Contrastive Learning for Multi-Label Classification

Vladimir Zaigrajew
Wroclaw University of
Science and Technology
Wroclaw, Poland
`vladimirzaigrajew@gmail.com`

Maciej Zieba
Wroclaw University of
Science and Technology,
Tooploox
Wroclaw, Poland
`maciej.zieba@pwr.edu.pl`

Abstract

Contrastive learning is one of the most popular methods in self-supervised learning in recent years, achieving state-of-the-art performance in unsupervised training of deep neural networks for computer vision tasks. However, these methods mainly deal with binary classification, leaving out multi-label tasks, where additional dependency information between classes is needed to achieve satisfying performance. Although several proposed methods in contrastive learning have been developed to learn complex cross-label structures, they rely on specific architectures that introduce constraints and computational complexity. In this work, we propose a novel approach to contrastive supervised learning for multi-label classification (MultiSupCon). The main contribution is a new loss function that allows us to gain knowledge about the degree of label overlap between pairs of samples. We analyze our MultiSupCon loss to achieve best performance on CelebA dataset with top-1 mAP of 73.9% on the TResNet-M model, which is 2.8% above the best results for compared contrastive learning methods reported for this architecture.

1 Introduction

One of the recently popular techniques in machine learning and more specifically in DNN, is the use of contrastive learning. Contrastive learning is under the domain of (deep) metric learning [4], which is within the scope of representation learning [1]. Representation learning deals with training ML algorithms to learn useful representations, specifically those that are interpretable, have latent features or can be used for transfer learning. The goal of contrastive learning is to learn an embedding space in which similar data samples lie close together while dissimilar ones are far apart.

For multi-label problem, however, these methods fundamentally disregard the complex topological structure between objects. This has stimulated research into approaches to capture and explore label correlations in different ways. This stimulation has also spurred considerations in one of the recently popular methods of contrastive learning, known for great impact in the self supervised learning (SSL) domain, to be used for multi-label classification. One notable approach was to move contrastive learning from SSL to supervised training, which resulted in an architecture such as SupCon [6]. But this method was adapted to supervised learning for multi-class rather than multi-label classification which resulted in ignoring the complex topological structure between objects. However, the loss given in that article exhibits interesting properties which, with a slight modification, can be transformed to solve multi-label problem without losing knowledge of the dependencies between classes.

The aim of this work is to create a new approach to contrastive learning for multi-label classification problem that is not dependent on a particular architecture. We rely on the loss function proposed in SupCon for supervised learning and modify it to make it more suitable for multi-label problems. However, adjusting the loss appropriately is a complex task, and the multi-label problem also provides a new challenge: how to maintain relationships between objects and how to determine which samples

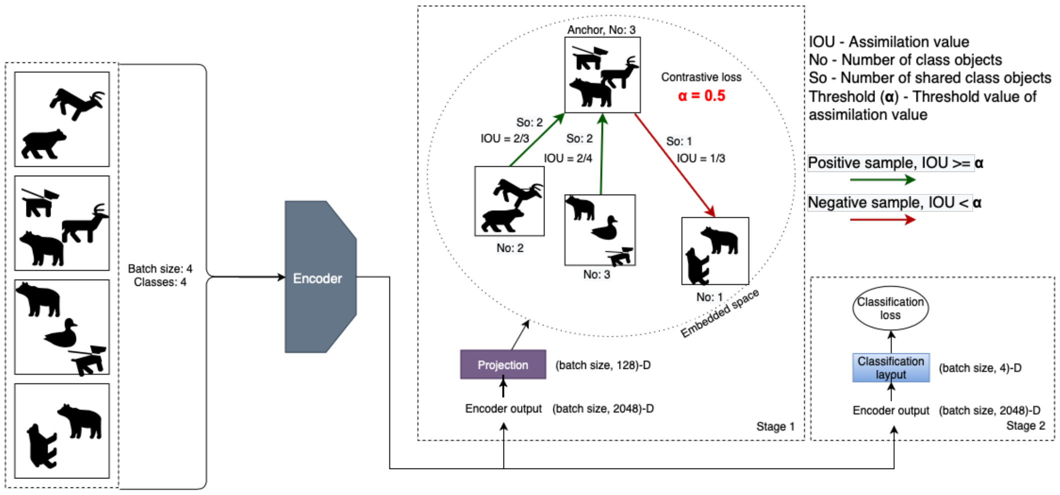


Figure 1: Architecture of multi-label supervised contrastive framework.

are considered similar and which are not. By creating the new loss function, we want to show that with its help we can train neural networks which will be able to learn appropriate patterns that will contain information about the correlation between classes.

In the paper, we make the following contributions: 1) we propose a novel loss function for contrast learning training network to tackle multi-label classification; 2) we show, that our novel approach has beaten reference contrast learning methods by a large margin of 5.6% mAP; 3) we show that our novel loss has better manipulation and separation capabilities then previous contrastive learning methods in representation space.

2 Out Approach

General Architecture. In Figure 1 we provide a general architecture of our approach. Our method is derived from supervised contrastive learning (which is structurally similar to self-supervised contrastive learning as in SimCLR [2]) and adapted for multi-label classification. As in SupCon pipeline, we first apply double-augmentation to obtain a batch size twice bigger to ensure at least one similar class object for every sample in the original batch. We use the proposed augmentation technique $Aug(\cdot)$, where for any given input $\mathbf{x}$, we generate two random augmentations, $\tilde{\mathbf{x}} = Aug(\mathbf{x})$, obtaining a different representation *view* of the sample, where each *view* is carrying some subset of the information from the initial sample. Next, each of the augmented examples is transformed by *Encoder network, Enc*$(\cdot)$, that maps $\mathbf{x}$ to a representation vector $\mathbf{r} = Enc(\mathbf{x}) \in \mathbb{R}^{D_E}$ ($D_E = 2048$ is the size of the representation vector before the final layer in the M-TResNet [10] model, which has been used in all our experiments in this thesis). The batch of encoded examples is then passed through a projection network that maps $\mathbf{r}$ to a representation vector $\mathbf{z} = Proj(\mathbf{r}) \in \mathbb{R}^{D_P}$. We instantiate $Proj(\cdot)$ as a multilayer perceptron [3] with a single hidden layer of size 2048 and an output vector of size $D_P = 128$. We normalize the output of this network so that it lies on the unit hypersphere, which allows us to use the inner product to measure distances in the projection space. Further, based on the value of the labels' *intersection of the union (IOU)* and its threshold (hyperparameter), we select which samples should be accepted as an anchor. Further, we use the selected anchors to calculate **Multi-Label Contrastive Loss** for training. At inference, we discard the $Proj(\cdot)$ to maintain the original neural network, using the already trained $Enc(\cdot)$. We attach the Classification layer and train it with a frozen $Enc(\cdot)$ on a multi-label classification task.

Contrastive Loss Functions. For a set of N randomly sampled sample/label pairs $\{\mathbf{x}_k, \mathbf{y}_k\}_{k=1...N}$, the corresponding batch used for training is made up of $2N$ pairs $\{\tilde{\mathbf{x}}_l, \tilde{\mathbf{y}}_l\}_{l=1...2N}$, where $\tilde{\mathbf{x}}_{2k}$ and $\tilde{\mathbf{x}}_{2k-1}$ are two random augmentations often referred to as "views" of $\mathbf{x}_k$ ($k = 1...N$) ($\tilde{\mathbf{x}}_{2k-1}, \tilde{\mathbf{x}}_{2k} \sim Aug(\mathbf{x}_k)$) and $\tilde{\mathbf{y}}_{2k-1} = \tilde{\mathbf{y}}_{2k} = \mathbf{y}_k$. In the remainder of this work, we will refer to the set of N samples as a "batch" and to the set of $2N$ augmented samples as a "multi-view batch".

Self Supervised Contrastive Loss The most popular self supervised contrastive loss, which has been used in SimCLR and also in previous work [12], is N-Pairs loss also known as InfoNCE [9] and NT-Xent [11]. Looking at a multi-view batch, we have $2N$ data points, where there are one positive and $2N-2$ negative pairs. The loss presents as:

$$\mathcal{L}^{self} = \sum_{i\in I} \mathcal{L}_i^{self} = -\sum_{i\in I} \log \frac{\exp(\mathbf{z}_i \cdot \mathbf{z}_{j(i)}/\tau)}{\sum_{a\in A(i)} \exp(\mathbf{z}_i \cdot \mathbf{z}_a/\tau)} \tag{1}$$

where $\mathbf{z}_l = \mathit{Proj}(\mathit{Enc}(\tilde{\mathbf{x}}_l)) \in \mathbb{R}^{D_P}$, symbol $\cdot$ denotes the inner (dot) product, and $A(i) = I \setminus \{i\}$. Index i is called an anchor, index $j(i)$ is called a positive, and the remaining $2(N-1)$ indices $(k \in A(i) \setminus \{j(i)\})$ are negatives. Note that the loss presented consists, for each anchor i, of 1 positive pair and $2N-2$ negative pairs.

Supervised Contrastive Loss. To take advantage of the presence of labels and make more than one sample belong to the same class, the authors of SupCon presented their novel loss function. Initially, they proposed two different loss functions that could use supervised data, however based on Jensen's inequality and further testing authors concluded that one loss function is much better than the other. Superior loss function is presented:

$$\mathcal{L}_{out}^{sup} = \sum_{i\in I} \mathcal{L}_{out,i}^{sup} = \sum_{i\in I} \frac{-1}{|P(i)|} \sum_{p\in P(i)} \log \frac{\exp(\mathbf{z}_i \cdot \mathbf{z}_p/\tau)}{\sum_{a\in A(i)} \exp(\mathbf{z}_i \cdot \mathbf{z}_a/\tau)}, \tag{2}$$

where $P(i) = \{p \in A(i) : \tilde{\mathbf{y}}_p = \tilde{\mathbf{y}}_i\}$ is the set of indexes of all the positives in the multi-view batch distinct from i, and $|P(i)|$ is its cardinality.

Multi-Label Contrastive Loss. In the case of multi-label problem, the loss, used in SupCon, shows one major drawback. Consider the situation when two sets of labels are regarded as being in the same class. This will only occure when the sets of labels are equal. However, it can rarely happen that there are identical sets of labels in the whole batch or even in the dataset, which can lead to the possibility of losing the benefit of generalization to any number of positives. Therefore, in this work, we define $N(i) = \{p \in A(i) : s_{p,i} \geq c\}$, where $c \in [0,1]$ is a hyperparameter representing the threshold value that indicates how similar sets of labels should be considered in CL. We define supervised multi-label loss as

$$\mathcal{L}_{ml}^{sup} = \sum_{i\in I} \mathcal{L}_{ml,i}^{sup} = \sum_{i\in I} \frac{-1}{|N(i)|} \sum_{p\in N(i)} s_{i,p} \cdot \log \frac{\exp(\mathbf{z}_i \cdot \mathbf{z}_p/\tau)}{\sum_{a\in A(i)} \exp(\mathbf{z}_i \cdot \mathbf{z}_a/\tau)} \tag{3}$$

where $s_{i,p}$ represents the weight in the aggregate loss (assimilation value); if the sets of labels are identical, then we have $s_{i,p} = 1$, otherwise $s_{i,p} < 1$. If we assume $c = 0$, then $N(i) = A(i)$ is meaningful, since completely different examples (with no common labels) will receive a weight equal to $s_{i,p} = 0$.

Assimilation Function. The last part is to determine how to obtain the assimilation value $s_{i,p}$. This value should indicate the degree of overlap between the two sets of labels, with the value of 0 meaning zero overlap and 1 giving a perfect match. In their paper [8], the authors seek to create an intuitive and effective similarity metric for the multi-label problem. They propose a new version of the triplet loss similarly to the approach taken in our work using a value to indicate the level of overlap between labels. The authors calculate a similarity measure based on the Jaccard index listed above:

$$s_{ij} = \frac{\sum_{n=1}^{|L|} \min(y_{in}, y_{jn})}{\sum_{n=1}^{|L|} \max(y_{in}, y_{jn})} \tag{4}$$

where $|L|$ is the size of the considered label space, and $y_{in} \geq 0$ is the number of occurrences of the n-th label for the i-th example. This similarity measure has the advantage of being in the interval $[0,1]$, is robust to the sparsity of the label vectors and treats both binary (for multi-label) and non-binary (for multi-object) label vectors. For the same reason, this similarity measure is the chosen assimilation function for calculating $s_{i,p}$ values for the new loss function proposed in this work.

Table 1: Classification results on downstream task for contrastive learning methods. MultiSupCon methods are indexed by threshold value for the assimilation parameter. The best results for each metric are in **bold**.

Method	mAP [%]		F1 score [%] (val)		
	train	val	micro	macro	sample
MultiSupCon$_{0.1}$	66.5	65.1	72.1	54.4	74.1
MultiSupCon$_{0.3}$	72.0	71.1	74.3	61.2	76.2
MultiSupCon$_{0.5}$	**74.3**	**73.9**	**75.5**	**64.6**	**77.5**
SupCon	69.4	68.3	72.5	57.7	74.7
SimCLR	67.6	66.8	71.1	56.0	73.5

3 Experiments

The purpose of the experiment is to compare our new loss functions MultiSupCon with derived contrast learning methods (SimCLR and SupCon) using benchmark datasets. We use standard data partitioning for the analysis, and the prediction target are attributes in the 64x64 image version of the CelebA dataset. We use the structures proposed by the SupCon authors applied to the TResNet-M model. The training pipeline consists of two stages, in the first we train our model with contrastive learning methods, where the last classification layer is removed making our model an encoding network through which we pass the batch for forward propagation. The batch is then propagated through a *Proj*$(\cdot)$, which is built as a multilayer perceptron [3] with a single hidden layer of 2048 and an output vector of $D_P = 128$. to obtain a vector of normalized embeddings. The modified network is trained with contrastive learning for 80 epochs on the CelebA dataset training set. After training the model with contrastive learning, we transfer the knowledge to the downstream task, where the network is now trained with cross-entropy loss [14] for multi-label attribute classification on images. We drop the *Proj*$(\cdot)$ and reattach the final classification layer to the frozen *Enc*$(\cdot)$, thus reconstructing the original neural network. Due to computational limitations, we trained models in the first stage with a smaller batch size than 512 on a GeForce RTX 2080 Ti GPU, and the others on an Nvidia Tesla V100 GPU. In the lower stage task, the experiments were conducted on a GeForce RTX 2080 Ti. For more information on implementation details, see Appendix A.

The results of preliminary experiments are shown in Tab. 1. We compared our approach with prior contrastive learning methods and evaluated them using a downstream task with the same parameters in each method. The proposed approach outperforms both methods in every metric with significant improvement. In addition, the threshold parameter is important in terms of tuning to get the best performance, this can be seen in the table, where at small value of this parameter the model performs worse than prior methods. The SupCon method is a variation of our method, where the threshold value is equal to 1.0 (maximum), because the overlap between two labels must be identical to consider two samples as similar as it is in SupCon, concluding the tuning of this hyperparameter is significant for performance.

Fig. 2 in Appendix B illustrates on 3D scatter plots how the approaches from contrastive learning stage have modeled CelebA dataset in an embedded space of *Proj*$(\cdot)$. MultiSupCon methods, show much more separation in the embedded space, where we can identify three main clusters of points in each version of the method. In contrast, the SimCLR and SupCon structures are closer to the initial shape of the dataset. This indicates greater manipulation possibilities in the embedded space, which can lead in donwstream tasks to better classification, as we have more separable space.

4 Conclusion

In this work we present a novel loss function for contrastive learning for multi-label problems that could be used in semi-supervised training to obtain better performance. The loss function presented can be easily applied to almost any model where embeddings can be extracted from the pre-final layer. This model shows promising results even when training on uncorrelated labels with large average label lengths, which should produce less information from our loss function. In the further work we want to evaluate our method on well known multi-label datasets such as MS-COCO or NUS-WIDE. We noticed that the recent paper [13] has a similar approach to ours, exploiting the more correlated label sets of ImageNet, DeepFashion In-Shop, iNaturalist and ModelNet40, on which we would like to also train and compare our novel loss.

References

[1] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[3] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[4] Mahmut Kaya and Hasan Şakir Bilge. Deep metric learning: A survey. *Symmetry*, 11(9):1066, 2019.

[5] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[6] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in Neural Information Processing Systems*, 33:18661–18673, 2020.

[7] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[8] Jonathan Mojoo and Takio Kurita. Deep metric learning for multi-label and multi-object image retrieval. *IEICE Transactions on Information and Systems*, 104(6):873–880, 2021.

[9] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[10] Tal Ridnik, Hussam Lawen, Asaf Noy, Emanuel Ben Baruch, Gilad Sharir, and Itamar Friedman. Tresnet: High performance gpu-dedicated architecture. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1400–1409, 2021.

[11] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. *Advances in neural information processing systems*, 29, 2016.

[12] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3733–3742, 2018.

[13] Shu Zhang, Ran Xu, Caiming Xiong, and Chetan Ramaiah. Use all the labels: A hierarchical multi-label contrastive learning framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16660–16669, 2022.

[14] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems*, 31, 2018.

Appendix

A Pipeline implementation

The setup for the experiments is based on the GitHub repository of SupCon[1], where a pipeline for testing SupCon and SimCLR on binary classification is provided. The implementation is done in the PyTorch[2] framework for DNN in Python language. The SupCon repository is lacking an implementation of the TResNet family of architectures, as the repository provides only ResNet models, as well as unprepared to run on CelebA dataset. Additionally, the implementation of defined validation metrics in the aspect of multi-label classification is needed. With the SupCon repository modified with those changes, and with the introduced new loss function, the repository for research is completed.

For the TResNet architecture, we followed the implementation proposed in the GitHub repository of ML_Decoder[3] (where the EMA module is also derived from), with four models of this family presented in the repository: TResNet-S, TResNet-M, TResNet-L, TResNet-XL. Additional implementations were necessarily to add a $Proj(\cdot)$ and a classification head to these networks, but having already prepared the code for ResNet in the SupCon repository, we only adjusted the TResNet models to have a similar structure to the ResNet models. An additional method, adopted to speed up training, was to use mixed precision.

The next configuration step necessary to create the pipeline was to prepare the dataset and adjust the code to be able to train on it. The structure of the CelebA data file, which is available for download from the official site, is:

- imgalignceleba.zip – the cropped and aligned version of images,
- listevalpartition.csv – recommended partitioning of images into training (80%), validation (10%) and testing (10%) sets,
- listbboxceleba.csv – bounding box information for each image, where coordinates and respective attributes are provided,
- listlandmarksalign_celeba.csv – image landmarks (left eye, right eye, nose, left mouth, right mouth) and their respective coordinates,
- listattrceleba.csv – attribute labels for each image with "1" indicating presence of the attribute, while "-1" absence of it.

From this data structure, we used only the imgalignceleba.zip file, where the aligned and cropped versions of the images with the size of 64x64 are found, along with listattrceleba.csv, where the list of labels/attributes for each image is provided. In addition, we used listevalpartition.csv to partition the dataset into the train, val, test as recommended by the authors. This was also suggested by our exploratory analysis, which confirmed that the proposed partitions are evenly distributed. In the code, a dataset class was created in PyTorch in order to use the given dataset structure to derive images and labels for them. Finally, to complete the part of the pipeline that will be responsible for handling CelebA dataset, it is necessary to define the augmentation for the data. Heavy data augmentation is needed to create a noisy version of the sample to increase heterogeneity, or more precisely, the distance between positive pairs. In our case, we followed the augmentation techniques that were used in SupCon, which included:

1. RandomResizedCrop – croping a random portion of image and resizing it to the size of 64x64 to insure the same size across all the images,
2. RandomHorizontalFlip – horizontally flipping the given image randomly with a probability 0.5,
3. RandomApply of ColorJitter – randomly change the brightness, contrast and saturation from the range [0.6, 1.4], as well as hue from the range $[-0.1, 0.1]$; these changes will occur with a probability 0.8 for each image,

[1]`https://github.com/HobbitLong/SupContrast`

[2]`https://pytorch.org`

[3]`https://github.com/Alibaba-MIIL/ML_Decoder`

4. RandomGrayscale – randomly convert image to grayscale with a probability 0.2.

These augmentation techniques were set for the train partition in each stage of the pipeline. For the val partition, where we tested stage 2, we only used resizing to make sure the images had the size of 64x64.

The most important part of the implementation was the inclusion of our loss and a small adjustment of SupCon and SimCLR losses. In all the loss functions tested, we added the multiplication part by the assimilation value matrix. For SimCLR, this matrix consists of ones on the diagonal and zeros elsewhere for the original and augmented versions of the samples to ensure that the assimilation values do not change the original SimClR loss formula. For SupCon and MultiSupCon, we calculate the assimilation matrix using Eq. 4. For SupCon only values equal to 1 (where the sample labels are the same) are left, and all other values are changed to 0. For MultiSupCon this assimilation matrix is only threshold conditioned, where values equal and higher of this threshold are preserved as they are, and the rest are changed to 0, as stated in Eq. 3.

Finally, we added validation visualization in stage 1 in the form of a scatter plot of the $Proj(\cdot)$ output for samples from the entire training set. For this, the UMAP dimension reduction algorithm [7] was used, which maps a 2048D vector to a 3D vector. For stage 2, the validation of the train partition consisted of calculating the mAP value (the algorithm was in the ML_Decoder repository) and validating the val partition using the mAP and F1 score metrics. After each epoch the scores of validation metrics were gathered and saved in Weights & Biases[4] experiments tracker framework. To ensure reproducibility, a seed of 31 was set for the PyTorch and NumPy libraries.

In each case, the model was trained for 80 epochs using a stochastic gradient optimizer (SGD) with learning rate of 0.5, moment of 0.9 and a weight decay of $1e{-}4$ in the contrative learning step, where $Enc(\cdot)$ and $Proj(\cdot)$ were trained with the contrastive learning loss. In addition, cosine annealing was applied with three step rate decay for epochs (50,60,70) with value of 0.1 as in the SupCon training pipeline. In addition, models that were trained on a batch size larger than 256, were also implemented with a warm-up as suggested in [5], where good techniques and tips for training on a large batch size were presented. Each warm-up started with a learning rate of 0.1 and increased gradually over 10 epochs until a learning rate of 0.5 was reached. After each epoch, validations were performed in the context of visualizing the output from $Proj(\cdot)$ on the entire dataset in a scatter plot using UMAP. In the second stage, $Proj(\cdot)$ was discarded and a classification layer was added to the frozen $Enc(\cdot)$. In this stage, the new model was trained for 51 epochs with a batch size of 256 with a similar set of parameters and with cross-entropy loss. Here, SGD was set with a learning rate of 1, a moment of 0.9 and with the absence of weight decay.

B Limitations

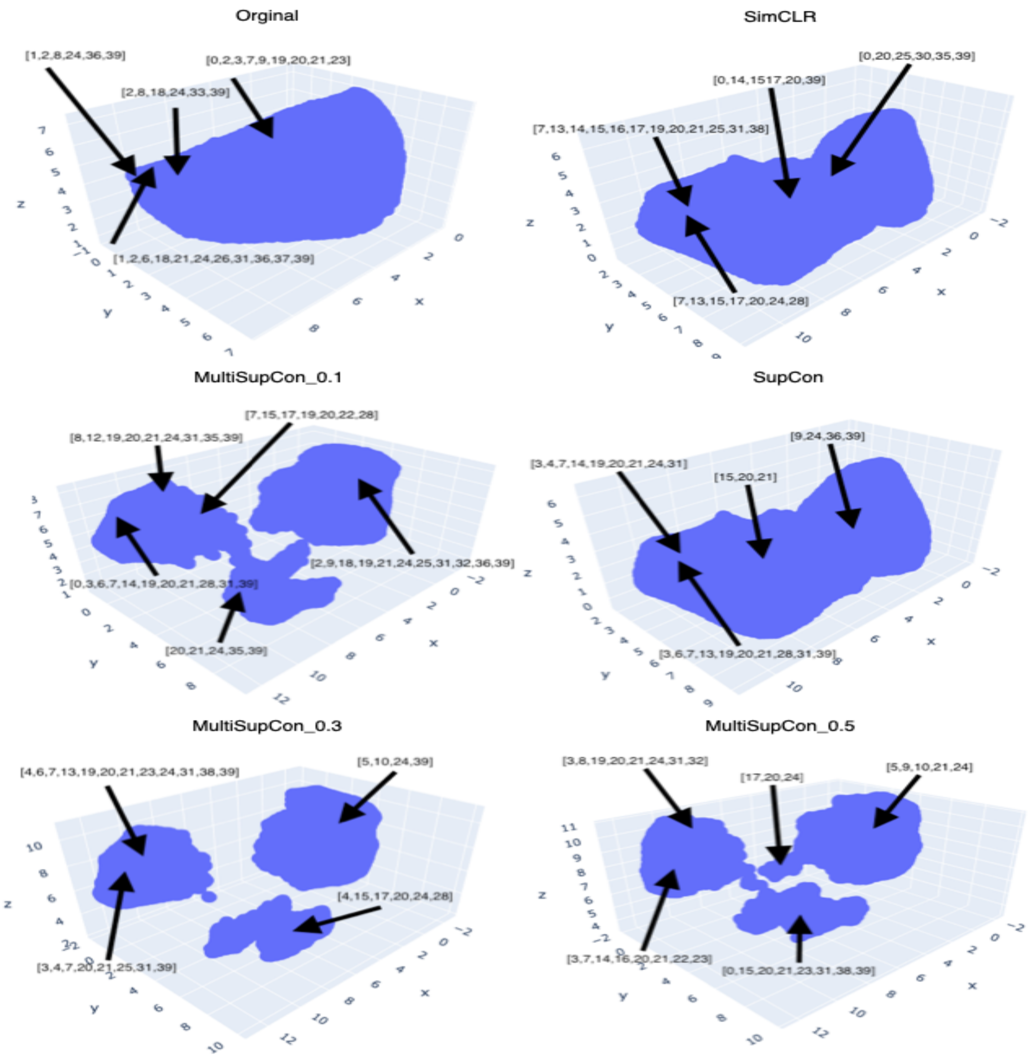
The findings of this study have to be seen in light due to computational constrains. Due to the batch size ablation studies recommended in SupCon (results presented for the best value, but the studies were conducted for a wide range of values) and the lack of a sufficient GPU, we decided to test only on CelebA (64x64) dataset to conduct full ablation studies.

C Visualization of embedded spaces

To verify what models learned in contrastive learning stage, we visualized the embedding space of initial training step and final epoch for each prior method and each trained variants of our MulitSupCon method with a bottom index indicating the threshold value in the loss. Each visualization was done with UMAP transforming a 128 dimensional vector to a 3 dimensional vector to visualize it with a 3d scatter plot. Each point in the plot has a specimen label, which we selectively visualize on each plot. The sample point label list contains the class numbers that the samples was annotated with.

[4]`https://wandb.ai/site`

Figure 2: Visualization of embedded spaces for contrastive learning methods with UMAP and scatter plots.


Orginal
[1,2,8,24,36,39]
[0,2,3,7,9,19,20,21,23]
[2,8,18,24,33,39]
[1,2,6,18,21,24,26,31,36,37,39]
SimCLR
[0,20,25,30,35,39]
[0,14,1517,20,39]
[7,13,14,15,16,17,19,20,21,25,31,38]
[7,13,15,17,20,24,28]
MultiSupCon_0.1
[7,15,17,19,20,22,28]
[8,12,19,20,21,24,31,35,39]
[2,9,18,19,21,24,25,31,32,36,39]
[0,3,6,7,14,19,20,21,28,31,39]
[20,21,24,35,39]
SupCon
[9,24,36,39]
[3,4,7,14,19,20,21,24,31]
[15,20,21]
[3,6,7,13,19,20,21,28,31,39]
MultiSupCon_0.3
[4,6,7,13,19,20,21,23,24,31,38,39]
[5,10,24,39]
[4,15,17,20,24,28]
[3,4,7,20,21,25,31,39]
MultiSupCon_0.5
[3,8,19,20,21,24,31,32]
[17,20,24]
[5,9,10,21,24]
[3,7,14,16,20,21,22,23]
[0,15,20,21,23,31,38,39]
x
y
z