A Tale of Color Variants: Representation and Self-Supervised Learning in Fashion E-commerce

Ujjal Kr Dutta, Sandeep Repakula, Maulik Parmar, Abhinav Ravi
Data Sciences-Image Sciences, Myntra
{ujjal.dutta,sandeep.r,abhinav.ravi}@myntra.com, maulik2087@gmail.com

Abstract

In this paper, we address a crucial problem in fashion e-commerce (with respect to customer experience, as well as revenue): color variants identification, i.e., identifying fashion products that match exactly in their design (or style), but only to differ in their color. We propose a generic framework, that leverages deep visual Representation Learning at its heart, to address this problem for our fashion e-commerce platform. Our framework could be trained with supervisory signals in the form of triplets, that are obtained manually. However, it is infeasible to obtain manual annotations for the entire huge collection of data usually present in fashion e-commerce platforms, such as ours, while capturing all the difficult corner cases. But, to our rescue, interestingly we observed that this crucial problem in fashion e-commerce could also be solved by simple color jitter based image augmentation, that recently became widely popular in the contrastive Self-Supervised Learning (SSL) literature, that seeks to learn visual representations without using manual labels. This naturally led to a question in our mind: Could we leverage SSL in our use-case, and still obtain comparable performance to our supervised framework? The answer is, Yes! because, color variant fashion objects are nothing but manifestations of a style, in different colors, and a model trained to be invariant to the color (with, or without supervision), should be able to recognize this! This is what the paper further demonstrates, both qualitatively, and quantitatively, while evaluating a couple of state-of-the-art SSL techniques, and also proposing a novel method.

Introduction

In this paper, we address a very crucial problem in fashion e-commerce, namely, automated *color variants identification*, i.e., identifying fashion products that match exactly in their design (or style), but only to differ in their color (Figure 1). Our motivation to pick the use-case of color variants identification for fashion products comes from the following reasons: i) Fashion products top across all categories in online retail sales (Jagadeesh et al. 2014), ii) Most often users hesitate to buy a fashion product solely due to its color despite liking all other aspects of it. Providing more color options increases add-to-cart ratio, thereby generating more revenue, along with improved customer experience.

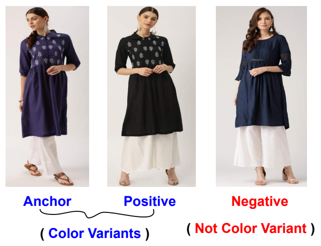


Figure 1: Illustration of the *color variants identification* problem. The images belong to www.myntra.com.

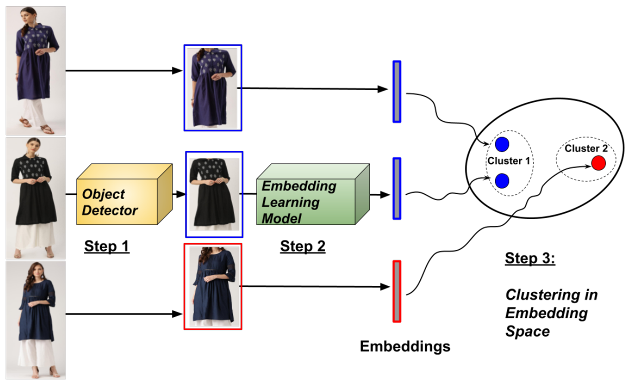


Figure 2: Proposed framework to address color variants identification in fashion e-commerce.

At Myntra (www.myntra.com), a leading fashion e-commerce platform, we address this problem by leveraging deep visual Representation/ Embedding Learning. Our proposed framework, as shown in Figure 2, is generic in nature, consisting of an Embedding Learning model (Step 2), which is trained to obtain embeddings (i.e., representation of visual data in a vector space), in such a way that embeddings of color variant images are grouped together. Having obtained the embeddings, we can perform clustering (Step 3) on them to obtain the color variants (embeddings falling into the same cluster would correspond to images that are color variants). Also, as a typical Product Display Page (PDP) image in a fashion e-commerce platform usually contains multiple fashion products, we apply an object detector to localise our *primary* fashion product of interest, as a preprocessing step (Step 1).

It should be noted that the *main component in the pipeline is the Representation/ Embedding Learning model* in Step 2, and this is what we mainly focus in our paper. We discuss different strategies of training the Embedding Learning component. One way is to obtain manual annotations (class labels) indicating whether two product images are color variants to each other. These class labels are used to obtain triplet based constraints (details to be introduced later), consisting of an anchor, a positive and a negative (Figure 1).

Despite performing well in practice, a key challenge in the real-world setting of fashion e-commerce is the large scale of data to be annotated, which often becomes infeasible, and tedious. *Could we somehow get away with this annotation step ?*, this is what we asked ourselves. Interestingly, we noted that color variants of fashion products are in essence, manifestations of the same fashion *stlye/ design* after applying color jittering, which is a widely popular image augmentation technique applied in visual Self-Supervised Learning (SSL). SSL finds image embeddings without requiring manual annotations! Thus, we go one step further, and try to answer the question of whether SSL could be employed in our use-case, for achieving comparable performance with that of a supervised model using manually labeled examples. For this, we evaluated a number of State-Of-The-Art (SOTA) SSL techniques from the recent literature, but found certain limitations in them, for our use-case. To address this, we make certain crafty modifications, and propose a novel SSL method as well.

Following are the **major contributions of the paper**:

1. A generic visual Representation Learning based framework to identify color variants among fashion products (to the best of our knowledge, studied as a research paper for the first time).
2. A systematic study of a supervised method (with manual annotations), as well as existing SOTA SSL methods to train the embedding learning model.
3. A novel contrastive loss based SSL method that focuses on parts of an object to identify color variants.

Related Work: The problem of visual embedding/ metric learning refers to that of obtaining vector representations/embeddings of images in a way that the embeddings of similar images are grouped together while moving away dissimilar ones. Several supervised (Sun et al. 2020; Gu and Ko 2020) and unsupervised (Dutta, Harandi, and Sekhar 2020a; Cao, Chen, and Lim 2019; Li, Kan, and He 2020; Dutta, Harandi, and Sekhar 2020b) approaches have been proposed in the recent literature. Contrastive learning (Chen et al. 2020; He et al. 2020; Grill et al. 2020; Chen and He 2021), a paradigm of visual SSL (Jing and Tian 2020), groups together embeddings obtained from augmentations of the same image, without making use of manual annotations, as needed in Supervised approaches.

Proposed Framework

Our proposed framework to identify color variants, as already introduced in Figure 2, consists of the stages: i) Object Detection, ii) Embedding Learning and iii) Clustering. As the original input image usually consists of a human model wearing secondary fashion products as well, we perform object detection (using any off-the shelf object detector) to localise the primary fashion product of interest. The core component of our framework, i.e., the Embedding Learning model, can either be trained using a supervised method (when manual annotations are available), or a SSL method. The latter is useful when obtaining manual annotations is infeasible, at large scale.

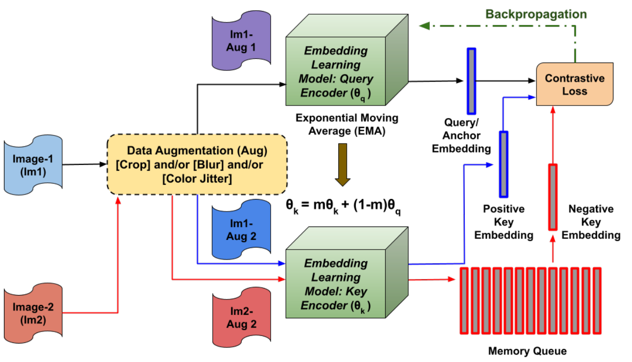

Backpropagation
Im1- Aug 1
Embedding Learning Model: Query Encoder (θ_q)
Contrastive Loss
Exponential Moving Average (EMA)
Query/ Anchor Embedding
Image-1 (Im1)
Data Augmentation (Aug) [Crop] and/or [Blur] and/or [Color Jitter]
$\theta_k = m\theta_k + (1-m)\theta_q$
Im1- Aug 2
Positive Key Embedding
Negative Key Embedding
Embedding Learning Model: Key Encoder (θ_k)
Image-2 (Im2)
Im2- Aug 2
Memory Queue

Figure 3: Contrastive SSL based Embedding Model.

We first discuss the supervised method, where we require a set of triplet images obtained from manually annotated examples (consisting of an anchor, positive, and negative, as shown in Figure 1). The positive (p) is usually an image that consists of a product that is a color variant of the product contained in the anchor (a) image. The negative (n) is an image that consists of a product that is not a color variant of the products contained in the anchor and positive images. Let, the embeddings for the triplet of images be denoted as $(\boldsymbol{x}_a, \boldsymbol{x}_p, \boldsymbol{x}_n)$. These triplets are used to train a supervised triplet loss based deep neural network model (Schroff, Kalenichenko, and Philbin 2015), the objective of which is to bring the embeddings $\boldsymbol{x}_a$ and $\boldsymbol{x}_p$ closer, while moving away $\boldsymbol{x}_n$. This is achieved by minimizing the following:

$$\mathcal{L}_{triplet} = max(0, \lambda + \delta^2(\boldsymbol{x}_a, \boldsymbol{x}_p) - \delta^2(\boldsymbol{x}_a, \boldsymbol{x}_n)), \quad (1)$$

such that $\delta^2(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2$ denotes the squared Euclidean distance between the pair of examples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, with $\|\boldsymbol{x}_i\|_2^2$ being the squared l_2 norm of $\boldsymbol{x}_i$, and $\lambda > 0$ being a margin.

An appropriate clustering algorithm could be applied on the embeddings, such that an obtained cluster contains embeddings of images of products that are color variants to each other. However, the supervised model needs manual annotations which may be infeasible to obtain in large real-world datasets (such as those present in our platform). Thus, we explore SSL to identify color variants.

SSL Based Embedding Learning Model

SSL seeks to learn visual embeddings without requiring manual annotations. Figure 3 illustrates the contrastive loss based paradigm of SSL, using the SOTA approach called

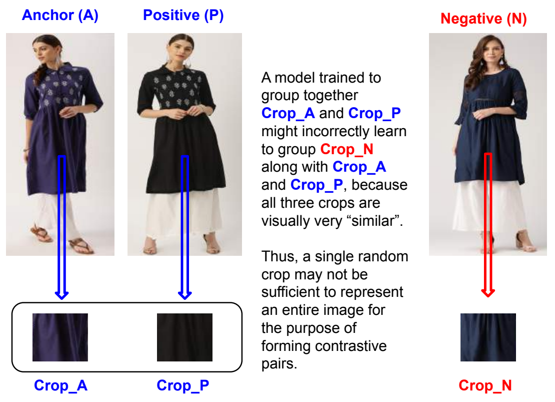

Anchor (A)
Positive (P)
Negative (N)
A model trained to group together Crop_A and Crop_P might incorrectly learn to group Crop_N along with Crop_A and Crop_P, because all three crops are visually very "similar".
Thus, a single random crop may not be sufficient to represent an entire image for the purpose of forming contrastive pairs.
Crop_A
Crop_P
Crop_N

Figure 4: Drawbacks of random crop in our use-case.

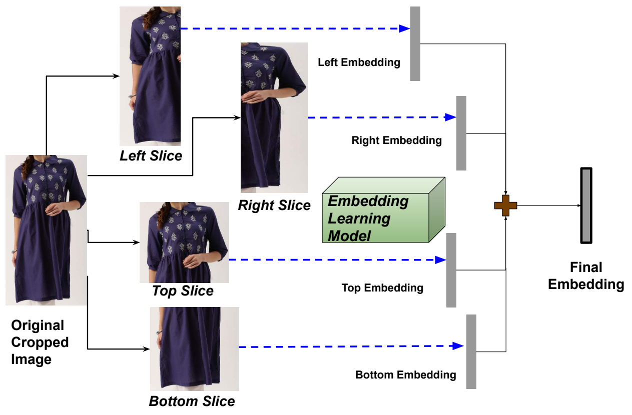


Figure 5: Illustration of our slicing based approach.

MOCOv2 (He et al. 2020). Given an image (Im 1), two augmentations (Im 1-Aug 1 and Im 1-Aug2) are obtained, which are passed through the Embedding Learning model that is maintained as two branches: The Query Encoder and the Key encoder, with parameters θ_q and θ_k respectively. It should be noted that the Key encoder is simply a copy of the Query Encoder, and is maintained as a *momentum encoder* obtained using an Exponential Moving Average (EMA) of the latter. The respective embeddings obtained for Im 1-Aug 1 and Im 1-Aug2 are treated as an anchor-positive pair. But, as we do not have manual annotations, to obtain the negative embedding (for avoiding a model collapse if just anchor and positive are pulled), one simply passes augmentation (Im 2-Aug 2) of another distinct image Im 2 (from within a mini-batch) through the key encoder. Additionally, to facilitate comparison with a large pool of negatives, another practice is to store the mini-batch embeddings obtained from the key encoder, in a separate memory module stored as a queue. The final embeddings of inference images are obtained by passing through the query encoder, which is the only branch that is updated via backpropagation (and not the key branch).

Other recent SOTA methods are built in a similar fashion as MOCOv2, with minor modifications. For instance, the BYOL (Grill et al. 2020) method does not make use of negatives (and hence no memory queue), but uses the EMA based update/ momentum encoder. The SimSiam method (Chen and He 2021) neither uses negatives, nor momentum encoder. All these SSL methods (MOCOv2, BYOL and SimSiam) employ random cropping in the data augmentation pipeline. However, when we already have object detection in our pipeline, the standard random crop step used in existing SSL methods may actually miss out important regions of a fashion product, which might be crucial in identifying color variants (Figure 4).

For this reason, we rather choose to propose a novel SSL variation that considers multiple slices/ patches (left, right, top and bottom) of the primary fashion object (after object detection), and simultaneously obtain embeddings for each of them. The final sum-pooled embedding[1] is then used to optimize a SSL based contrastive loss (Figure 5). We make use of negative pairs in our method because we found the performance of methods that do not make use of negative pairs (eg, SimSiam (Chen and He 2021), BYOL (Grill et al. 2020)) to be sub-optimal in our use-case. We also found merits of momentum encoder and memory queue in our use-case, and thus include them in our method. Following is the Normalized Temperature-scaled cross entropy (NT-Xent) loss (Chen et al. 2020) based objective of our method:

$$\mathcal{L}_{\boldsymbol{q}} = -\log \frac{\exp(\boldsymbol{q}\boldsymbol{k}_q/\tau)}{\sum_{i=0}^{K}\exp(\boldsymbol{q}\boldsymbol{k}_i/\tau)} \tag{2}$$

Here, $\boldsymbol{q} = \sum_v \boldsymbol{q}^{(v)}$, $\boldsymbol{k}_i = \sum_v \boldsymbol{k}_i^{(v)}, \forall i$. In (2), $\boldsymbol{q}$ and $\boldsymbol{k}_i$ respectively denote the *final* embeddings obtained for a query and a key, which are essentially obtained by adding the embeddings obtained from across all the views, as denoted by the superscript v for $\boldsymbol{q}^{(v)}$ and $\boldsymbol{k}_i^{(v)}$. Also, $\boldsymbol{k}_q$ represents the positive key corresponding to a query $\boldsymbol{q}$, τ denotes the temperature parameter, whereas exp() and log() respectively denote the exponential and logarithmic functions. K in (2) denotes the size of the memory queue.

We call our method as **Patch-Based Contrastive Net (PBCNet)**. We conjecture that considering multiple patches i) do not leave things to chance (as in random crops), and provides a deterministic approach to obtain embeddings, ii) enables us to borrow more information (from other patches) to make a better decision on grouping a pair of similar embeddings. Our conjecture is supported not only by evidence of improved discriminative performances by the consideration of multiple fine patches of images, in literature (Wang et al. 2018), but also by our experimental results.

Experiments

We evaluated the discussed methods on a large, challenging internal collection of images on our platform (www.myntra.com). Here, we report our results on a collection of Kurtas images. We used the exact same set to train the supervised (with labeled training data, obtained by our in-house team) and self-supervised methods (without labeled training data)

[1]Rather than concatenating or averaging, we simply add them together to maintain simplicity of the model.

Dataset	(w/o norm)	SimSiam_v0	(w/o norm)	SimSiam_v1
Data 1	0.5	0.5	**1**	0.67
Data 2	0.5	**0.67**	0	**0.25**
Data 3	0	**0.4**	0	**0.33**

Table 1: Effect of Embedding Normalization (wrt CGacc).

for a fair comparison. For inferencing, we use 6 dataset splits (based on brand, gender, MRP), referred to as Data 1-6. **Performance metrics used:** To evaluate the methods, we made use of the following performance metrics: i) Color Group Accuracy (**CGacc**), an internal metric with business relevance, that reflects the *precision*, ii) Adjusted Random Index (**ARI**), iii) Fowlkes-Mallows Score (**FMS**), and iv) Clustering Score (**CScore**), computed as: $CScore = \frac{2.ARI.FMS}{ARI+FMS}$. While we use CGacc to compare the methods for all the datasets (*Data 1-6*), the remaining metrics are reported only for *Data 4-6*, where we obtain the ground-truth labels for evaluation. All the performance metrics take values in the range $[0, 1]$, where a higher value indicates a better performance.

Methods Compared: The focus of our experiments is to evaluate the overall performance by using different embedding learning methods (while fixing the object detector and the clustering algorithm). We compare the supervised triplet loss based model, and our proposed PBCNet method, against the following SOTA SSL methods: **SimSiam** (Chen and He 2021), **BYOL** (Grill et al. 2020), and **MOCOv2** (He et al. 2020).

We implemented all the methods in PyTorch. For all the compared methods, we fix a ResNet34 (He et al. 2016) as a base encoder with 224×224 image resizing, and train all the models for a fixed number of 30 epochs, for a fair comparison. The number of epochs was fixed based on observations on the supervised model, to avoid overfitting. For the purpose of object detection, we made use of YOLOv2 (Redmon and Farhadi 2017), and for the task of clustering the embeddings, we made use of the Agglomerative Clustering algorithm with Ward's method for merging. In all cases, the 512-dimensional embeddings used for clustering are obtained using the avgpool layer of the trained encoder. A margin of 0.2 has been used in the triplet loss for training the supervised model.

Systematic Study of SSL for Our Use-case

We now perform a systematic study of the typical aspects associated with SSL, especially for our particular task of color variants identification. For this purpose, we make use of a single Table 2, where we provide the comparison of various SSL methods, including ours.

Effect of Data augmentation on our task: Firstly, for illustrating the effect of different data augmentations, we consider two variants of SimSiam (for its simplicity and strength): i) SimSiam_v0: A version of SimSiam, where we used the entire original image as the query, and a color jittered image as the positive, and with a batch size of 12, and ii) SimSiam_v1: SimSiam with standard SSL (Chen et al. 2020) augmentations (`ColorJitter`, `RandomGrayscale`, `RandomHorizontalFlip`, `GaussianBlur` and `RandomResizedCrop`), and a batch size of 12. For all cases, the following architecture has been used for SimSiam: `Encoder`{ ResNet34 $\rightarrow$ (avgpool) $\rightarrow$ `ProjectorMLP`(512$\rightarrow$4096$\rightarrow$123) }$\rightarrow$`PredictorMLP`(123$\rightarrow$4096$\rightarrow$123).

From Table 2, we observed that considering standard augmentations with random crops leads to a better performance than that of SimSiam_v0. Thus, for all other self-supervised baselines, i.e., BYOL and MOCOv2 we make use of standard augmentations with random crops. However, we later show that our proposed way of considering multiple patches in PBCNet leads to a better performance.

Effect of l2 normalization for our task: Table 1 shows the effect of performing $l2$ normalization on the embeddings obtained using SimSiam. We found that without using any normalization, in some cases (eg, Data 2-3) there are no true color variant groups out of the detected clusters (i.e., zero precision), and hence the performance metrics become zero. Thus, for all our later experiments, we make use of $l2$ normalization on the embeddings as a de facto standard, for all the methods.

Effect of Batch Size and Momentum Encoding in SSL for our task: For studying the effect of batch size in SSL for our task, we introduce a third variant of SimSiam, i.e., SimSiam_v2: This is essentially SimSiam_v1 with a batch size of 128. We then consider SimSiam_v1, SimSiam_v2 and BYOL, where the first makes use of a batch size of 12, while the others make use of batch sizes of 128. We observed that a larger batch size usually leads to a better performance. This is observed from Table 2, by the higher values of performance metrics in the columns for SimSiam_v2 and BYOL (vs SimSiam_v1). Additionally, we noted that the momentum encoder used in BYOL causes a further boost in the performance, as observed in its superior performance as compared to SimSiam_v2 that has the same batch size. It should be noted that except for the momentum encoder, the rest of the architecture and augmentations used in BYOL are exactly the same as in SimSiam. We observed that increasing the batch size in SimSiam does not drastically or consistently improve its performance, something which its authors also noticed (Chen and He 2021).

Effect of Memory Queue in SSL for our task: We also inspect the effect of an extra memory module/queue being used to facilitate the comparisons with a large number of negative examples. In particular, we make use of the MOCOv2 method with the following settings: i) queue size of 5k, ii) temperature parameter of 0.05, iii) a MLP (512$\rightarrow$4096$\rightarrow$relu$\rightarrow$123) added after the avgpool layer of the ResNet34, iv) SGD for updating the query encoder, with learning rate of 0.001, momentum of 0.9, weight decay of $1e^{-6}$, and v) value of 0.999 for m in the momentum update. It is observed from Table 2, by the columns of MOCOv2 and BYOL, that the performance of the former is superior. As BYOL does not use a memory module, but MOCOv2 does, we conclude that using a separate memory module significantly boosts the performance of SSL in our task. Moti-

		Supervised	Self-Supervised					
Dataset	Metric	**Triplet Net**	SimSiam_v0	SimSiam_v1	SimSiam_v2	BYOL	MOCOv2	**PBCNet (Ours)**
Data 1	CGacc	0.67	0.5	0.67	1	0.5	1	**1**
Data 2	CGacc	1	0.67	0.25	1	0.75	**0.8**	0.75
Data 3	CGacc	0.75	0.4	0.33	0	0.5	0.5	**0.6**
Data 4	CGacc	0.67	0.4	0.5	0.5	0.5	**1**	0.85
	ARI	0.69	0.09	0.15	0.12	0.27	0.66	**0.75**
	FMS	0.71	0.15	0.22	0.20	0.30	0.71	**0.76**
	CScore	0.700	0.110	0.182	0.152	0.281	0.680	**0.756**
Data 5	CGacc	1	0	0.5	0.33	0.5	1	**1**
	ARI	1	0	0.09	0.28	0.64	1	**1**
	FMS	1	0.22	0.30	0.45	0.71	1	**1**
	CScore	1	0	0.135	0.341	0.674	1	**1**
Data 6	CGacc	0.83	0.5	0.5	0.8	0.6	1	**1**
	ARI	0.44	0.07	0.06	0.04	0.20	0.58	**0.79**
	FMS	0.49	0.12	0.17	0.13	0.24	0.64	**0.80**
	CScore	0.466	0.089	0.090	0.063	0.214	0.610	**0.796**

Table 2: Comparison of our proposed method against the supervised and state-of-the-art SSL baselines, across all the datasets.

vated by our observations so far, we choose to employ both momentum encoding and memory module in our proposed PBCNet method.

Comparison Against State-of-the-art

In Table 2, we provide the comparison of our proposed SSL method **PBCNet** against the SSL SOTA baselines and the supervised baseline across all the datasets. It should be noted that in Table 2, any performance gains for a specific method is due to the intrinsic nature of the same, and not because of a particular hyperparameter setting. This is because we report the *best performance for each method* after adequate tuning of the clustering distance threshold and other parameters, and not just their default hyperparameters.

Following are the configurations that we have used in our **PBCNet** method: i) memory module size of 5k, ii) temperature parameter of 0.05, iii) the FC layer after the avgpool layer of the ResNet34 was removed, iv) SGD for updating the query encoder, with learning rate of 0.001, momentum of 0.9, weight decay of $1e^{-6}$, and v) value of 0.999 for m in the momentum update.

We made use of a batch size of $32 (= 128/4)$ as we have to store tensors for each of the 4 slices simultaneously for each mini-batch (we used a batch size of 128 for the other methods). For data augmentation, we first apply a color distortion in the following order: i) `ColorJitter(0.8 * s, 0.8 * s, 0.8 * s, 0.2 * s)` with s=1, p=0.8, ii) `RandomGrayscale` with p=0.2, iii) `GaussianBlur((3, 3), (1.0, 2.0))` with p=0.5. After the color distortion, we apply our slicing technique. For the second image (positive/negative) we apply the same series of transformations.

From Table 2, it is clear that the SimSiam method despite its strong claims of not using any negative pairs, nor momentum encoder nor large batches, performs poorly as compared to our supervised method (shown in bold blue color). The BYOL method which also does not make use of negative pairs, performs better than the SimSiam method in our use case, by virtue of its momentum encoder.

Among all compared SSL baselines, it is the MOCOv2

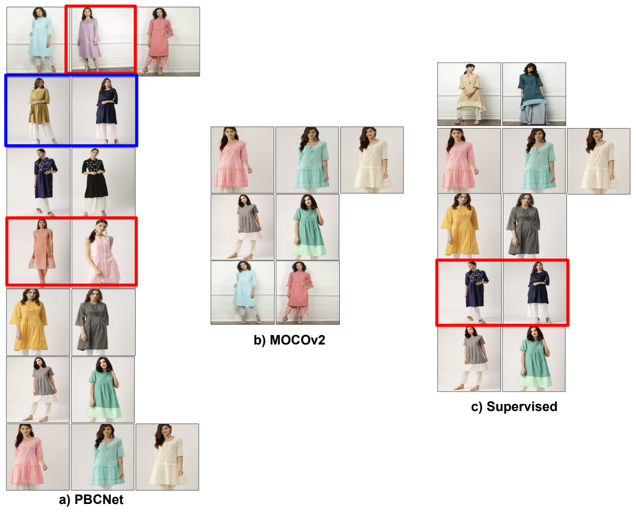


Figure 6: Color variants groups on *Data 4*.

method that performs the best. This is due to the reason of the memory queue that facilitates the comparison with a large number of negative examples. This shows that the importance of considering negative pairs still holds true, especially for challenging use-cases like the one considered in the paper. However, our proposed self-supervised method **PBCNet** clearly outperforms all the baselines. The fact that it outperforms MOCOv2 can be attributed to the patch-based slicing used, which is the only different component in our method in comparison to MOCOv2 that uses random crop. Another interesting thing that we observed is the fact that despite using much lesser batch size of 32, our method outperforms the baselines. In a way, we were able to extract and leverage more information by virtue of the slicing (by borrowing information from the other patches simultaneously), even with smaller batches.

We also noticed that the supervised baseline performs quite good in our task, even without any data augmentation pipeline as used in the SSL methods. However, by virtue

Dataset		Data 4		Data 5		Data 6	
Method	Clustering	**ARI**	**FMS**	**ARI**	**FMS**	**ARI**	**FMS**
PBCNet	Agglo	**0.75**	**0.76**	**1.00**	**1.00**	**0.79**	**0.80**
	DBSCAN	0.66	0.71	1.00	1.00	0.66	0.71
	Affinity	0.30	0.42	0.22	0.41	0.24	0.37
MOCOv2	Agglo	**0.66**	**0.71**	**1.00**	**1.00**	**0.58**	**0.64**
	DBSCAN	0.66	0.71	1.00	1.00	0.37	0.40
	Affinity	0.20	0.32	0.04	0.26	0.25	0.41
BYOL	Agglo	**0.27**	**0.30**	**0.64**	**0.71**	**0.20**	**0.24**
	DBSCAN	0.17	0.28	0.64	0.71	0.02	0.24
	Affinity	0.03	0.14	0.28	0.45	0.01	0.11

Table 3: Effect of the clustering technique used.

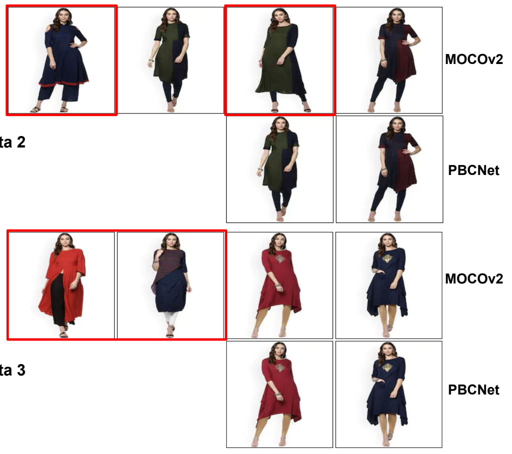
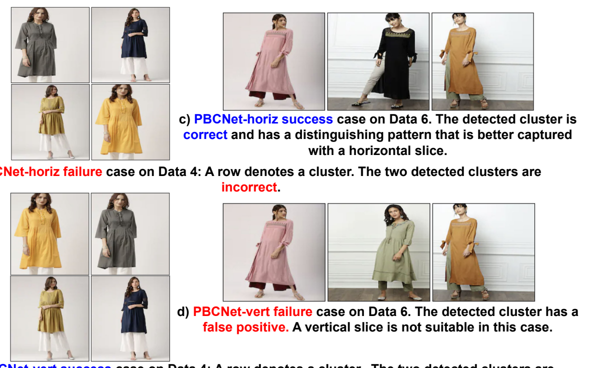


Figure 7: MOCOv2 yields false positives (shown in red box).


a) PBCNet-horiz failure case on Data 4: A row denotes a cluster. The two detected clusters are incorrect.

b) PBCNet-vert success case on Data 4: A row denotes a cluster. The two detected clusters are correct.

c) PBCNet-horiz success case on Data 6. The detected cluster is correct and has a distinguishing pattern that is better captured with a horizontal slice.

d) PBCNet-vert failure case on Data 6. The detected cluster has a false positive. A vertical slice is not suitable in this case.

Figure 8: Trade off between vertical and horizontal slicing.

Dataset	Data 1	Data 2	Data 3	Data 4		
Method	CGacc	CGacc	CGacc	CGacc	ARI	FMS
PBCNet-horiz	1	**1**	0.5	0.66	0.65	0.67
PBCNet-vert	1	0.6	0.6	**1**	**0.88**	**0.89**
PBCNet	1	0.75	**0.6**	0.85	0.75	0.76
Dataset	Data 5			Data 6		
Method	CGacc	ARI	FMS	CGacc	ARI	FMS
PBCNet-horiz	1	1	1	1	**0.81**	**0.82**
PBCNet-vert	1	1	1	0.83	0.48	0.5
PBCNet	1	1	1	1	0.79	0.8

Table 4: Effect of Slicing on PBCNet

of the large memory queue, and the data augmentations like color jitter, which are pretty relevant to the task of color variants identification, stronger SSL methods like MOCOv2 and PBCNet are in fact capable of achieving comparable performance against the supervised baseline. Having said that, if we do not have adequate labeled data in the first place, we cannot even use supervised learning. Hence, enabling data augmentations and slicing strategy in the supervised model has not been focused, because the necessity of our approach comes from the issue of addressing the lack of labeled data, and not to improve the performance of supervised learning (which any how is label dependent).

Effect of Clustering: In Table 3, we report the performances obtained by varying the clustering algorithm to group embeddings obtained by different SSL methods, on *Data 4-6*. We picked the Agglomerative, DBSCAN and Affinity Propagation clustering techniques that do not require the number of clusters as input parameter (which is difficult to obtain in our use-case). In general, we observed that the Agglomerative clustering technique leads to a better performance in our use-case. Also, for a fixed clustering approach, using embeddings obtained by our PBCNet method usually leads to a better performance.

Qualitative results: Sample qualitative comparisons of color variants groups obtained on *Data 4* using our PBCNet method, MOCOv2 and the supervised baseline are provided in Figure 6. Each of the rows for a column corresponding to a method represents a detected color variants cluster for the considered method. A row has been marked with a red box if the entire cluster contains images that are not color variants to each other. A single image is marked with a red box if it is the only incorrect image, while rest of the images are color variants. We observed that our method not only detects clusters with higher precision (which MOCOv2 does as well), but it also has a higher recall, which is comparable to the supervised method. We also make use of a blue box to show a detected color group by our method which contains images that are color variants, but are difficult to be identified at a first glance, even for humans.

Additionally, Figure 7 shows a few color groups identified in the datasets *Data 2 & 3* using MOCOv2 and our PBCNet. We observed that MOCOv2 detected groups with false positives, while our PBCNet method did not. This could happen because when a random crop is obtained by MOCOv2, it need not necessarily be from a *distinctive* region of an apparel that helps to identify its color variant (eg, in Figure 7, the bent line like pattern separating the colored and black region of the apparels of Data 2, and the diamond like shape in the apparels of Data 3). We argue that a random crop might have arrived from such a *distinctive* region given that the size of the crop is made larger, etc. But that still leaves things to random chance. On the other hand, our slicing technique being deterministic in nature, *guarantees* that all the regions of an object *would* be captured. We would also like to mention that our slicing approach is agnostic to the fashion apparel type, i.e, the same is easily applicable for any fashion article type (Tops, Shirts, Shoes, Trousers, Track pants, Sweatshirts, etc). In fact, this is how a human identifies color variants as well, by looking at the article along both horizontal and vertical directions, to identify distinctive patterns. Even humans cannot identify an object if we restrict our vision to only a particular small crop.

Effect of the slicing: We also study 2 variants of our PBCNet method: i) PBCNet-horiz (computing an embedding only by considering the top and bottom slices), and ii) PBCNet-vert (computing an embedding using only the left and right slices). The results are shown in Table 4. In *Data 4*, PBCNet-vert performs better than PBCNet-horiz, and in *Data 6*, PBCNet-horiz performs better than PBCNet-vert (significantly). The performance of the two versions is also illustrated in Figure 8. We observed that a single slicing do not work in all scenarios, especially for apparels.

Although the horizontal slicing is quite competitive, it may be beneficial to consider the vertical slices as well. This is observed by the drop in performance of PBCNet-horiz in *Data 3-4* (vs PBCNet). This is because some garments may contain distinguishing patterns that may be better interpreted only by viewing vertically, for example, printed texts (say, *adidas* written vertically), floral patterns etc. In such cases, simply considering horizontal slices may actually split/ disrupt the vertical information. It may also happen that mixing of slicing introduces some form of redundancy, as observed by the occasional drop in the performance of PBCNet when compared to PBCNet-horiz (on *Data 6*) and PBCNet-vert (on *Data 4*). However, on average PBCNet leads to an overall consistent and competitive performance, while avoiding drastically fluctuating improvements or failures. We suggest considering both the directions of slicing, so that they could collectively represent all necessary and distinguishing patterns, and if one slicing misses some important information, the other could compensate for it.

Conclusions

In this paper, a generic visual Representation Learning based framework to identify color variants in fashion e-commerce has been studied (to the best of our knowledge, for the first time). A systematic study of a supervised method (with manual annotations), as well as existing SOTA SSL methods to train the embedding learning component of our framework has been conducted. A novel contrastive loss based SSL method that focuses on parts of an object to identify color variants, has also been proposed.

Acknowledgments

We would like to thank Dr Ravindra Babu Tallamraju for his support, feedback, and in being a source of inspiration and encouragement throughout the project.

References

Cao, X.; Chen, B.-C.; and Lim, S.-N. 2019. Unsupervised deep metric learning via auxiliary rotation loss. *arXiv preprint arXiv:1911.07072*.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. 2020. A simple framework for contrastive learning of visual representations. In *Proc. of International Conference on Machine Learning (ICML)*, 1597–1607. PMLR.

Chen, X.; and He, K. 2021. Exploring Simple Siamese Representation Learning. *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Dutta, U. K.; Harandi, M.; and Sekhar, C. C. 2020a. Unsupervised Deep Metric Learning via Orthogonality Based Probabilistic Loss. *IEEE Transactions on Artificial Intelligence (TAI)*, 1(1): 74–84.

Dutta, U. K.; Harandi, M.; and Sekhar, C. C. 2020b. Unsupervised Metric Learning with Synthetic Examples. In *Proc. of Association for the Advancement of Artificial Intelligence (AAAI)*.

Grill, J.-B.; Strub, F.; Altché, F.; Tallec, C.; Richemond, P. H.; Buchatskaya, E.; Doersch, C.; Pires, B. A.; Guo, Z. D.; Azar, M. G.; et al. 2020. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*.

Gu, G.; and Ko, B. 2020. Symmetrical Synthesis for Deep Metric Learning. In *Proc. of Association for the Advancement of Artificial Intelligence (AAAI)*.

He, K.; Fan, H.; Wu, Y.; Xie, S.; and Girshick, R. 2020. Momentum contrast for unsupervised visual representation learning. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 9729–9738.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

Jagadeesh, V.; Piramuthu, R.; Bhardwaj, A.; Di, W.; and Sundaresan, N. 2014. Large scale visual recommendations from street fashion images. In *Proc. of ACM Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD)*, 1925–1934.

Jing, L.; and Tian, Y. 2020. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*.

Li, Y.; Kan, S.; and He, Z. 2020. Unsupervised Deep Metric Learning with Transformed Attention Consistency and Contrastive Clustering Loss. In *Proc. of European Conference on Computer Vision (ECCV)*.

Redmon, J.; and Farhadi, A. 2017. YOLO9000: better, faster, stronger. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 7263–7271.

Schroff, F.; Kalenichenko, D.; and Philbin, J. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 815–823.

Sun, Y.; Cheng, C.; Zhang, Y.; Zhang, C.; Zheng, L.; Wang, Z.; and Wei, Y. 2020. Circle loss: A unified perspective of pair similarity optimization. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6398–6407.

Wang, G.; Yuan, Y.; Chen, X.; Li, J.; and Zhou, X. 2018. Learning discriminative features with multiple granularities for person re-identification. In *Proc. of ACM International Conference on Multimedia (ACMMM)*, 274–282.