Color Variants Identification in Fashion e-commerce via Contrastive Self-Supervised Representation Learning

Ujjal Kr Dutta, Sandeep Repakula, Maulik Parmar and Abhinav Ravi
Data Sciences-Image Sciences, Myntra
Bengaluru, Karnataka, India
{ujjal.dutta,sandeep.r,parmar.m,abhinav.ravi}@myntra.com

Abstract
In this paper, we utilize deep visual Representation Learning to address an important problem in fashion e-commerce: color variants identification, i.e., identifying fashion products that match exactly in their design (or style), but only to differ in their color. At first we attempt to tackle the problem by obtaining manual annotations (depicting whether two products are color variants), and train a supervised triplet loss based neural network model to learn representations of fashion products. However, for large scale real-world industrial datasets such as addressed in our paper, it is infeasible to obtain annotations for the entire dataset, while capturing all the difficult corner cases. Interestingly, we observed that color variants are essentially manifestations of color jitter based augmentations. Thus, we instead explore Self-Supervised Learning (SSL) to solve this problem. We observed that existing state-of-the-art SSL methods perform poor, for our problem. To address this, we propose a novel SSL based color variants model that simultaneously focuses on different parts of an apparel. Quantitative and qualitative evaluation shows that our method outperforms existing SSL methods, and at times, the supervised model.

1 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. At Myntra (www.myntra.com), a leading e-commerce platform, we address this problem by leveraging deep visual Representation/Embedding Learning.

Firstly, we obtained 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, consisting of an anchor, a positive and a negative (Figure 1). The positive is usually an image that consists of a product that is a color variant of the product contained in the anchor image. The negative is an image that consists of a product that is not a color variant of the products contained in the anchor and positive images. These triplets are used to train a supervised triplet loss based neural network model [Schroff et al., 2015; Veit et al., 2017] in order to obtain deep embeddings of fashion products. Having obtained the embeddings, we perform clustering on them to group the color variants.

A key challenge in this supervised approach is that of obtaining manual annotations, which not only requires fashion domain expertise, but is also infeasible, given the large scale of our platform, and the huge number and complexity of the fashion products. As visual Self-Supervised Learning (SSL) obtains image embeddings without requiring manual annotations, we consider it as a candidate to address our problem, i.e., lack of annotations for our large data. The motivation for this comes from the fact that typical visual SSL methods employ a color jitter based data augmentation step.

Interestingly, color variants of fashion products are in essence, manifestations of color jittering. It should be noted that a Product Display Page (PDP) image in a fashion e-commerce platform may contain multiple fashion products. Thus, we must apply an object detector to localise our primary fashion product of interest (Figure 3). However, when we already employ object detection, the standard random crop step used in existing SSL methods may actually miss out im-
The problem of visual embedding/metric learning refers to variation that considers multiple slices/patches of the primary examples need to be brought closer. Of pairs, triplets or tuples, that indicate embeddings of which constraints in embedding learning are either in the form of anchor, positive and negative (to the best of our knowledge addressed for the first time). As discussed using Figure 2, a random crop of an image may not be sufficient to represent an entire image for the purpose of forming contrastive pairs. Thus, a single random crop may not be the best representative to form contrastive pairs for our contribution is only specific to the embedding learning problem by using a supervised embedding learning model.

Visual SSL [Jing and Tian, 2020] refers to the learning of representations of images without making use of class labels. A popular paradigm of SSL is contrastive learning, that groups together embeddings obtained from augmentations of the same image. Many recent SSL approaches have been proposed that vary in their implementation details (for example, using momentum encoding, memory module, only positive pairs, etc) [Chen et al., 2020; He et al., 2020; Grill et al., 2020; Chen and He, 2021].

## 2 Proposed Approach

In this section, we discuss the Representation Learning methods used to address the problem of color variants identification.

### 3.1 Supervised Color Variants Model

Firstly, we shall discuss the proposed supervised approach of addressing the color variants identification problem. Our proposed pipeline leveraging the supervised model is illustrated in Figure 3. The pipeline consists of the following major components (or steps) in the same order: 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 to localise the primary fashion product of interest. Having obtained the cropped image of the fashion article, we form triplet based constraints (in the form of anchor, positive and negative) using the available manual annotations. These triplets are used to train an embedding learning model. The obtained embeddings are then grouped together by using an appropriate clustering algorithm. An obtained cluster then contains embeddings of images of fashion 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 industrial datasets (such as those present in our platform). Thus, we now propose a novel self-supervised representation learning model to identify color variants without making use of manual annotations.

### 3.2 Self-Supervised Color Variants Model

As discussed using Figure 2, a random crop of an image may not be the best representative to form contrastive pairs for SSL, at least for our task. Hence, we propose a method that simultaneously considers multiple, fixed, slices/patches of an object to form embeddings. This is illustrated in Figure 4. As our contribution is only specific to the embedding learning component, Figure 4 focuses only on this. As illustrated, the
Also, adding an extra memory module in the form of a queue together to maintain simplicity of the model. A large number of negatives. Thus, BYOL [Grill et al., 2020] to be sub-optimal in our use-case. The objective of our method is similar to the commonly used Normalized Temperature-scaled cross entropy (NT-Xent) loss [Chen et al., 2020]. In particular, we make use of two branches for the encoders of our embedding learning model, one for the query and another for the key [Chen et al., 2020; Grill et al., 2020]. In practice, an encoder is a Convolutional Neural Network (CNN) that takes a raw input image and produces an embedding vector. The query encoder is a CNN that obtains embeddings for the anchor images, while the key encoder is a copy of the same CNN that obtains embeddings for the positives and negatives. Gradients are backpropagated only through the query encoder, which is used to obtain the final representations of the inference data.

Following is the objective of our method:

$$L_q = -\log \frac{\exp(qk_q/\tau)}{\sum_{i=0}^{K} \exp(qk_i/\tau)} \quad (1)$$

Here, \( q = \sum_v q_v \), \( k_i = \sum_v k_i(v) \), \( q \) and \( k_i \) 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 \( q(v) \) and \( k_i(v) \). Also, \( k_q \) represents the positive key corresponding to a query \( q \), \( \tau \) denotes the temperature parameter, whereas \( \exp() \) and \( \log() \) respectively denote the exponential and logarithmic functions.

During our experiments, we observed that simple methods like SimSiam [Chen and He, 2021] do not actually perform too well for our use-case. On the contrary, we found benefits of components such as the momentum encoder, as present in BYOL [Grill et al., 2020] and MOCOv2 [He et al., 2020]. Also, adding an extra memory module in the form of a queue helps in boosting the performance due to comparisons with a large number of negatives. Thus, \( K \) in (1) denotes the size of the memory module. Our method is called as **Patch-Based Contrastive Net (PBCNet)**.

## 4 Experiments

We evaluated the discussed methods on a large (orders of magnitudes of \( 10^5 \)) internal collection of challenging real-world industrial images on our Myntra platform (www.myntra.com) that hosts various fashion products. In this section, we report our results on a collection of roughly \( 0.13 \) million Kurtas images from our internal database. We used the exact same set to train the supervised (with labeled training data) and self-supervised methods (without labeled training data) for a fair comparison. For inferencing, we used the entire \( 0.13 \) million Kurtas images, which are present in the form of different dataset splits (based on brand, gender, MRP). We refer to our 6 dataset splits as Data 1-6. Details of the data and the performance metrics (CGacc for all splits, ARI, FMS and CScore for splits 4-6, a higher value indicates a better performance) are deferred to the supplementary material.

**Methods Compared:** Following are the methods that we have compared for representation learning:

1. **Triplet Loss based Deep Neural Network** [Schroff et al., 2015; Veit et al., 2017]: This is our supervised baseline that is trained using triplet based constraints obtained using the labeled data.
2. **SimSiam** [Chen and He, 2021]: This is a recently proposed State-Of-The-Art (SOTA) visual SSL method that neither uses negative pairs, nor momentum encoder, nor large batches.
3. **BYOL** [Grill et al., 2020]: This is another recently proposed SOTA SSL method that also does not make use of negative pairs, but makes use of batch normalization and momentum encoding.
4. **MOCOv2** [He et al., 2020]: This is a SOTA SSL method that makes use of negative pairs, a momentum encoder, as well as a memory queue.
5. **PBCNet**: This is our proposed novel self-supervised method.

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 \( 224 \times 224 \) 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.

### 4.1 Systematic Study of SSL for color variants identification

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.

Convergence behavior with data augmentation for our task: For illustrating how the convergence behavior of SSL methods changes with respect to a different data augmentation, we pick the SimSiam SSL method for its simplicity and strong claims. We consider two variants of SimSiam: 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: A version of 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 → (avgpool) → ProjectorMLP(512→4096→relu→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 $\theta$ in the momentum update. It 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→4096→relu→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 $\theta$ 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. Motivated by our observations so far, we choose to employ both momentum encoding and memory module in our proposed PBCNet method.

4.2 Comparison of PBCNet against the state-of-the-art

In Table 2, we provide the comparison of our proposed self-supervised method PBCNet against various self-supervised state-of-the-art 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 distance threshold (details in supplementary) 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 $\theta$ 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...
the 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 use-case. Also, for a fixed clustering approach, using embedding clustering techniques that do not require the 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 of data augmentations like color jitter and cropping, which are pretty relevant to the task of color variants identification, stronger SSL methods like MOCOv2 and PBCNet are in fact capable of surpassing the performance of the supervised baseline as well, in some cases. 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. However, by virtue of data augmentations like color jitter and cropping, which are pretty relevant to the task of color variants identification, stronger SSL methods like MOCOv2 and PBCNet are in fact capable of surpassing the performance of the supervised baseline as well, in some cases. 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 Slicing:** In Table 4, we report the performances obtained by varying the slicing 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.
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.

5 Conclusions

In this paper, we utilize deep visual Representation Learning to address the problem of identification of color variants (images of objects exactly similar in design, but not color), particularly for fashion products. A supervised triplet loss based deep neural network model for visual Representation Learning has been proposed to identify the color variants. A systematic study of existing state-of-the-art self-supervised methods has been done to solve the proposed problem, while alleviating the need for manual annotations. Also, a novel contrastive loss based self-supervised representation learning method that focuses on parts of an object has been proposed. This is done to make the model better informed of the discriminative regions of an image, in order to identify color variants.

References

[Cao et al., 2019] Xuefei Cao, Bor-Chun Chen, and Ser-Nam Lim. Unsupervised deep metric learning via auxiliary rotation loss. *arXiv preprint arXiv:1911.07072*, 2019.

[Chen and He, 2021] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[Chen et al., 2020] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for con-

![Figure 7: A few groups obtained on Data 2 & 3 using MOCOv2 have false positives (shown in red box), while our PBCNet method does not yield such groups.](image)

![Figure 8: Trade off between vertical and horizontal slicing.](image)
Supervised learning of visual representations. In Proc. of International Conference on Machine Learning (ICML), pages 1597–1607. PMLR, 2020.

[Dutta et al., 2020a] Ujjal Kr Dutta, Mehrtash Harandi, and C Chandra Sekhar. Unsupervised metric learning with synthetic examples. In Proc. of Association for the Advancement of Artificial Intelligence (AAAI), 2020.

[Dutta et al., 2020b] Ujjal Kr Dutta, Mehrtash Harandi, and Chellu Chandra Sekhar. Unsupervised deep metric learning via orthogonality based probabilistic loss. IEEE Transactions on Artificial Intelligence (TAI), 1(1):74–84, 2020.

[Grill et al., 2020] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. arXiv preprint arXiv:2006.07733, 2020.

[Gu and Ko, 2020] Geonmo Gu and Byungsoo Ko. Symmetrical synthesis for deep metric learning. In Proc. of Association for the Advancement of Artificial Intelligence (AAAI), 2020.

[He et al., 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016.

[He et al., 2020] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 9729–9738, 2020.

[Jagadeesh et al., 2014] Vignesh Jagadeesh, Robinson Piramuthu, Anurag Bhardwaj, Wei Di, and Neel Sundareshan. Large scale visual recommendations from street fashion images. In Proc. of ACM Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD), pages 1925–1934, 2014.

[Jing and Tian, 2020] Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2020.

[Levi et al., 2020] Elad Levi, Tete Xiao, Xiaolong Wang, and Trevor Darrell. Reducing class collapse in metric learning with easy positive sampling. arXiv preprint arXiv:2006.05162, 2020.

[Li et al., 2020] Yang Li, Shichao Kan, and Zhihai He. Unsupervised deep metric learning with transformed attention consistency and contrastive clustering loss. In Proc. of European Conference on Computer Vision (ECCV), 2020.

[Redmon and Farhadi, 2017] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 7263–7271, 2017.

[Schroff et al., 2015] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 815–823, 2015.

[Sun et al., 2020] Yifan Sun, Changmiao Cheng, Yuhang Zhang, Chi Zhang, Liang Zheng, Zhongdao Wang, and Yichen Wei. Circle loss: A unified perspective of pair similarity optimization. In Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 6398–6407, 2020.

[Veit et al., 2017] Andreas Veit, Serge Belongie, and Theofanis Karatzos. Conditional similarity networks. In Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 830–838, 2017.

[Wang et al., 2018] Guanshuo Wang, Yufeng Yuan, Xiong Chen, Jiwei Li, and Xi Zhou. Learning discriminative features with multiple granularities for person re-identification. In Proc. of ACM International Conference on Multimedia (ACMMM), pages 274–282, 2018.

6 Supplementary: Additional Details and Results (omitted from the main text due to space constraints)

We evaluated the discussed methods on a large (orders of magnitudes of $10^5$) internal collection of challenging real-world industrial images on our Mynta platform (www.mynta.com) that hosts various fashion products. In this section, we report our results on a collection of roughly $3.13$ million Kurtas images from our internal database. A disjoint set of $6$ images were manually annotated by our in-house team to provide class labels indicating their color variants information. This labeled data was used to form triplets to train the supervised triplet loss based color variants model. We also used the exact same set to train the self-supervised methods for a fair comparison. For inferencing, we used the entire $0.13$ million Kurtas images, which are present in the form of different dataset splits (based on brand, gender, MRP). We refer to our dataset splits as Data 1-6. Table 5 provides details of the datasets used to compare the methods, along with their meta-data on our platform. It should be noted that the datasets Data 1-3 do not have manual annotations available for them, whereas the remaining three have been annotated by the taggers.

| Dataset | Article Type | Gender | Brand | MRP | Ground Truth Available |
|---------|--------------|--------|-------|-----|------------------------|
| Data 1  | Kurtas       | Women  | STREET 9 | 1899 | No                     |
| Data 2  | Kurtas       | Women  | STREET 9 | 1399 | No                     |
| Data 3  | Kurtas       | Women  | STREET 9 | 1499 | No                     |
| Data 4  | Kurtas       | Women  | all about you | 1699 | Yes                    |
| Data 5  | Kurtas       | Women  | all about you | 2499 | Yes                    |
| Data 6  | Kurtas       | Women  | all about you | 2199 | Yes                    |

Table 5: Details of datasets used to compare the methods.

| Dataset | Config Type | Gender | Brand | MRP | Ground Truth Available |
|---------|-------------|--------|-------|-----|------------------------|
| Data 1  | RCU | women | STREET 9 | 1899 | No                     |
| Data 2  | RCU | women | STREET 9 | 1399 | No                     |
| Data 3  | RCU | women | STREET 9 | 1499 | No                     |
| Data 4  | RCU | women | all about you | 1699 | Yes                    |
| Data 5  | RCU | women | all about you | 2499 | Yes                    |
| Data 6  | RCU | women | all about you | 2199 | Yes                    |

Table 6: Distance thresholds in Agglomerative clustering used to produce the best results (for each method) across all the datasets, along with the number of detected and correct color groups. Note that we have used the following notations: Config: Configuration, t:threshold, nCG:n detected groups, nDG:n detected groups, Sup:Supervised, S0:SimSiam_v0, S1:SimSiam_v1, S2:SimSiam_v2, B:BYOL, M:MOCoV2, and PB:PBNet.

$^3$Company compliance policies prohibit open-sourcing/ revealing exact dataset specifics.
Table 7: Varying values of performance metrics against distance thresholds. For a method, only those thresholds are reported against a dataset, using which Agglomerative Clustering produces non-zero performance metrics, thus indicating a meaningful clustering.

**Performance metrics used**: To evaluate the methods, we made use of the following performance metrics (CGacc and CScore are defined by us for our use-case):

1. **Color Group Accuracy (CGacc)**: It refers to the ratio of the number of correct color groups to that of the number of detected color groups. Here, detected color groups are the clusters identified by the clustering algorithm that have a size of at least two. A correct color group is a cluster among the detected color groups, which contains at least half of its examples which are actually color variants to each other (while implementing we take a floor function). It should be noted that this performance metric has a direct business relevance due to the fact that it reflects the precision. Also, it is computed by our in-house catalog team by performing manual Quality Check (QC).

2. **Adjusted Random Index (ARI)**: Let s (and d, respectively) be the number of pairs of elements that are in the same (and different, respectively) set in the ground truth class assignment, and in the same (and different, respectively) set in the clustering. Then, the (unadjusted) Random Index is computed as: $RI = \frac{s}{N^2}$, where $N$ is the number of elements clustered. The Adjusted RI (ARI) is then computed as: $ARI = \frac{RI - \mathbb{E}[RI]}{\max(0, 1 - \mathbb{E}[RI])}$. ARI ensures that a random label assignment will get a value close to zero (which RI does not).

3. **Fowlkes-Mallows Score (FMS)**: It is computed as: $FMS = \frac{TP}{\sqrt{(TP+FP)(TP+FN)}}$, where $TP$ is the number of pairs that belong to the same clusters in both the ground truth, as well as the predicted cluster labels, $FP$ is the number of pairs that belong to the same clusters in the ground truth, but not in the predicted cluster labels, and $FN$ is the number of pairs that belong in the same clusters in the predicted cluster labels, but not in the ground truth.

4. **Clustering Score (CScore)**: It is computed as: $CScore = \frac{ARI + FMS}{2\timesARI + FMS}$.

It should be noted that while we use CGacc to compare the methods for all the datasets (Data 1-6), the remaining metrics are reported only for the datasets Data 4-6, where we have the ground-truth labels. Also, all the performance metrics take values in the range $[0, \infty)$, where a higher value indicates a better performance.

Also, for each method, the distance threshold used to obtain the results, and the corresponding number of detected as well as the number of correct groups obtained, are reported in Table 6. From Table 6, we observed that our method is capable of detecting more number of color groups with a usually higher precision, when compared to its competitors.

In Table 7, we report the performance of all the compared self-supervised methods (including ours) on all the datasets, with respect to the different metrics against varying values of the distance threshold used in the Agglomerative clustering.