

Datapoint/label embedding Architecture: As mentioned above, CLIP uses separate encoders to encode images and text that can offer a bag of embeddings $\mathcal{E}_{\text{CP}}(X_i) \in \mathbb{R}^{m_i \times D}$. This bag of embeddings was fed into a fresh instantiation of the self-attention block $\mathcal{A}_S$ to obtain datapoint/label embeddings that we name $\hat{\boldsymbol{x}}^{\text{CP}}$ and $\hat{\boldsymbol{z}}_l^{\text{CP}}$. These are analogous to the $\hat{\boldsymbol{x}}^1$ and $\hat{\boldsymbol{z}}_l^1$ representations used by MUFIN. On the other hand, VisualBert uses a BERT architecture and pre-trained Resnet101 embeddings for images as tokens along with textual tokens to itself encode each datapoint/label as a vector to give embeddings $\hat{\boldsymbol{x}}^{\text{VB}}$ and $\hat{\boldsymbol{z}}_l^{\text{VB}}$. Note that VisualBERT was not offered the self-attention block since it itself performs similar self-attention operations within the layers of its BERT (large) architecture.
Retrieval: For VisualBERT, retrieval was performed using an ANNS structure created over the label embeddings $\hat{\boldsymbol{z}}_l^{\text{VB}}$. For CLIP, since it offers bag embeddings $\mathcal{E}_{\text{CP}}(Z_L) \in \mathbb{R}^{m_l \times D}$ one per descriptor of the label, augmented retrieval was implemented similar to MUFIN.
Scoring Architecture: Both CLIP and VisualBERT were offered independent instantiations of the cross-attention block $\mathcal{A}_C$. For CLIP, it was applied to the bag embeddings $\mathcal{E}_{\text{CP}}(X_i)$ and $\mathcal{E}_{\text{CP}}(Z_l)$. For VisualBERT that does not offer bag embeddings, cross-attention was applied over $\hat{\boldsymbol{x}}_i^{\text{VB}}$ and $\hat{\boldsymbol{z}}_l^{\text{VB}}$ instead. Both methods were offered one-vs-all classifiers $\boldsymbol{w}_l, l \in [L]$.
Training and Fine-Tuning Details: In the first experiment, the encoding architectures of CLIP and VisualBERT were frozen and only the self-/cross-attention architectures were trained in a four module manner identical to MUFIN. The results of this experiment correspond to the rows titled "VisualBert" and "CLIP" in Tab. 2. In the next experiment, the encoders used within CLIP and VisualBERT were additionally trained in Modules I-IV. The results of this experiment correspond to the rows titled "VisualBert-fine-tuned" and "CLIP-fine-tuned" in Tab. 2. Without fine-tuning the encoders, both architectures offered poor performance 30-50% worse than MUFIN in terms of P@1. Fine-tuning significantly improved the performance for both methods but they remained 5-13% worse than MUFIN in terms of P@1.

Table 5. Hyper-parameters used to train MUFIN on the public datasets. MUFIN uses the AdamW optimizer and learning rate scheduler with a warm start of 1000 iterations.

Dataset	Module I			Module IV		
	Epochs	Learning Rate (lr)	Batch Size (B)	Epochs	Learning Rate (lr)	Batch Size (B)
MM-AmazonTitles-300K	200	2e-4	1024	20	5e-5	200
Polyvore-Disjoint	200	2e-4	1024	10	5e-5	200



(a) Query Datapoint (b) Retrieved results

Figure 8. Predictions by MUFIN for sample test points in the MM-AmazonTitles-300K dataset. $\mathbf{Z}_1$ (resp. $\mathbf{Z}_{-1}$) in the title indicates that the retrieved product (label) was relevant (resp. irrelevant) for the query datapoint in the ground truth. Not only does MUFIN mostly recommend products relevant according to the ground truth, but the occasional recommendation not in the ground truth list is often a good recommendation nevertheless. For instance, in the first row, the last recommendation does not appear in the ground truth but is a chair cover relevant to the query. (Figure best viewed under magnification)

D. Hyperparameters

(Discussion continued from the **Hyperparameters** subsection in Sec. 4)

MUFIN uses a ViT-32 [8] architecture as the image encoder $\mathcal{E}_V$ with 16×16 patches and a SentenceBert [31] architecture as the text encoder $\mathcal{E}_T$. The AdamW optimizer with a one-cycle cosine scheduler with warm start of 1000 iterations was used. MUFIN was trained on a 24-core Intel Skylake 2.4 GHz machine with 4 V100 GPUs for 48 hrs on the A2Q-4M dataset. To reiterate the main training parameters mentioned in Sec. 3, model parameters were learnt in Module I using the contrastive loss with a margin of $\gamma = 0.2$. In Module IV, these encoders were fine-tuned using the cosine embedding loss with a margin of $\gamma = 0.5$ with $|\mathcal{S}_i| = 2$ randomly sampled positive labels and $|\mathcal{T}_i| = 12$ hard negative labels taken from the shortlist R_i obtained from the augmented retrieval step. For details of other hyper-parameters please refer to Tab. 5.

E. Additional Experimental Results

Fig. 8 presents predictions by MUFIN for a few sample test points in the MM-AmazonTitles-300K dataset. Not only does MUFIN mostly recommend products relevant according to the ground truth, but the occasional recommendation not in the ground truth list is often a good recommendation nevertheless.

Tab. 6 presents MUFIN's performance on various categories of the MM-AmaonTitles-300K dataset of which a snapshot was provided in Fig. 3. For most categories, MUFIN is indeed the best method and MUFIN $(\alpha = 1)$ is the second-best method. In particular, MUFIN could give accuracy gains up to 6% on various categories (e.g. Musical Instruments).

F. Ablation

(Discussion continued from **Ablation** Sec. 4.2)

MUFIN makes several design choices with respect to key components in its architecture and training pipeline. its key

Table 6. MUFIN's performance on various categories of the MM-AmaonTitles-300K dataset of which a snapshot was provided in Fig. 3. For each category, the best performance is highlighted in bold black font, the second-best performance is left in normal black font and the third-/fourth- performances are stylized in light gray. Note that for most categories, MUFIN is indeed the best method and MUFIN ($\alpha = 1$) is the second-best method. In particular, MUFIN could give accuracy gains up to 6% on various categories (e.g. Musical Instruments).

	# Labels	MUFIN / MUFIN ($\alpha = 1$)	SiameseXML [5]	VisualBert [19]
Overall	303,296	**34.76** / 33.19	32.99	32.24
AMAZON FASHION	681	**24.71** / 23.85	23.90	22.98
All Beauty	329	**19.70** / 19.04	18.98	18.80
Appliances	767	**27.51** / 27.17	24.60	25.69
Arts Crafts and Sewing	16,843	**41.62** / 39.18	39.06	38.25
Automotive	18,384	**14.39** / 14.12	12.84	13.40
Cell Phones and Accessories	6,410	**33.32** / 32.81	32.52	32.06
Clothing Shoes and Jewelry	25,379	**29.18** / 28.03	28.75	26.80
Electronics	22,449	**32.65** / 31.07	29.92	29.96
Grocery and Gourmet Food	24,676	**47.41** / 44.72	44.94	44.18
Home and Kitchen	37,111	**35.75** / 34.63	33.88	34.19
Industrial and Scientific	7,333	**36.55** / 34.01	33.77	33.19
Luxury Beauty	3,455	60.74 / 58.98	**61.56**	60.72
Musical Instruments	3,835	**26.18** / 24.49	20.82	23.74
Office Products	16,763	**38.09** / 36.33	36.13	35.17
Patio Lawn and Garden	11,631	**36.33** / 35.03	32.84	34.25
Pet Supplies	12,102	**41.76** / 40.03	37.54	38.47
Prime Pantry	3,328	37.45 / 38.78	**40.21**	38.61
Sports and Outdoors	24,842	**34.24** / 31.99	32.30	30.62
Tools and Home Improvement	23,437	**37.03** / 35.03	34.87	34.11
Toys and Games	43,541	**47.92** / 46.13	45.73	45.55

component- sampling, retrieval, representation and ranker. These ablation experiments attempt to ascertain the differential contribution of each of these design choices to MUFIN's final performance. The results of the ablation experiments are detailed in Tab. 4.

Sampling: Recall that in Module I, MUFIN uses hard negative as well as hard positive sampling to focus training on datapoint-label pairs that offer the most prominent gradients. In the MUFIN-no +ve variant, hard positive sampling was replaced with random positives. In the MUFIN-no +ve, -ve variant, both hard positive and hard negative sampling was replaced with random positive and negatives. As Tab. 4 indicates, "MUFIN-no +ve" and "MUFIN-no +ve, -ve" lead to the loss of 2% and 2.5% in P@1 and 1% and 1.5% in P@5. This points to the need for effective training using hard negatives and positives.

Retrieval: Recall that to perform augmented retrieval, MUFIN represents each label using its corresponding bag-of-embeddings of product images as well as text and creating ANNS structures over an expanded set of $\sum_{l \in [L]} m_l$ vectors, the label l getting represented using m_l vectors, one per descriptor. We refer to this default variant as MUFIN-P-I-bag. The alternate variant MUFIN-P-I-vec represents each label using a single embedding namely $\boldsymbol{z}_l^1$. Results shows that MUFIN-P-I-bag could be 1% and 1.6% more accurate in P@1 and P@5 indicating the benefits of augmented retrieval.

Note about the Retrieval ablation experiment: The performance numbers for MUFIN-P-I-bag and MUFIN-P-I-vec given in Tab. 4 are those of the MUFIN model learnt after Module I i.e. sans the cross-attention layer and one-vs-all classifiers. This is why the absolute numbers for MUFIN-P-I-bag (e.g. 42.72 P@1) are lower than MUFIN (52.3 P@1). The difference can be attributed to the inclusion of the cross-attention block and one-vs-all classifiers.

Representation: MUFIN adapts the representation of a datapoint to a label using its cross attention block $\mathcal{A}_C$. In the MUFIN-ConCat variant, the cross attention block was replaced with a simpler architecture that concatenates the datapoint ($\boldsymbol{x}^1$) and label ($\boldsymbol{z}_l^1$) representations and applies two feedforward layers of the form $2D \rightarrow 2D \rightarrow D$ to obtain an alternate label-adapted representation analogous to $\hat{\boldsymbol{x}}^{2,l}$. Results show that MUFIN could be 2.69% and 1.87% more accurate than

MUFIN-ConCat in terms of P@1 and P@5. The MUFIN-no $\mathcal{A}_S$ variant replaces the self-attention block $\mathcal{A}_S$ with a simple feed-forward layer (in Modules I-IV) and observed a loss of 2.3% and 1.6% in terms of P@1 and P@5. The ablations indicate the effectiveness of MUFIN's self- and cross-attention compared to alternatives.

Ranker: MUFIN uses a cross-attention layer and one-vs-all classifiers to perform datapoint-label scoring. In the MUFIN-no $\mathcal{A}_C$ variant, the cross-attention block is completely bypassed, effectively yielding $\hat{\boldsymbol{x}}^{2,l} = \hat{\boldsymbol{x}}^1$ on top of which one-vs-all classifiers are then applied. In the MUFIN-(MUFIN =($\alpha = 1$)) on the other hand, MUFIN completely ignores the one-vs-all classifiers by explicitly setting $\alpha = 1$. Experiments show that MUFIN scoring mechanism employing both a cross-attention block and one-vs-all classifiers can be 2-3% more accurate in P@1 than the MUFIN-no$\mathcal{A}_C$ and MUFIN-($\alpha = 1$) variants.