VISUALLY SIMILAR PRODUCTS RETRIEVAL FOR SHOPSY

Prajit Nadkarni and Narendra Varma Dasararaju

Flipkart Internet Pvt. Ltd, India

ABSTRACT

Visual search is of great assistance in reseller commerce, especially for non-tech savvy users with affinity towards regional languages. Product attributes available in e-commerce have potential for building better visual search systems [2, 20, 29]. We design a visual search system for reseller commerce using a multi-task learning approach and address challenges like image compression, cropping, etc, faced in reseller commerce. Our model consists of three tasks: attribute classification, triplet ranking and variational autoencoder (VAE). We introduce an offline triplet mining technique which utilizes information from multiple attributes to capture relative order within data. This technique displays better performance compared to traditional triplet mining [27] baseline. We compare and report incremental gain achieved by our unified multi-task model over each individual task separately. The efsfectiveness of our method is demonstrated using in-house dataset of images from the Lifestyle business-unit of Flipkart. To efficiently retrieve images in production, we use Approximate Nearest Neighbor (ANN) index.

KEYWORDS

Content based image retrieval, Visual search, Multi-task Learning, Triplet loss, Variational autoencoder.

1. INTRODUCTION

Reseller commerce in India is a continuously growing multi-billion dollar market, which helps resellers utilize social platforms like Facebook and Whatsapp to bring commerce to the Next 500 Million customers. Resellers influence and assist their customers by curating the right products and doing order management, thus building a layer of trust and assistance for users to perform online shopping via social platforms. Shopsy is an app by Flipkart that allows resellers to share products with ease to their end customers and earn money by enabling commerce. Resellers communicate with the end user over social platforms in a way to keep all the Shopsy constructs hidden, thus building their business with no intermediary. Therefore, reseller communication with the end user does not include any product links, but is done entirely using images and written description.

Reseller commerce covers the following use cases: (1) Reseller promotes the products by sharing the images or description of the product with their end-user. The user then shows interest in buying a specific product and shares back the product image. (2) Reseller wants to check the availability of a product that the user found interesting on social media, in the Shopsy catalog. It should be noted that images shared by the users with the reseller may be cropped or contain additional markings to highlight specific aspects of the product. Images also undergo compression while being shared over chat. The reseller may also add their logo on the image to promote their business. Over time as the reseller promotes multiple products to multiple users, it becomes hard to locate the requested product using textual search. It is difficult to describe visual characteristics of a product using words. Searching products using text tends to surface products from head brands and does not guarantee retrieval of the required item. Visual search tackles these limitations as it captures the exact visual patterns of the query image and retrieves the best matched item. Therefore, we built a visual search system for reseller commerce to handle the above mentioned use cases.

In recent years, visual search has been built across many companies including Alibaba's Pailitao [36], Pinterest Flashlight and Lens [13, 34, 35], Google Lens [24], Microsoft's Visual Search [12], etc. These applications demonstrate large scale visual search systems for massive updating data. They focus on the quality of recommendations to improve user engagement. Flipkart catalog also contains millions of products and our primary aim is to assist the reseller in retrieving the exact item. The images in our catalog update upon the introduction of new products. Therefore, we design a system that offers high precision and low latency, while considering the size of our catalog and the update rate.

In this work, we consider products only in the fashion category, as currently the reseller commerce in India is focused on fashion. Recent works in fashion [2, 20, 29] have demonstrated the use of product attributes to build high quality visual embeddings using a combination of attribute classification and triplet ranking loss. We design a multi-task model that learns from three different tasks: attribute-classification, triplet ranking and variational autoencoder. Finally, we highlight our production constraints and build an end to end visual search system for our use case.

Our key contributions can be summarized as follows:

- We build a visual search system for reseller commerce and highlight challenges in this domain like image compression, cropping, scribbling on the image, etc.

- We present a triplet mining technique that uses information from multiple attributes to capture relative order within the data. It gives us twice as good performance as the traditional triplet mining technique that uses a single label/attribute, which we have used as a baseline.

- We build a multi-task model to learn high-quality visual embeddings and attain a 4% incremental gain over the best individual task.

- We highlight the business requirements and infrastructure constraints for our reseller commerce environment, and demonstrate an end to end visual search system that offers high precision and low latency, while considering our catalog size and the data update rate.

We present experiments and choices made for selecting an appropriate Approximate Nearest Neighbor (ANN) index for our production use case.

2. Related Works

Large scale visual search systems have been built across many companies [12, 13, 24, 33–36], demonstrating large scale indexing for massive updating data. There has also been research in domain specific image retrieval systems, designed for fashion products [2, 6, 20, 29, 37]. They leverage the product attribute information available in the e-commerce domain to build high quality visual embeddings. Other works that focus on extracting visual attributes for e-commerce [1, 7, 23] demonstrate multi-class classification techniques. Parekh et al. [23] employ a masking technique to handle missing attribute values, a practical approach when dealing with products across different verticals. We use the same masking technique and build a multi-task learning approach with attribute classification and triplet ranking loss.

Distance metric learning techniques are primarily designed for image retrieval systems, with the seminal works like contrastive-loss [5] and triplet-loss [27]. Triplet loss considers a data point as anchor and associates it with a positive and a negative data point, and constrains the distance of an anchor-positive pair to be smaller than the anchor-negative pair. These methods have evolved over time, with early generations like Schroff et al. [27], where they introduced a semi-hard negative mining approach. This is an online triplet mining technique which computes useful triplets on the fly by sampling hard positive/negatives from within a minibatch. Later, techniques evolved to incorporate information beyond a single triplet like Lifted Structured loss [31], N-Pair loss [30], etc. These losses associate an anchor point with a single positive and multiple negative points, and consider their relative hardness while pushing or pulling these points. The above losses consider the rich data-to-data relations and are able to learn fine-grained relations between them. However, these losses suffer from high training complexity □(□2) or □(□3) where M is the number of data points, thus slow convergence. Recent works like Proxy-NCA [21], Proxy Anchor [16], etc, resolve the above complexity issue by introducing proxies, thus aiding in faster convergence.

In all of the above losses, pair-based or proxy-based, the positives and negatives are chosen based on the class label, ie. Positives are from the same class as anchor and negatives from a different class. For instance, in the face-recognition setting, to ensure enough positives in each mini-batch, Schroff et al. [27] used a mini-batch of 1800 such that around 40 faces are selected per identity per minibatch. In the case of proxy based losses, all proxies are part of the model and are kept in memory. Since each proxy represents a class, it puts a limit on the number of classes. Applying these techniques to e-commerce is challenging, where the possible class labels could be a product-id or a product-vertical (eg. t–shirt, shoe, watch, etc). In e-commerce, we have over millions of products with only 3–4 images per product, that appear on its product page, and the total number of verticals range only in a few hundreds. Choosing the class label as product-id can be too restrictive as there are only a few positives to learn from, and in the proxy based setting it would lead to millions of proxies. Choosing the product-vertical as class label makes the relation between data points too slack and thus we lose the fine grained intra-vertical details (e.g. discriminating one t–shirt pattern from another). Thus, applications in the e-commerce domain resort to using product attributes for mining the triplets.

Ak et al. and others [2, 6], etc, choose triplets such that the anchor and the positive must have the same attribute value whereas the negative is chosen with a different attribute value. For instance, given that the anchor is a 'blue' color, positive can be any image with 'blue' color. Serra at el. [29] use images with noisy tags (e.g. red-sweater, red-tshirt) and use similarity score 'intersection over union' between the tags. They then choose positives that have a similarity score above a threshold and negatives with a score below the threshold. Shankar et al. [28] prepare triplets with three levels (positive, in-class-negative, out-of-class-negative), and use a 'basic image similarity scorer' (e.g. pretrained AlexNet, color-histogram, PatternNet) for selecting candidates across levels. Drawing ideas from the above works, we define an offline triplet mining technique that prepares candidates across multiple levels, such that it captures the relative order within the data.

We sample the candidates under each level based on the percentage of attributes matched.

Another technique that has been used for image retrieval applications is Autoencoder [11, 19]. Autoencoder is a type of artificial neural network where the output is the same as the input. It has an encoder, a decoder and a bottleneck layer in the middle which captures the latent representation of the data. Thus, the bottleneck layer learns the most important characteristics of the image in an unsupervised way. A Variational Autoencoder (VAE) [18] has the same structure as an autoencoder but uses a probabilistic approach to learn the latent representation. Unlike an autoencoder, VAE learns the disentangled embedding representation [10], i.e. where a single latent dimension is affected by only one generative factor and is invariant to changes in other factors. Thus, the underlying embedding spaces have a smooth continuous transformation over a latent dimension. For instance, a latent dimension which captures color variations, arranges the red t-shirt closer to maroon than a green t-shirt. This aspect can be beneficial in retrieving similar products along with the exact match. Sarmiento at el. [26] demonstrated the use of VAE for similar image retrieval of fashion products.

We seek to combine the benefits of attribute-classification, triplet ranking and VAE to design a model for image retrieval. Ren at el. and others [15, 25] have shown performance gain of multi-task models over individual tasks. Kendall at el. and others [4, 15] have explored ideas on balancing multiple loss objectives. In our work, we have taken the naive approach to combining multiple loss objectives and computed the linear sum of the normalized losses for each individual task.

3. DATASET

3.1. Datasets

In this work, we use the in-house dataset of product-images from the Lifestyle business unit of Flipkart. To limit the size of the dataset for training, we consider only products that were ordered in the last one year. It has approximately 10+ million images, across 2+ million products spanning 250+ product verticals. A product has 3 to 4 images on average that are displayed on its product page. Figure-1 shows the distribution of images for top product verticals.

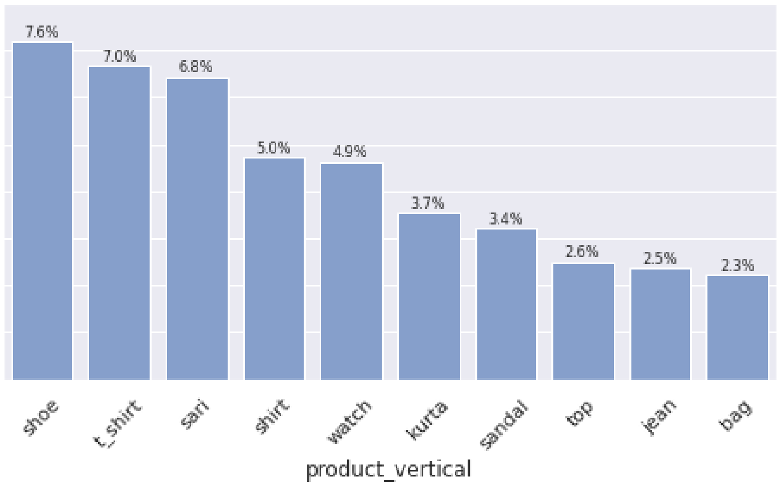


Figure 1. Distribution of top-10 product-verticals in the Lifestyle business-unit (1 year).

Percentage denotes the proportion of vertical images in the dataset. The remaining verticals constitute 53.79% of the dataset.

We create training, indexing, and querying subsets using the above dataset. These include:

- *lifestyle_1y*: Contain images of all products from Lifestyle business-unit ordered in the last 1 year.

- *lifestyle_1y_train* (~1 million images): Random 1M images sampled from *lifestyle_1y* dataset. We split this into train-test-validation sets in the ratio 85:10:5.

- *lifestyle_1y_4v_index* (~3 million images): We use this set to build an index for evaluating the image retrieval task, detailed in Section-5.1. The lifestyle_1y dataset was too large to conduct multiple runs of experiments across different models. Therefore, we prepare this subset with all the images from top 4 verticals. We take 'all the images' from these verticals (and not exclude images from *lifestyle_1y_train*) to ensure that it represents the indexed data in the final production environment.

- *lifestyle_1y_4v_query* (100k images): We use this as a test set to query the index, detailed in Section-5.1. This dataset contains random 100k images from lifestyle_1y_4v_index dataset which are not present in training dataset lifestyle_1y_train. The images in this dataset are augmented using augmentations described in Section-3.3.

3.2. Product Attributes

For each image in the dataset, we assign a total of 12 attributes of its associated product. These are used for attribute classification tasks detailed in Section-4.1.2. We have chosen 8 product-aspect attributes and 4 product-taxonomy attributes. There may be missing values for some of the product attributes, e.g. a shoe will have no value for an attribute sleeve-length. This is a common scenario when we deal with products across different verticals. Detailed list of attributes and their sample values are shown in Table-1.

Table 1. Product Attributes (for the training data of 1M images)

Attribute Name	**Type**	**Total Values**	**Sample Values**
analytic_category	taxonomy	29	WomenWesternCore, WomenEthnicCore, etc
analytic_sub_category	taxonomy	77	WesternWear, MensTShirt, Watch, etc
cms_vertical	taxonomy	147	shoe, sari, t_shirt, watch, shirt, etc
analytic_vertical	taxonomy	300	WomenSari, MensRoundAndOthersTShirt, etc
color	aspect	33	Black, Multicolor, Blue, White, etc
ideal_for	aspect	11	Women, Men, Men & Women, Girls, Boys, etc
material	aspect	27	Polyester, PU, Genuine Leather, etc
occasion	aspect	26	Casual, Sports, Party & Festive, Formal, etc
outer_material	aspect	23	Synthetic, Mesh, Synthetic Leather, etc
pattern	aspect	39	Solid, Printed, Self Design, Embroidered, etc
sleeve	aspect	20	Full Sleeve, Half Sleeve, Short Sleeve, etc
type	aspect	328	Round Neck, Analog, Straight, Fashion, etc

3.3. Image Preprocessing

We apply following augmentations to the images:

- cropping (random 180 × 180 crops on 224 × 224 image)
- color-augmentation (gray-scale, saturation, brightness)
- horizontal-flip
- rotation (0 to 90 degrees)
- overlay a 80 × 80 logo on top of 224 × 224 image
- jpeg-compression (quality: 20–50)

These are done to handle the image modifications introduced while sharing images over chat, with respect to the reseller use cases (Section-1). We also consider the possibilities of other augmentations that may happen in the photo editor like horizontal-flip, rotation, changes in brightness, saturation.

3.4. Triplet Generation

We prepare triplets of images to learn the distance metric using triplet-loss, described in Section-4.1.3. Product-aspect attributes and product-vertical information are used to sample effective triplets to account for relative order in the data. We preprocess the dataset and prepare four candidate-levels for each anchor image, where each level contains a list of images:

- Level-0: list of images belonging to the same product as that of the anchor image.
- Level-1: list of images belonging to products from the same vertical as anchor image, and greater than 80% product-aspect attribute match with the product in anchor image.
- Level-2: list of images belonging to products from the same vertical as anchor image, and less than 80% product-aspect attribute match with the product in anchor image.
- Level-3: list of images belonging to products from a different vertical than that of anchor image.

To limit the complexity of the above process, random sampling is used and the size of the list in each level is limited to 10 (chosen based on training complexity and number of epochs).

During training, first, an anchor-image is randomly sampled from the dataset to generate an image-triplet. A candidate positive image is then sampled by first randomly choosing a level out of 0,1,2 and then randomly choosing an image from the list of images in that level. To generate a negative image, we select the next consecutive level to that of the positively chosen level, and then randomly choose an image from the list of images in that level. For instance, given an anchor image, we sample a positive from level-1 and the negative from the consecutive level-2. We apply augmentations (Section-3.3) to only the anchor image, and no augmentation is applied to the positive and negative images.

3.5. Training Data

For multi-task learning, we sample data by first choosing an image, and then club together corresponding product-attributes and image triplets using the process described in above sections. Figure-2 shows the combined data prepared for the multi-task model.

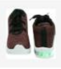
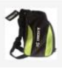
AnchorAug	Anchor	Positive	Negative	analytic_category	analytic_sub_category	cms_vertical	analytic_vertical	color	ideal_for	material	occasion	outer_material	pattern	sleeve	type
				WomenWesternCore	WesternWear	shirt	WomenShirt	Green	Women	-1	Casual	-1	Printed	Short Sleeve	-1
				EyeWear	Sunglass	sunglass	MenSunglass	-1	Men & Women	-1	-1	-1	-1	-1	Rectangular
				WomenEthnicContemporary	EthnicWearBottom	salwar	WomenSalwar	Blue	Women	-1	-1	-1	Solid	-1	-1
				EyeWear	Sunglass	sunglass	MenSunglass	-1	Men & Women	-1	-1	-1	-1	-1	Aviator
				KidsFootwear	KidsUnisexFootwear	kids_sandal	KidsUnisexFootwear	-1	Boys & Girls	-1	-1	-1	-1	-1	Clogs

Figure 2. Random training samples.

Only the anchor image is augmented which is shown in column AnchorAug. Note the similarity of positive and negative images with the anchor image, considering their levels mentioned in Section-3.4. Columns on the right represent attributes of the product in the anchor image, -1 denotes a missing attribute value for that product.

4. METHOD

Our visual search system can be described in two parts: (1) First, a Convolutional Neural Network (CNN) is trained to generate embeddings that capture the notion of visual similarity. (2) Next, an Approximate Nearest Neighbor (ANN) Index is built to return similar images for the given query image embedding. We describe the embedding generation in Section-4.1 and give the details of ANN Index in Section-4.2.

4.1. Embedding Generation

4.1.1. Backbone-CNN

A modified version of Resnet50 [9] is used as our backbone CNN model. We concat the output of conv-block3 with the final output from conv-block-5 as shown in Figure-3. Our intuition is that, out of 5 blocks of convolution, the initial layers learn the fine-grained details (e.g. lines or blobs) while the later layers learn more abstract concepts like object class (e.g. shoe, t-shirt). Thus, the middle layers tend to capture the interesting aspects that may be able to distinguish within the same class of object (e.g. one t-shirt design from the other).

As shown in Figure-3, the Resnet50 takes the input image of size 224×224. We use Global Average Pooling (GAP) to cast 2D conv maps to 1D. The pooled outputs from conv-block-5 and conv-block3 are concatenated to get a 1D vector of length 2560. Further, 3 fully-connected layers (dim: 1024, 1024, 2048) are added with batch normalization and relu activation.

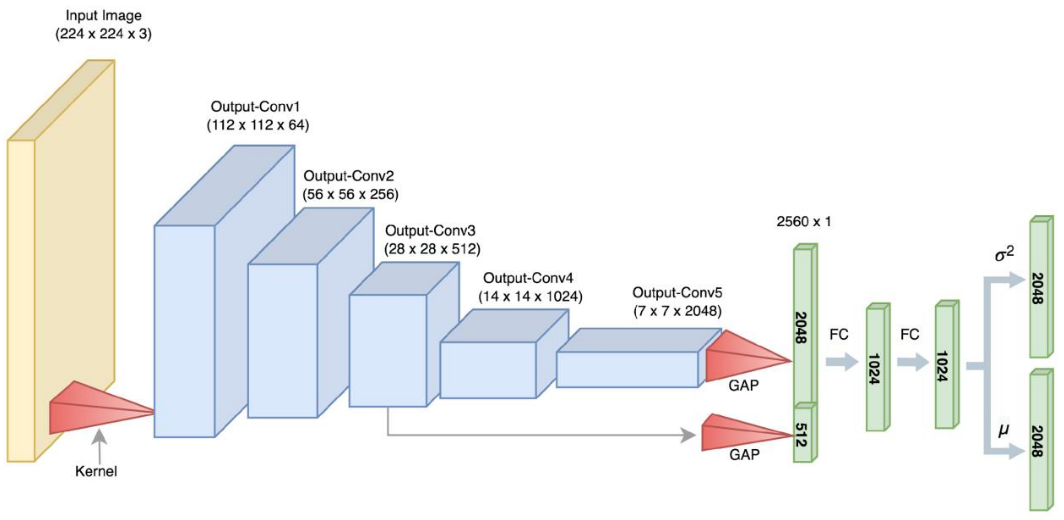


Figure 3. Encoder Network.

We have used modified Resnet50 [9], where we concat the output of conv-block-3 with the final output from conv-block-5

4.1.2. Attribute Classification Loss

We use the embeddings generated by the above model to predict the product attributes. Thus, images with common product attributes will tend to have embedding vectors that are similar to each other. Our data contain multiple attributes that span across multiple product categories, thus not all attributes of a product are assigned values (e.g. sleeve-length is not applicable for shoes). To handle the missing values, we use the masking technique as mentioned in [23], where they mask the loss for the attributes with missing value. The model design for Attribute Classification task is shown in Figure-4.

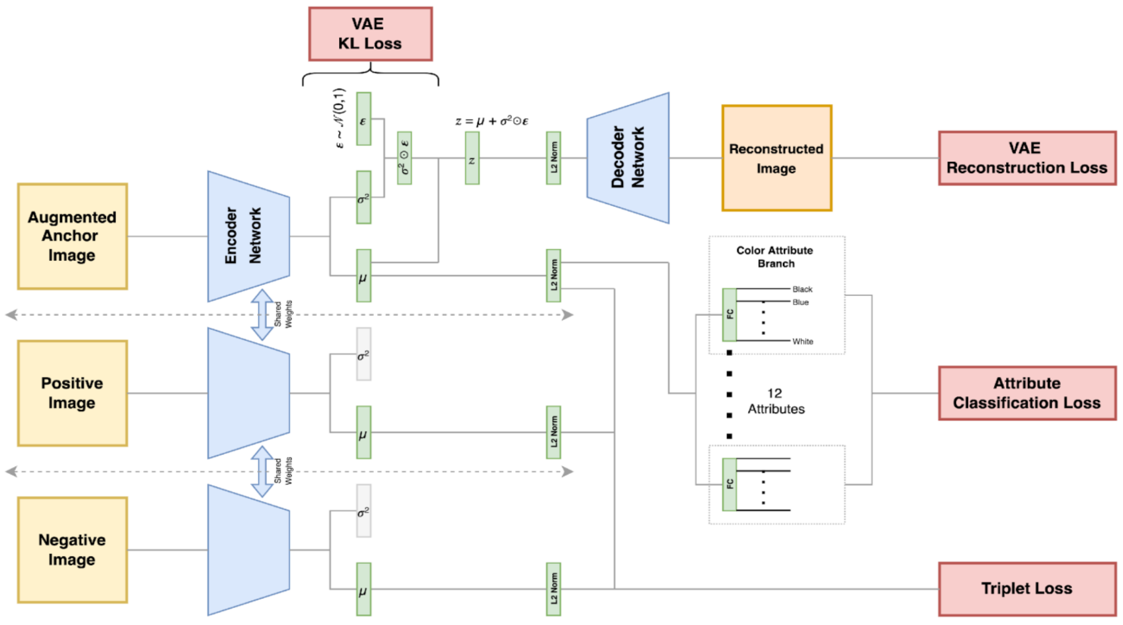


Figure 4. Multi-Task Model

For each attribute, we use multi-class prediction with a single label. 12 product attributes are used as described in Section-3.2. For each attribute branch, the embedding vector is passed to a dropout layer followed by a fully-connected layer with softmax activation. Then, categorical cross-entropy is used to compute loss for each attribute. Here, we mask the loss for an attribute with missing value. Finally, the loss across all attributes is averaged to compute the per sample loss for the attribute-classification task, which is given by:

$$Loss_{att\_clf}(y, \hat{y}) = \frac{1}{A}\sum_{a=1}^{A} mask^a \times CCE(y^a, \hat{y}^a)$$

$$mask^a = \begin{cases} 0, & \text{if } y^a = -1 \ (missing\ value\ for\ attribute\ a) \\ 1, & \text{otherwise} \end{cases}$$

where, A is the number of attributes, CCE is the categorical cross entropy, $y\hat{}^a$ is the predicted label and y^a is the ground truth label for attribute a. For missing values, the ground truth label y^a is set to -1.

4.1.3. Triplet Loss

In this task, we aim to learn the relative order in the data points by using a distance metric. We consider a triplet of images, which are sampled such that the anchor image is more similar to a positive image than a negative image (Section-3.4). Then the embedding vector is learnt for each image, such that the distance between anchor-positive is less than the distance between anchor-negative, in the underlying embedding space. The embedding vectors are L2-normalized, which ensures that the embedding vectors are mapped to the surface of a n-dimensional hypersphere of radius 1. This enables us to use the euclidean distance as the similarity measure between the two vectors, since the euclidean distance over the surface of the unit hypersphere is bound between 0 to 2. We then compute the distance between anchor-positive and anchor-negative, which is then passed to the triplet-loss [27].

During the implementation, we take a triplet of images prepared in Section-3.4, and pass each of the images through the encoder-network, as shown in the Figure-4. The weights of the encoder-network are shared across the three paths. L2-normalization is then applied to the mean-□ layer output of the encoder-network. These L2-normalized image embeddings are used to calculate the squared euclidean distance. The model finally outputs two distances, i.e. distance between anchor-positive and anchor-negative, which are fed to a Hinge Loss function. The final loss from triplet task is given by:

$$Loss_{triplet}(a, p, n) = [d(a, p) - d(a, n) + M]_+$$

where a, p, n are the embedding vectors of anchor, positive and negative images respectively. The function □ (.) is the squared euclidean distance. M is the margin, and $[.]_+$ is the hinge function. We have used a margin of 0.2 (considering that squared euclidean distance ranges between 0 to 4 on the surface of n-dim hypersphere).

We compare our triplet mining approach with triplet semi-hard mining technique [27] as the baseline. The semi-hard technique uses cms_vertical as the class label and samples the positives and negatives from within a mini-batch (size: 256), i.e. positives are chosen from the same class as the anchor and negatives from a different class.

4.1.4. VAE Loss

In a variational autoencoder, there is an encoder, a decoder and a latent bottleneck layer in the middle. VAEs ensure latent embedding layer is normally distributed by using a new layer with parameters mean □ and standard deviation □ of a normal distribution □ (□, □^{2}). During optimization, the normal distribution □ (□, □^{2}) is forced to be as close as possible to reference standard normal distribution □ (0, 1) using Kullback-Leibler (KL-divergence). In the process of reconstruction, the latent bottleneck layer learns the salient features of the image whilst the KL-divergence ensures the disentanglement [10] within the learnt embedding space. Thus, independent latent units become sensitive to latent factors such as color, pattern, object-shape. And the embeddings learn a smooth continuous transformation over the values of these latent factors. For instance, all the t-shirts could lie in one dimension with gradual transition in their color. Our intuition is that this would cause the images in the embedding space to be arranged such that images with similar looking attributes lie near each other with gradual transition in the attribute value. Therefore, when the images are retrieved using nearest-neighbor algorithm, they are ordered with exact match followed by only a slight variation in the object characteristics.

Our model design for variational autoencoder is shown in Figure-4. Here, the encoder-network outputs the mean(□) and log variance(log □) for a given input image. It is followed by a sampling layer which returns □ = □ + □^{2}·□, □ ~ □ (0, 1). The latent embedding is then L2-normalized, because downstream ANN indices work well if the vectors are L2-normalized. This is then passed to decoder-CNN. Our decoder-CNN architecture uses transposed-convolutions with batch-normalization and relu activations. The last layer uses sigmoid activation to output the pixel values of the reconstructed image. We have used the approach from [22] to design the kernels for our decoder-CNN. This helps to reduce the checkerboard artifacts that appear in the reconstructed image. The end to end model takes an augmented image (Section-3.3) as input, and it outputs the reconstructed image of exactly the same dimensions as the input image.

The loss objective for VAE-Task is two fold. (1) For the reconstruction loss, we use binary cross-entropy [17] between the original non-augmented input image and the reconstructed output image. (2) The KL-divergence loss [32] between the encoder distribution □(□|□) = □ (□|□(□), Σ(□)) □ℎ□□□ Σ = □□□□(□$^2_1$, . . . , □$^2_□$) and the prior standard normal distribution □ (0, 1).

$$Loss_{recon}(x,\hat{x}) = \sum_{i=1}^{n} BCE(x_i,\hat{x}_i)$$

$$Loss_{kl}(\mu,\sigma) = \frac{1}{2}\left[-\sum_{i=1}^{z}\left(\log\sigma_i^2+1\right)+\sum_{i=1}^{z}\sigma_i^2+\sum_{i=1}^{z}\mu_i^2\right]$$

where BCE is binary cross-entropy, □, x̂ are the input and the reconstructed images respectively, □ = 224 × 224 × 3 is total image dimensions. □, □ are the output of mean and variance layers respectively as shown in Figure-4, □ = 2048 is the number of latent dimensions. In practice, we output log-variance instead of the variance for numerical stability.

We normalize each of the above loss values across their dimensions, and add them to get the final per sample loss for the VAE-Task.

$$Loss_{vae} = \frac{1}{n}Loss_{recon} + \frac{1}{z}Loss_{kl}$$

4.1.5. Multi Task Loss

Multi-Task learning aims to learn a single embedding representation of the given input image, which then outputs multiple values corresponding to different tasks. The overall setup for multi-task learning is shown in Figure-4. The model predicts multiple outputs against each of the tasks: (1) For attribute classification task, it outputs softmax-probabilities of attribute-values against each of the 12 product-attributes using the augmented anchor-image embedding. (2) For the triplet ranking task, it outputs two distances: distance between, and the distance between. (3) For VAE task, it takes the augmented-anchor-image as input, and outputs the reconstructed image with the same dimensions as the input image. We add the loss from each of the above tasks to construct the final per sample loss:

$$Loss_{multi\_task} = Loss_{att\_clf} + Loss_{triplet} + Loss_{vae}$$

4.1.6. Image Embedding

To generate the embedding for a given query image using the above multi-task model, we take the output from mean-layer (□) and apply the L2-normalization.

4.2. ANN-Index

The only way to guarantee retrieval of exact nearest neighbors in n-dimensional space is exhaustive search, which is not practical to be used at query time in production. Thus, we resort to Approximate Nearest Neighbor (ANN) technique, which speeds up the search by preprocessing the data into efficient indices. There are many readily available libraries that implement ANN Indices e.g. Faiss [14], ScaNN [8], Annoy [3], etc. Our primary aim is to assist the reseller in finding the exact item, thus we focus on high precision, followed by a throughput (queries-per-second) which ranges in few thousands. The total number of images in our system ranges in a few millions, and are updated only upon the introduction of new products in the catalog. Based on this business use case, we conducted exhaustive grid-search over multiple ANN indices, and found ScaNN [8] and HNSW [14] to work the best for our production use case. The detailed results of our experiments are presented in Section-5.3.

Given the embeddings generated in the previous section (Section-4.1.6), we first use PCA to reduce the embedding dimensionality from 2048d to 256d. An ANN index is then built using the embeddings of all the images in our database. When a user searches using an image, the ANN index is queried using the image embedding and the top-k nearest neighbor images are retrieved.

4.3. Implementation

We train our model using a batch-size of 32. Adam(□1 : .9, □2 : .999) is used as an optimizer with a learning-rate of 10−3 for initial 100 epochs and then reduced to 10−5 till convergence. We end our training using early-stopping with patience of 10 epochs. Model is trained on a single Tesla V100 GPU for a total of 148 epochs.

4.4. Evaluation

We evaluate our model on the image retrieval task, and use precision@k to measure the success. Precision@k returns the number of relevant results among the top k retrieved items. In production, we consider a query as successfully resolved if the exact item is present in the top k retrieved items, thus prec@k correctly measures our success. The detailed results are presented in Section-5.1.

5. EXPERIMENTS

5.1. Image Retrieval Task

We set up the image-retrieval task such that it replicates the process in our production environment. In production, when a user searches using an image, the top-k items are retrieved from the ANNindex. We prepare two datasets, index-dataset and query-dataset. The lifestyle_1y_4v_index (Section-3.1) is used as an index-dataset which represents the images in our production database. The lifestyle_1y_4v_query is used as the query dataset which contains the augmented images and represent user queries that we may expect in our live system.

First, we build the ANN index with the index-dataset using the process described in Section-4.2. Next, for each image in the querydataset, top-k items are retrieved from the ANN-Index. If the query image-id is present in the top-k retrieved items, it is marked as success (1) else as a failure (0). Finally, we average over all the samples to compute precision@k.

We build separate models using each of the individual tasks (Attribute Classification, Triplet Loss, Triplet Semi-Hard, VAE) and compare it with our combined multi-task model and report the incremental gains achieved by multi-task technique. As shown in Figure-5, the unified multi-task model performs better than the model learnt using each of the individual tasks. Also, we see that our triplet mining technique performs twice as better than the online triplet semi-hard mining [27] which uses cms_vertical as the class label. In Table-2, we compare the model performance for each individual image augmentation and display the results for prec@k=4. We see 4% gain on average with the unified multi-task model over the best individual task performance.

Table 2. Retrieval Score (prec@4) per augmentation-type. For models (T-SH: triplet semi-hard, A: attribute-classification, T: triplet-loss, V: VAE, MT: multi-task)

Augmentation	**T-SH**	**A**	**T**	**V**	**MT**
all_augmentation	0.02	0.58	0.50	0.10	0.64
compression	0.83	0.97	0.98	0.98	0.97
crop	0.12	0.85	0.85	0.20	0.89
hor_flip	0.20	0.91	0.95	0.93	0.95
no_augmentation	0.98	1.00	1.00	1.00	1.00
logo_overlay	0.62	0.97	0.98	0.95	0.98
rotation	0.18	0.89	0.85	0.77	0.93
average	0.42	0.88	0.87	0.70	**0.91**

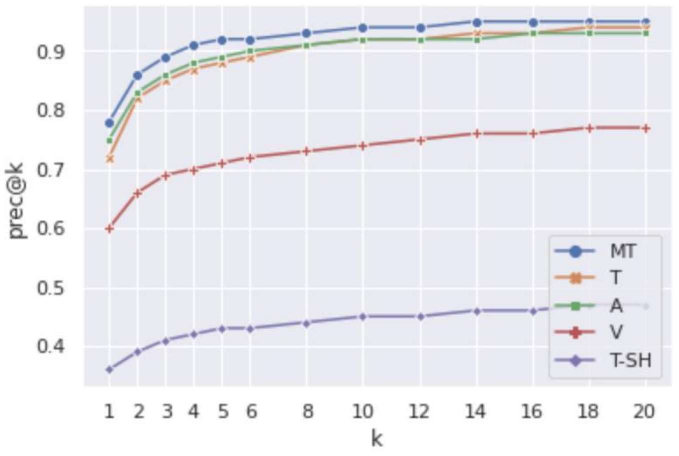


Figure 5. Image retrieval performance of models (T-SH: triplet semi-hard, A: attribute-classification, T: triplet-loss, V: VAE, MT: multi-task).

We measure precision@k, where k is the top retrieved images from the ANN index.

5.2. Results of Individual Task

We also present the performance of each individual task within our multi-task model. Table-4 displays the accuracy for the attribute prediction task against each individual attribute. Our final triplet loss is 0.052 where 77.9% of samples are correctly ordered with zero triplet-loss. For the VAE task, the final reconstruction loss is 0.477 and KL-divergence loss is 0.013. Sample predictions for each task of the multi-task model are shown in Appendix-A.

Table 3. Attribute Classification Accuracy

Attribute Name	**Accuracy**
analytic_category	0.828
analytic_sub_category	0.756
cms_vertical	0.762
analytic_vertical	0.646
color	0.542
ideal_for	0.852
material	0.543
occasion	0.703
outer_material	0.407
pattern	0.625
sleeve	0.697
type	0.555

5.3. Selection of ANN Index

Our goal is to choose the Approximate Nearest Neighbor (ANN) index that best suits our business use case, outlined in Section-4.2. Our production environment, detailed in Section-6, demands an index that offers high precision, high QPS (queries-per-second), low memory footprint and preferably a short build time. Considering these requirements, we experimented with different indices (Table3), and found ScaNN [8] and HNSW [14] to be well suited for our production use case. We also experimented with quantization techniques like IVF(Inverted File Index), PQ(Product Quantization) and SQ(Scalar Quantization), if supported in the index library.
For this analysis, we build the index using lifestyle_1y_4v_index dataset, and use lifestyle_1y_4v_query as the query dataset. We chose to have only crop-augmented images since it represents the most common use case in our system. It also provides consistency in the image distribution while comparing multiple ANN Indices. We test around 3–4 values of each hyper-parameter using a grid search, and list the best configuration for each index in the Table-4. All indices are constrained to run on a single cpu-core. We can see from Table-4, the top performing index(ScaNN, HNSW) experienced no drop in performance compared with exhaustive search (FlatL2). Further, the top index offers a QPS of ~ 1k per cpu core and very modest memory footprint (~ 3GB / 3M images) which perfectly fits our production use case.

Table 4. ANN Index performance, ordered by precision then QPS(queries per second). The index uses a single cpu-core, and was built with a total of 3 million images.

Index	Precision@4	QPS	Build Time (sec)	Size(GB)
ScaNN	0.87	999.19	79	3.37
HNSW_Flat	0.87	730.19	313	3.40
HNSW_SQ	0.87	679.08	565	1.16
IVF_Flat	0.87	102.28	335	3.01
IVF_SQ	0.87	63.23	383	0.78
Annoy	0.87	62.25	2354	6.43
FlatL2	0.87	1.19	4	2.98
HNSW_PQ	0.84	1388.29	287	0.61
HNSW_2Level	0.80	786.88	464	0.52

6. Production Environment

Considering the scale of reseller commerce in India and the available product discovery features (text search, content landing pages) in Shopsy, we estimate our visual search QPS to range in a few thousands during the peak business hours. Since our primary aim is to assist the reseller in locating the exact product, we focus on retrieving the images with high precision. Further, to ease the implementation process, we want to ensure our index fits on a single node (not distributed), thus limiting the memory footprint of the ANN index to be less than ~ 80GB. The images in our system update upon introduction of new products in our catalog, which is not a frequent activity. But we prefer a short index build time because it adds convenience to the index updation process. Our top ANN index (Table-4) is able to satisfactorily address all the above constraints with minimal infrastructure demands. The embedding generation model (Section-4.1.6) is able to process 9 QPS on a CPU core and 14 QPS on a GPU (since this is a single image and not batched). Thus, we aim to fulfill the business requirements by scaling the number of nodes (e.g. 5 CPU instances of 20 cores each, can support around a thousand QPS). Although we solve for a number of use cases like cropping, compression, etc, of catalog images, another important use case is 'images in the wild', i.e. photos our users take with their cameras, which we would like to solve before we deploy the entire solution to production.

7. CONCLUSION

In this work, we build an end to end visual search system for reseller commerce, which helps to retrieve products more accurately compared to text search. We describe the evolution of distance metric learning over time and application of these methods to the e-commerce domain. We introduce an offline triplet mining technique which captures relative order within the data.

Comparing our approach with the triplet semi-hard mining technique, which uses product-vertical as class label, shows twice as good performance as the latter. This indicates that attribute information plays a significant role when mining triplets for e-commerce. Further, we combine the benefits of multiple image retrieval approaches using multi-task learning and achieve a 4% gain over the best individual learning approach. Finally, we highlight our business requirements and production environment constraints, and present the experiments conducted to select an ANN index that best suits our use case.

REFERENCES

[1] Sandeep Singh Adhikari, Sukhneer Singh, Anoop Rajagopal, and Aruna Rajan. 2019. Progressive Fashion Attribute Extraction. arXiv:1907.00157 [cs.LG]

[2] Kenan E. Ak, Joo Hwee Lim, Ying Sun, Jo Yew Tham, and Ashraf A. Kassim. 2021. FashionSearchNet-v2: Learning Attribute Representations with Localization for Image Retrieval with Attribute Manipulation. arXiv:2111.14145 [cs.CV]

[3] Erik Bernhardsson. 2018. Annoy: Approximate Nearest Neighbors in C++/Python. https://pypi.org/project/annoy/ Python package version 1.13.0.

[4] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks. arXiv:1711.02257 [cs.CV]

[5] S. Chopra, R. Hadsell, and Y. LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). 539–546 vol. 1. https://doi.org/10.1109/CVPR.2005.202

[6] Antonio D'Innocente, Nikhil Garg, Yuan Zhang, Loris Bazzani, and Michael Donoser. 2021. Localized Triplet Loss for Fine-grained Fashion Image Retrieval. In 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). 3905–3910.

[7] Beatriz Quintino Ferreira, Luís Baía, João Faria, and Ricardo Gamelas Sousa. 2018. A Unified Model with Structured Output for Fashion Images Classification. arXiv:1806.09445 [cs.CV]

[8] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating Large-Scale Inference with Anisotropic Vector Quantization. In International Conference on Machine Learning. https: //arxiv.org/abs/1908.10396

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 770–778. https://doi.org/10.1109/CVPR.2016.90

[10] Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. betaVAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In ICLR.

[11] Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. science 313, 5786 (2006), 504–507.

[12] Houdong Hu, Yan Wang, Linjun Yang, Pavel Komlev, Li Huang, Xi (Stephen) Chen, Jiapei Huang, Ye Wu, Meenaz Merchant, and Arun Sacheti. 2018. WebScale Responsive Visual Search at Bing. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery &; Data Mining (London, United Kingdom) (KDD '18). Association for Computing Machinery, New York, NY, USA, 359–367. https://doi.org/10.1145/3219819.3219843

[13] Yushi Jing, David Liu, Dmitry Kislyuk, Andrew Zhai, Jiajing Xu, Jeff Donahue, and Sarah Tavel. 2015. Visual Search at Pinterest. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (Sydney, NSW, Australia) (KDD '15). Association for Computing Machinery, New York, NY, USA, 1889–1898. https://doi.org/10.1145/2783258.278862

[14] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with GPUs. arXiv preprint arXiv:1702.08734 (2017).

[15] Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. arXiv:1705.07115 [cs.CV]

[16] Sungyeon Kim, Dongwon Kim, Minsu Cho, and Suha Kwak. 2020. Proxy Anchor Loss for Deep Metric Learning. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 3235–3244. https://doi.org/10.1109/CVPR42600. 2020.00330

[17] Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, and Max Welling. 2014. Semi-Supervised Learning with Deep Generative Models. In Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2 (Montreal, Canada) (NIPS'14). MIT Press, Cambridge, MA, USA, 3581–3589.

[18] Diederik P Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. arXiv:1312.6114 [stat.ML]

[19] Alex Krizhevsky and Geoffrey E. Hinton. 2011. Using very deep autoencoders for content-based image retrieval. In ESANN.

[20] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. 2016. DeepFashion: Powering Robust Clothes Recognition and Retrieval With Rich Annotations. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[21] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh. 2017. No Fuss Distance Metric Learning Using Proxies. In 2017 IEEE International Conference on Computer Vision (ICCV). IEEE Computer Society, Los Alamitos, CA, USA, 360–368. https://doi.org/10.1109/ICCV.2017.47

[22] Augustus Odena, Vincent Dumoulin, and Chris Olah. 2016. Deconvolution and Checkerboard Artifacts. Distill (2016). https://doi.org/10.23915/distill.0000

[23] Viral Parekh, Karimulla Shaik, Soma Biswas, and Muthusamy Chelliah. 2021. Fine-Grained Visual Attribute Extraction From Fashion Wear. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops. 3973–3977.

[24] Rajan Patel. 2018. Google Lens: real-time answers to questions about the world around you. Retrieved Jan 11, 2022 from https://blog.google/products/googlelens/google-lens-real-time-answers-questions-about-world-around-you/

[25] Zhongzheng Ren and Yong Jae Lee. 2017. Cross-Domain Self-supervised Multitask Feature Learning using Synthetic Imagery. arXiv:1711.09082 [cs.CV]

[26] James-Andrew Sarmiento. 2020. Exploiting Latent Codes: Interactive Fashion Product Generation, Similar Image Retrieval, and Cross-Category Recommendation using Variational Autoencoders. arXiv:2009.01053 [cs.CV]

[27] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A unified embedding for face recognition and clustering. In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 815–823. https://doi.org/10. 1109/CVPR.2015.7298682

[28] Devashish Shankar, Sujay Narumanchi, H A Ananya, Pramod Kompalli, and Krishnendu Chaudhury. 2017. Deep Learning based Large Scale Visual Recommendation and Search for E-Commerce. arXiv:1703.02344 [cs.CV]

[29] Edgar Simo-Serra and Hiroshi Ishikawa. 2016. Fashion Style in 128 Floats: Joint Ranking and Classification Using Weak Data for Feature Extraction. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 298–307. https://doi.org/10.1109/CVPR.2016.39

[30] Kihyuk Sohn. 2016. Improved Deep Metric Learning with Multi-class N-pair Loss Objective. In Advances in Neural Information Processing Systems, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.), Vol. 29. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2016/file/6b180037abbebea991d8b1232f8a8ca9-Paper.pdf

[31] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. 2016. Deep Metric Learning via Lifted Structured Feature Embedding. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[32] user3658307 (https://stats.stackexchange.com/users/128284/user3658307). [n. d.]. Deriving the KL divergence loss for VAEs. Cross Validated. arXiv:https://stats.stackexchange.com/q/370048 https://stats.stackexchange. com/q/370048 URL:https://stats.stackexchange.com/q/370048 (version: 2020-03-03).

[33] Fan Yang, Ajinkya Kale, Yury Bubnov, Leon Stein, Qiaosong Wang, Hadi Kiapour, and Robinson Piramuthu. 2017. Visual Search at EBay. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (Halifax, NS, Canada) (KDD '17). Association for Computing Machinery, New York, NY, USA, 2101–2110. https://doi.org/10.1145/3097983.3098162

[34] Andrew Zhai, Dmitry Kislyuk, Yushi Jing, Michael Feng, Eric Tzeng, Jeff Donahue, Yue Li Du, and Trevor Darrell. 2017. Visual Discovery at Pinterest. In Proceedings of the 26th International Conference on World Wide Web Companion (Perth, Australia) (WWW '17 Companion). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 515–524. https://doi.org/10.1145/3041021.3054201

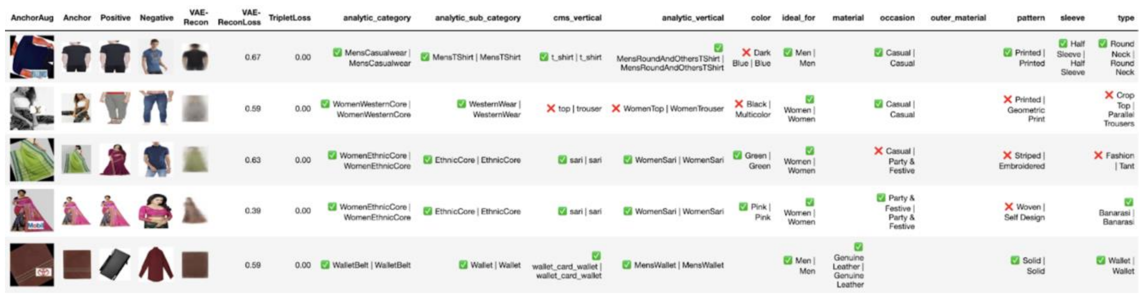
[35] Andrew Zhai, Hao-Yu Wu, Eric Tzeng, Dong Park, and Charles Rosenberg. 2019. Learning a Unified Embedding for Visual Search at Pinterest. 2412–2420. https: //doi.org/10.1145/3292500.3330739

[36] Yanhao Zhang, Pan Pan, Yun Zheng, Kang Zhao, Yingya Zhang, Xiaofeng Ren, and Rong Jin. 2018. Visual Search at Alibaba. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery &; Data Mining (London, United Kingdom) (KDD '18). Association for Computing Machinery, New York, NY, USA, 993–1001. https://doi.org/10.1145/3219819.3219820

[37] Bo Zhao, Jiashi Feng, Xiao Wu, and Shuicheng Yan. 2017. Memory-Augmented Attribute Manipulation Networks for Interactive Fashion Search. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 6156–6164. https: //doi.org/10.1109/CVPR.2017.652

A. MULTI-TASK MODEL PREDICTIONS

Sample predictions for each task of the multi-task model are shown in Figure-6.



AnchorAug	Anchor	Positive	Negative	VAE-Recon	VAE-ReconLoss	TripletLoss	analytic_category	analytic_sub_category	cms_vertical	analytic_vertical	color	ideal_for	material	occasion	outer_material	pattern	sleeve	type
					0.67	0.00	✓ MensCasualwear | MensCasualwear	✓ MensTShirt | MensTShirt	✓ t_shirt | t_shirt	✓ MensRoundAndOthersTShirt | MensRoundAndOthersTShirt	✗ Dark Blue | Blue	✓ Men | Men		✓ Casual | Casual		✓ Printed | Printed	✓ Half Sleeve | Half Sleeve	✓ Round Neck | Round Neck
					0.59	0.00	✓ WomenWesternCore | WomenWesternCore	✓ WesternWear | WesternWear	✗ top | trouser	✗ WomenTop | WomenTrouser	✗ Black | Multicolor	✓ Women | Women		✓ Casual | Casual		✗ Printed | Geometric Print		✗ Crop Top | Parallel Trousers
					0.63	0.00	✓ WomenEthnicCore | WomenEthnicCore	✓ EthnicCore | EthnicCore	✓ sari | sari	✓ WomenSari | WomenSari	✓ Green | Green	✓ Women | Women		✗ Casual | Party & Festive		✗ Striped | Embroidered		✗ Fashion | Tant
					0.39	0.00	✓ WomenEthnicCore | WomenEthnicCore	✓ EthnicCore | EthnicCore	✓ sari | sari	✓ WomenSari | WomenSari	✓ Pink | Pink	✓ Women | Women		✓ Party & Festive | Party & Festive		✗ Woven | Self Design		✓ Banarasi | Banarasi
					0.59	0.00	✓ WalletBelt | WalletBelt	✓ Wallet | Wallet	✓ wallet_card_wallet | wallet_card_wallet	✓ MensWallet | MensWallet		✓ Men | Men	✓ Genuine Leather | Genuine Leather			✓ Solid | Solid		✓ Wallet | Wallet

Figure 6. Sample predictions for each task of the multi-task model.

Only the anchor image is augmented which is shown in column AnchorAug. It is followed by anchor, positive and negative images sampled using triplet mining technique (Section-3.4). Column VAE-Recon shows the reconstructed anchor image (output of the decoder as shown in Figure-4). Columns VAEReconLoss and TripletLoss show the loss values of the respective row. Columns on the right represent each of the product attributes, and the values are in the format <predicted-value>|<ground-truth-value>. The green check highlights that attribute value is correctly predicted, while a red cross indicates that the prediction is incorrect. A blank cell indicates a missing attribute value for the product.

AUTHORS

Prajit Nadkarni is a data scientist at Flipkart Internet Pvt. Ltd. He received his B.Tech. in Computer Science and Engineering from Vellore Institute of Technology, Vellore, Tamil Nadu in 2013. His research interests include Machine Learning and Computer Vision.

Narendra Varma Dasararaju led Machine learning initiatives for a few key programs in Flipkart, namely: Search, User retention, social commerce, ugc etc. Narendra has more than 15 years of work experience in the field of machine learning / data sciences. Before Flipkart Narendra worked at 2 Big Tech companies and 2 investment banks.