MIX'EM: Unsupervised Image Classification using a Mixture of Embeddings

Ali Varamesh, Tinne Tuytelaars
ESAT-PSI, KU Leuven
{ali.varamesh,tinne.tuytelaars}@esat.kuleuven.be

Abstract. We present MIX'EM, a novel solution for unsupervised image classification. Our model generates representations that by themselves are sufficient to drive a general-purpose clustering method to deliver high-quality classification without supervision. MIX'EM integrates an internal mixture of embeddings module into the contrastive visual representation learning framework to disentangle the representation space at the category level. It generates a set of embeddings from a visual representation and mixes them to construct the contrastive loss input. Parallel to the contrastive loss, we introduce three techniques to train MIX'EM and avoid a degenerate solution; (i) we maximize entropy across mixture components to diversify them, and (ii) minimize component entropy conditioned on instances to enforce a clustered embedding space. Applying (i) and (ii) lead to the emergence of semantic categories through the mixture coefficients, making it possible to (iii) apply an associative embedding loss to enforce semantic separability directly. Subsequently, we run K-means on the representations to acquire semantic classification, which outperforms the state-of-the-art by a large margin. We conduct extensive experiments and analyses on STL10, CIFAR10, and CIFAR100-20 datasets, achieving 78%, 82%, and 44% accuracy, respectively. Essential to robust high accuracy is using MIX'EM to initialize K-means. Finally, we report impressively high accuracy baselines (70% on STL10) achieved solely by applying K-means to the "normalized" representations learned using the contrastive loss.

1 Introduction

In the span of a few years, supervised image classification has made remarkable progress and even surpassed humans on specific recognition tasks [1]. Its success depends on few important factors, namely, the stochastic gradient descent method to optimize millions of parameters, GPUs to accelerate high-dimensional matrix computations significantly, and access to vast amounts of manually annotated data [2]. Although a particular optimization method or high-performance hardware is not theoretically essential for supervised methods, access to labeled data is. In fact, access to large-scale labeled data is vital if we want to get the top performance [3,4]. Considering the research trend in modern computer vision, one of the next major challenges is doing unsupervised image classification. That means eliminating the costly and not always feasible

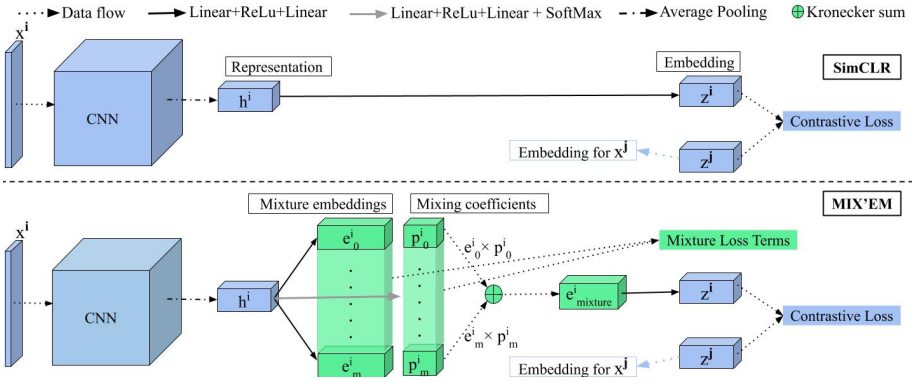


Fig. 1: Our proposed architecture with a mixture of embeddings module (bottom row), compared to the contrastive representation learning framework (top row). Components in green are devised in MIX'EM.

process of manual labeling [5,6]. In this context, visual representation learning has recently demonstrated great success in discarding manual labels by relying on self-supervision [7,8,9,10,11,12]. We argue that self-supervised representation learning is a bedrock for unsupervised recognition.

In self-supervised visual representation learning, a pretext (auxiliary) task provides a supervision signal (without manual labels) for training a representation encoder. Then, by training a linear classifier on representations generated by such an encoder (frozen), we can achieve high accuracy, close to that of an end-to-end trained fully supervised ImageNet classifier (76.5% vs. 78.3% top-1 accuracy, respectively) [11]. One particularly successful approach for designing a pretext task is to treat each image instance in a dataset as a unique class and use a supervised technique for training [9]. However, it is computationally prohibitive to implement this idea at the scale of millions of images. So, noise-contrastive estimation [13] is typically used to simplify the task into the classification of different views of images. In the new formulation, a reference image's transformations are classified as similar and any other sample dissimilar [9,12,11]. Representations learned using this pretext task have even surpassed supervised representations' performance on some variants of object detection and segmentation tasks [12].

The fact that a linear layer, with very limited discriminative power [14], can deliver such high accuracy on the complex ImageNet classification task, signals the presence of very strong discriminative semantic clues in the representation. If so, we hypothesize that by just knowing the expected number of classes, an off-the-shelf clustering method must achieve high quality as well. Indeed, our experiments prove that K-means trained on representations generated by the recent SimCLR method [11], once normalized, achieve accuracy of 70% on STL10 , compared to 87% by a supervised linear classifier.

We have designed MIX'EM essentially by taking inspiration from this observation, and recent works [15,16,17,18,19,20,21] showing that mixture models can divide their input-output space in a meaningful manner without being directly supervised . Our goal in this work is to impose semantic structure on the self-supervised representations to boost clustering accuracy. In other words, we want to generate representations that are already highly clustered or disentangled. We incorporate a mixture of embeddings module in the contrastive visual representation learning framework [11], as illustrated in figure 1. The mixture components [22,23,24] are expected to specialise on embedding different semantic categories. For a given sample, each component should generate an embedding and predict how much it contributes to the mixture of the component embeddings.

For unsupervised image classification, semantic clustering has also been used in some recent works [25,26,27]. However, contrary to these methods, in MIX'EM, our goal is to impose structure on the representations and make them highly reliable so that an off-the-shelf clustering method with no extra effort can be trained to deliver high accuracy. MIX'EM takes advantage of the contrastive visual learning framework for guiding training a mixture model without interfering with its mechanisms (see table 3).

We introduce three key techniques to successfully train the MIX'EM module end-to-end, along with the representation learning loss. Using the contrastive loss only, a naive attempt to training would quickly converge to a degenerate solution that assigns all samples to a single component, bypassing all other paths. To avoid this issue and achieve high accuracy, (i) we maximize the entropy of coefficient distribution to diversify the mixture components; (ii) we minimize the entropy of components conditioned on the input to enforce separation of the embedding space; and enabled by (i) and (ii), (iii) we use an associative embedding loss [28,29] to directly enforce semantic coherence inter/intra mixture components. Figure 2 presents visualizations of the embeddings after plugging in each of the loss terms. The resulting representations significantly boost K-means' performance to 78% on the STL10 dataset, without interfering with the contrastive learning process. We summarise our contributions as follows:

- We propose MIX'EM to disentangle the visual representation space semantically at the category level, such that an off-the-shelf clustering method can be applied to acquire robust image classification without supervision.
- We introduce three techniques to successfully train MIX'EM in an unsupervised manner and avoid degenerate solutions.
- MIX'EM initializes K-means and achieves significantly higher accuracy. This eliminates the need to run K-means with multiple random initializations.
- We show that visual representations learned using contrastive loss, once normalized, already yield high accuracy semantic clustering using K-means and outperforms most existing works.

2 Related Work

Our work relates to a few different lines of research. It is the most related to the self-supervised representation learning, as our goal in the first place is to train a

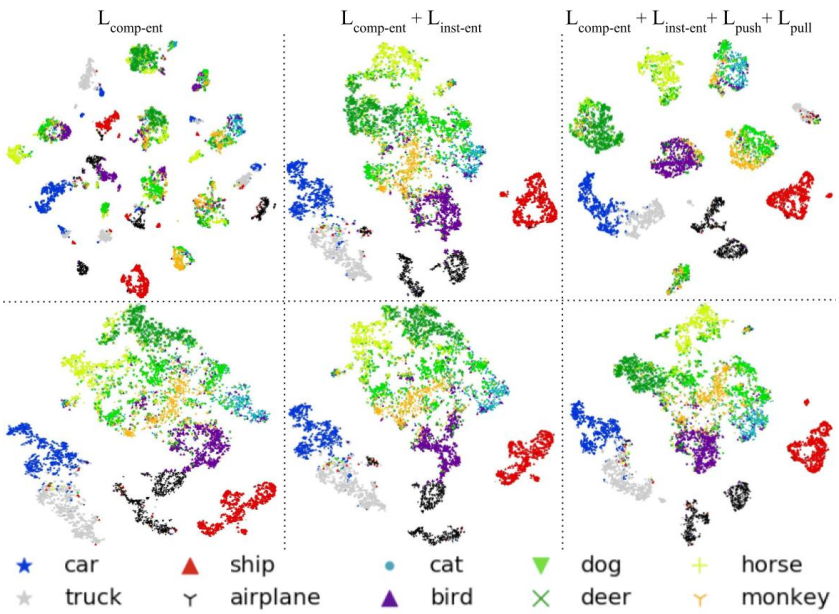


Fig. 2: tSNE visualization of embeddings learned by MIX'EM on STL10. Top row: embeddings from the dominant component ($e^i_{m'}$ s.t. $m' = \arg\max_{m'}(p^i_{m'})$); Bottom row: the final mixture embedding (z^i). By adding our loss terms (explained in 3.2), embedding space gets increasingly more disentangled at category level. Samples are marked and color coded based on their ground truth label. (see figure 3 for tSNE visualization of the representations).

better representation encoder without using manually labeled data. However, beyond that, we want the representations to be highly structured such that reliable semantic clustering is possible using an off-the-shelf clustering method. We develop our idea on the recent research [11,30] which empirically proves using noise contrastive loss [13] and heavy augmentation for visual representation learning outperforms other popular approaches, including mutual information maximization [31,7], generative models [32,33], image rotation prediction [8], predicting patch position [34], clustering [35], solving jigsaw puzzles [36], and image colorization [37]. In this work, we also advocate using noise contrastive loss for self-supervised representation learning. However, we are particularly interested in enforcing category-level semantic disentanglement of the representation space.

To the best of our knowledge, this is the first work to set out to impose semantic structure on self-supervised visual representations learned using the contrastive loss. We show that the representations generated by MIX'EM result in high accuracy semantic clustering only by applying K-means to them. Existing works on unsupervised image classification using self-supervised representations [27,26,25] should benefit from adapting our proposed module, as it is an internal module that can be plugged-in without altering the output mechanism.

There have been a few recent works with the same objective as ours, that is unsupervised image classification. IIC [25] is the best known, which directly generates semantic clustering assignments using a deep network. Its loss function maximizes the mutual information between augmented versions of the same image based on the cluster assignment. The intuitive goal is to force the clustering to be decided based on invariant information across different views of an image. [26] proposes a max-margin clustering criterion for simultaneous clustering and representation learning, such that clustering confidence is the highest based on a defined confidence index. Finally, simultaneous to our work, [27] proposes a pipeline with multiple stages that relies on the k- neighbor method to extract samples from supposedly same semantic categories from contrastive based representations. They use the sampled to train a clustering network and later for classification, treating clusters as pseudo labels. None of these works concern improving the representations directly in terms of category-level disentanglement in an unsupervised fashion.

Our work also relates to clustering-based approaches for deep self-supervised representation learning [35,38,39,40]. These models devise a branch in a deep network for clustering; that generates pseudo labels for training another branch of the network for a classification task. The training process either iterate between the two stages until it converges [35,38] or does it simultaneously [41]. Generating high-level semantic labels using clustering, however, is not the goal in this line of work. Instead, they combine clustering and classification in order to build a pretext task for representation learning. Often the best representations are achieved with over-clustering. For example, [35] achieves the best mAP on the Pascal VOC 2007 object detection task when the representation is learned using a 10000-way clustering stage.

Finally, [42] is also related to our work, where representations are split into "shared" and "exclusive" parts. It maximizes mutual information for the shared and minimizes for the external component across paired samples. However, they use supervision to pair images for training. Moreover, the work is not concerned with semantic clustering. Based on their problem formulation and results, the disentanglement focuses a foreground-background separation.

3 Method

In this section, first we review the contrastive learning framework as proposed in SimCLR [11] (the principles are similar to [12,9]). Next, we show how to incorporate a mixture of embeddings module in the contrastive learning framework.

3.1 Contrastive learning of visual representation

Contrastive learning of visual representations is built upon the intuition that different transformations of an image should have the same characteristics, which identifies them as bearing the same semantics and distinguishable from another category. In practice, this means that given a dataset with images containing a single dominant object (like ImageNet or CIFAR10/100), an ideal encoder should

map different augmented versions of an image to a very compact neighborhood. This interpretation is equivalent to considering every image in a dataset as a distinct class and training a classification network using cross-entropy loss [9]. This way, there will be as many class labels as the number of samples in the dataset. However, this is not a scalable formulation, and in practice, the task is approximated using noise contrastive estimation [13]. A simplified version of the solution, SimCLR, presented in [11], is based on doing instance classification in a given mini-batch of pairs corresponding to different views of images. Before describing MIX'EM, we first review the formulation developed in SimCLR.

In SimCLR, the goal is to train an encoder to generate visual representations. We denote the encoder with $h^i = f(x^i)$, where $x^i \in D$ is an RGB image from the unlabeled dataset D. Encoder f is implemented using a deep convolutional neural network. h^i is the representation intended to be used by downstream tasks. To train the encoder, noise contrastive learning is applied to h^i to pull together similar images in the representation space. Chen et al. [11] show that, before computing the contrastive loss, applying a further non-linear layer to h^i to map it into another latent space Z results in significant improvement. Later, this has been confirmed to be beneficial to the MOCO model, which also uses contrastive loss [30]. We denote the extra non-linear layer by g in $z^i = g(h^i)$.

TO train the encoder, given a random mini-batch of N images, $\{x^i\}_{i=1}^N$, every image is augmented twice using a sequence of random transformations to generate 2N samples $\{\hat{x}^i\}_{i=1}^{2N}$. Then, the similarity between every pair u and v of the 2N samples in the latent space Z is computed using function $sim(u, v) = z_u^T.z_v/||z_u||||z_v||$, where z_u and z_v are output of $g(f(.))$. Next, counteractive loss for a positive pair (i.e. two augmentations of the same image x_i) is implemented in form of cross entropy for 2N-1 way classification. Logits for cross-entropy are equal to the pairwise similarities of a given view $\hat{x}^i$ with its counterpart, and 2N-2 views from the remaining augmented samples. The cross-entropy is computed for both views of each of the N samples in the original mini-batch. The contrastive loss $l_c(i, j)$ for a positive pair x^i and x^j is shown in the Equ. (1), where τ is a temperature parameter [11]. The total contrastive loss $L_{contrast}$ is shown in Equ. (2).

$$l_c(i,j) = -\log \frac{exp(sim(\hat{x}^i, \hat{x}^j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} exp(sim(\hat{x}^i, \hat{x}^k)/\tau))} \tag{1}$$

$$L_{contrast} = \frac{1}{2N} \sum_{k=1}^{N} l_c(i,j) + l_c(j,i) \tag{2}$$

3.2 Mixture Embedding

SimCLR computes the contrastive loss after embedding the target representation into another space via the non-linear layer, g. In MIX'EM, we replace this layer with multiple parallel non-linear layers, each generating an embedding and a coefficient to determine how much each embedding contributes to the final

embedding, z^i. Figure 1 depicts the architecture of MIX'EM and how it differs from the regular contrastive representation learning pipeline. Given input x^i, and representation $h^i = f(x^i)$ generated by the encoder, our model replaces $z^i = g(h^i)$ with the function $z^i = g(\psi(h^i))$, where the function ψ is defined in Equ. (3). M in Equ. (3) indicates the number of mixture components that, in our experiments, is equal to the expected number of semantic categories. g_m is a non-linear layer, equal to g in structure, and specializes in generating samples for component m. Coefficient p_m^i indicates the probability that sample x_i is generated by the component m. Accordingly, P^i is the conditional probability distribution of the mixture components. It is computed from the representation h^i using a non-linear layer g_p and softmax activation.

$$\psi(h^i) = \sum_{m=1}^{M} p_m^i * g_m(h^i)) \qquad s.t. \qquad P^i = SoftMax(g_p(h^i)) \tag{3}$$

With the mixture module, we expect the network to distribute input samples across components, as this should make the task in hand easier to solve [22,18]. We expect each mixture component to generate embeddings for certain categories and guide the backpropagation process conditioned on an input. However, if we start training using the contrastive loss only, it will quickly learn a degenerate solution that assigns all samples to a single component, bypassing all other paths [38,35]. We devise three different loss terms to avoid such a degenerate solution and adequately train the network to meed our goals.

Entropy maximization across components In a degenerate solution, the distribution P^i provides the lowest information from an information theory point of view; always, a particular component is equal to one. However, ideally, we want the model to be more diverse in the assignment of the components. We expect it to be dependent on the input image, not to ignore it. Given that we do not have any means to directly supervise the mixture module, we can instead maximize the entropy of marginal of P^i, which would take the highest value when all components are equally probable. Although not all components are equally probable in practice, as we will show in the experiments, this term indeed avoids the degenerate solution. Moreover, the distribution would result in semantically meaningful components. That is, components will focus on different categories. We believe this is due to the simultaneous backpropagation of the contrastive loss, imposing minimal semantic regularization. In fact, without the contrastive loss, training would fail. The loss term for entropy maximization is shown in Equ. (4), and it is equal to the negative of entropy of P, $H(P)$.

$$L_{comp-ent} = -H(P) = \sum p_m \log p_m \qquad s.t. \qquad p_m = 1/N \sum_{i=0}^{N} p_m^i \tag{4}$$

Conditional component entropy minimization Maximizing entropy of marginal P diversifies the components. However, we would like to separate the

representation space based on the most discriminative aspect of objects. For a given image, ideally, we want one of the mixture components to have near full confidence so that it can be interpreted as an indicator of the true category. This, in turn, would mean reducing the entropy of the mixture components given an instance. We know that entropy would be minimized in practice if all probability mass is assigned to a single component. Therefore, given an image, we add a loss term that pushes the probability of the dominant component to the highest value, one. Equ. (5) shows the instance based entropy minimization loss term.

$$L_{inst-ent} = \sum_{i=0}^{N} 1 - max\{p_m^i\}_{m=1}^{M} \tag{5}$$

Associative embedding loss Both entropy-based loss terms above are principled techniques to guide the training. However, they do not directly take into account the semantics. Intuitively, samples' ideal assignment to the components should pick up on visual clues that minimize the distance between samples with the same dominant mixture component. At the same time, it should maximize the distance between samples with different dominant components. In a supervised setting, it is straightforward to implement a loss function like this; however, here we do not have access to semantic labels. The good news is that just training MIX'EM with $L_{comp-ent}$ and $L_{inst-ent}$ would result in each component specializing in one category. In quantitative words, evaluating MIX'EM by treating the dominant component index as cluster label, on STL10, we get an accuracy of 73% (row (3) of the fourth column in table 1.

We introduce a third loss term to enforce semantic coherence by relying on the index of the dominant component as a pseudo-ground-truth label. This loss, called associative embedding, is inspired by the work of Newell et al. [28,29] on scene graph generation and human pose estimation. Using the dominant component index as the class label, we want to pull the embeddings assigned to a component as close as possible to each other. We implement this by minimizing the distance between sample embeddings and average embedding for a component, which is the average of embeddings by each component m on samples with m as their dominant component. Simultaneously, we want to push the average embedding of different components away from each other. We implement this by directly maximizing the pairwise distance of the average embedding of components. Equ. (6)- (8) shows formal specification of pull loss and push loss. Note that E_m is vital for both push and pull losses, and we are able to compute it only by means of using the dominant components in MIX'EM.

$$\mu_m = \frac{1}{|E_m|} \sum_{i \in E_m} e_m^i \qquad s.t. \qquad E_m = \{i \mid \arg\min_{m'}(p_{m'}^i) == m\} \tag{6}$$

$$L_{pull} = \frac{1}{M} \sum_{m=1}^{M} \sum_{i \in E_m} ||e_m^i - \mu_m||_2 \tag{7}$$

$$L_{push} = \frac{1}{M} \sum_{m=1}^{M} \sum_{m'=1}^{M} \mathbb{1}_{[m \neq m']} ||\mu_m - \mu_{m'}||_2 \tag{8}$$

Total loss Equ. (9) shows the total loss we use to train MIX'EM.

$$L_{total} = L_{contrast} + \lambda_1 L_{comp-ent} + \lambda_2 L_{inst-ent} + \lambda_3 L_{push} + \lambda_4 L_{pull} \tag{9}$$

3.3 Clustering to acquire classification

Once MIX'EM is trained, we apply the K-means algorithm to the representation to generate the final clusters. Our experiments show that K-means on representations would provide superior performance compared to when applied to embeddings. We also tried using other off-the-shelf clustering methods from the scikit-learn library, including spectral clustering [43], and found similar results. Moreover, the dominant mixture component index also provides a highly accurate classification, as we will show in the next section.

4 Experiments

We conduct experiments on three standard datasets, STL10 [44], CIFAR10 [45], and CIFAR100−20 [45] (CIFAR100 with 20 super-classes). STL10 is a subset of ImageNet and is particularly designed to benchmark self-supervised and unsupervised methods. It includes 100k unlabeled images, 5k labeled training images from 10 categories, and 8k labeled test images. The unlabeled set comes from a distribution different from the labeled data to simulate a real-world situation further. We implement our model in PyTorch library and use the scikit-learn library for doing K-means clustering. We use ResNet-18 [1] for all our experiments. On CIFAR10 and CIFAR100-20 we remove the second convolutional layer and max-pooling from the standard ResNet-18 because the input images are already in small resolution, 32x32 [46].

For each dataset, we first train the bare SimCLR with embedding dimension 256 and without the mixture embedding component for 600 epochs. On STL10 we use the train and unlabled splits, and on CIFAR10/100-20 we use the train split at this stage. We call this model "base SimCLR encoder." Next, we continue training with and without the mixture embedding module on with a lower embedding dimension (32 in our experiments) to investigate various aspects under equal conditions. A higher embedding dimension would improve results. In the rest of the paper, when referring to SimCLR, we mean the version trained further on the base SimCLR encoder. Training a base encoder first is not required; however, we do so to use the limited computational resources at our disposal efficiently. Training hyper-parameters, including learning rate, mini-batch size, learning schedule, the loss weight terms λ_1, λ_2, λ_3, and λ_4, augmentation configuration are determined by trying few different values for each dataset. Details for

the training setup are provided in the supplementary material. We believe that extending the experiments to a larger dataset like ImageNet would be straightforward, given enough computational resources.

For evaluation of the semantic clustering, we use the three widely applied metrics: clustering accuracy (ACC), normalized mutual information (NMI), and adjusted rand index (ARI). For ACC, to map the cluster indices into the ground-truth labels, we use the Hungarian method. We utilize the implementation of the metrics provided in the scikit-learn library.

Following the standard routine for unsupervised settings [25,39], we train MIX'EM on the combination of all labeled data and evaluate on the test split. To gain a better understanding of the role of data in unsupervised learning, in ablation studies, we provide evaluations for the case when test data is not used in training, and the case where unlabeled split too is used for training (on STL10). Unless specified otherwise, any results reported in the paper are obtained by taking the average of five separate trainings and are accompanied by the standard deviation.

Table 1: Ablation study of different loss terms on STL10 by gradually adding them. For K-means with the random initialization, we run it 50 times and report the best value w.r.t inertia. "Max Comp" means doing clustering by treating the dominant component's index as the final cluster label.

		ACC		**NMI**		**ARI**	
	Method	**K−means**	**Max Comp**	**K−means**	**Max Comp**	**K−means**	**Max Comp**
(1)	SimCLR	65.01 ± 1.79	–	66.2 ± 0.66	–	43.94 ± 1.09	–
(2)	+ representation normalization	69.95 ± 0.78	–	67.05 ± 0.33	–	55.37 ± 0.78	–
	MIX'EM						
(3)	$+L_{comp-ent}$	69.88 ± 0.17	31.83 ± 0.29	66.52 ± 0.08	20.61 ± 0.15	54.43 ± 0.23	12.15 ± 0.17
(4)	$+L_{inst-ent}$	76.21 ± 0.12	73.45 ± 0.03	67.27 ± 0.07	64.3 ± 0.04	60.16 ± 0.16	55.88 ± 0.05
(5)	+ MIX'EM initializes K−means	76.21 ± 0.09	73.45 ± 0.03	67.23 ± 0.08	64.3 ± 0.04	60.12 ± 0.11	55.88 ± 0.05
(6)	$+L_{push} + L_{pull}$	$\mathbf{77.76 \pm 0.08}$	68.44 ± 0.44	$\mathbf{68.03 \pm 0.07}$	64.08 ± 0.1	$\mathbf{61.35 \pm 0.1}$	54.66 ± 0.09
(7)	− MIX'EM initializes K−means	70.78 ± 0.19	68.44 ± 0.44	67.57 ± 0.42	64.08 ± 0.1	55.95 ± 0.16	54.66 ± 0.09

5 Results and discussion

We start by analyzing the effect of each loss term on STL10. Table 1 shows starting from SimCLR [11], gradually adding various MIX'EM loss terms consistently improves the performers. Row (2) illustrates the importance of normalizing the representations before applying K-means. Rows (4) vs. (5), and (6) vs. (7) show using MIX'EM to initialize the K-means improves significantly and makes training much faster (see 5.1). In figure 3 we present the tSNE [47] visualization of the representations for SimCLR and MIX'EM. In line with the quantitative evaluation, MIX'EM representations are more disentangled and constitute more compact clusters. Using contrastive loss alone does not adequately pull samples from similar categories to each other, resulting in a sparse space. The mixture module guides the training process via mixing coefficients, forcing the encoder to allocate more compact regions to different categories (unsupervised).

We also visualize the embedding space using tSNE. Figure 2 displays the tSNE visualization of the embedding for the dominant component (top row)

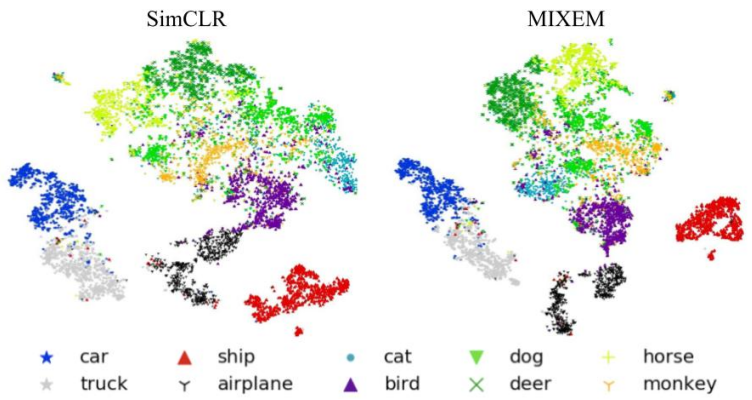


Fig. 3: tSNE visualization of representation on STL10 to illustrate the category-level disentanglement. Samples are marked and colored based on their true label.

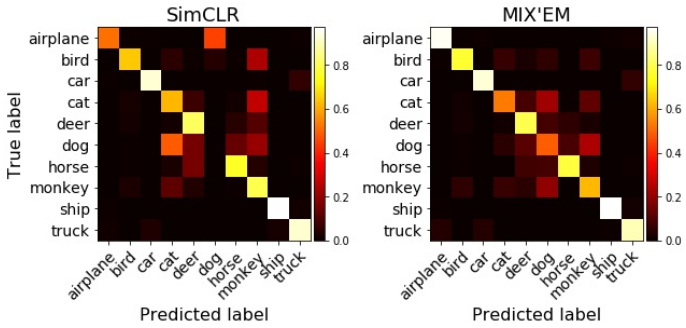
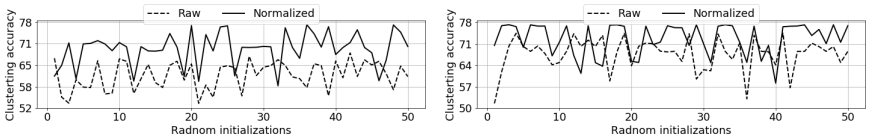


Fig. 4: Confusion matrix for prediction vs. true label on STL10.

and how in turn, it makes the mixture embeddings more disentangled (bottom row) as we gradually add the loss terms. For a category level analysis of MIX'EM, we show the accuracy confusion matrix in figure 4. The animal categories are more difficult to discriminate and benefit the most from MIX'EM. In the supplementary material, we provide visualizations to show how the categories are very hard to distinguish from some images.

5.1 Initializing K-means using MIX'EM

K-means lacks a robust initialization method. The standard procedure is to run the K-means many times using random initializations and choose the best one in terms of inertia, the cumulative distance of samples to their cluster center. We experimented with up to 100 runs and found 50 times to work the best on our models. However, this is not reliable or efficient in the long term. Figure 5 shows large fluctuations in accuracy across different runs. With random initialization, K-means is not guaranteed to find the best clustering within practical limits. Besides, in reality, the cost of running the model multiple times can be prohibitive.



(a) SimCLR (b) MIX'EM

Fig. 5: K-means accuracy with standard random initialization will fluctuate heavily over different runs on STL10 and does not guarantee converging to an optimal solution. The model corresponds to SimCLR (rows (1) and (2) in table 1)

Running K-means for 50 times on MIX'EM representations with the dimensionality 512 takes about 21 seconds for the relatively small STL10 dataset with 8k test images and ten classes. On 10k images of CIFAR100-20 and CIFAR100 with 20 and 100 classes, on average, it takes 112 and 221 seconds. Note that the issue can not be resolved by an efficient implementation (e.g.Mini-Batch K-means).

In MIX'EM, we use mean representation for each component, based on samples with the same dominant mixture component, to initialize K-means. Hence, we eliminate the need for many runs and consistently get higher accuracy. Rows (4),(5),(6) and (7) in table 1 show the performance with MIX'EM initialization. In particular, rows (6) and (7) illustrate how K-means with 50 random initialization can be far worse than using MIX'EM for initialization. A single run with MIX'EM initialization, on average, takes 0.27 , 1.5, and 3 seconds on STL10, CIFAR100-20, and CIFAR100, in order.

5.2 Comparison to the state-of-the-art

In table 2 we compare performance of MIX'EM to state-of-the-art. On the more challenging dataset of STL10, our model outperforms all other works by a large margin. On CIFAR 10,100-20 again, our model outperforms other works, except SCAN [27] (simultaneous to our work), that achieves better results when further trained with a classification objective. MIX'EM delivers results with very standard deviations, which would be of high importance in a real-world application. On SCAN standard deviation are as much as an order of magnitude higher on STL10 and CIFAR100-20. Since MIX'EM improves representations in terms of separability, SCAN should benefit from using it for the backbone representation encoder. On CIFAR100-20 for all models, the results are generally worse compared to other datasets. We believe this is due to the relatively confusing mappings of classes to super-classes. For example, "bicycle" and "train" both are mapped to "vehicles 1" and most animals are divided based on their size, rather than semantics.

5.3 Effect on contrastive representation learning

In MIX'EM, the contrastive loss is vital for successful training. This raises the question of how the mixture module influences the performance of representa-

Table 2: Comparison to the state-of-the-art.

Dataset	**CIFAR10**			**CIFAR100-20**			**STL10**		
Metric	**ACC**	**NMI**	**ARI**	**ACC**	**NMI**	**ARI**	**ACC**	**NMI**	**ARI**
DEC [48]	30.1	25.7	16.1	18.5	13.6	5.0	35.9	27.6	18.6
DAC [49]	52.2	40.0	30.1	23.8	18.5	8.8	47.0	36.6	25.6
DeepCluster [35]	37.4	–	–	18.9	–	–	65.6	–	–
ADC [50]	32.5	–	–	16.0	–	–	53	–	–
PICA [26]	0.561	0.645	0.467	–	–	–	0.592	0.693	0.504
IIC [25]	61.7	51.1	41.1	25.7	22.5	11.7	59.6	49.6	39.7
SCAN [27]	81.8 ± 0.3	71.2 ± 0.4	66.5 ± 0.4	42.2 ± 3.0	44.1 ± 1.0	26.7 ± 1.3	75.5 ± 2.0	65.4 ± 1.2	59.0 ± 1.6
SCAN [27] w/ classification	87.6 ± 0.4	78.7 ± 0.5	75.8 ± 0.7	45.9 ± 2.7	46.8 ± 1.3	30.1 ± 2.1	76.7 ± 1.9	68.0 ± 1.2	61.6 ± 1.8
Linear classifier	89.6 ± 0.2	79.91 ± 0.3	79.15 ± 0.35	79.69 ± 0.15	64.38 ± 0.15	61.54 ± 0.26	87.22 ± 0.09	77.1 ± 0.13	74.88 ± 0.15
SimCLR + K-means	79.72 ± 0.22	69.56 ± 0.28	62.06 ± 0.43	42.58 ± 0.74	43.7 ± 0.59	24.43 ± 0.81	69.95 ± 0.78	67.05 ± 0.33	55.37 ± 0.78
MIX'EM + K-means	81.87 ± 0.23	70.85 ± 0.26	66.59 ± 0.37	43.77 ± 0.51	46.41 ± 0.11	27.12 ± 0.33	77.76 ± 0.08	68.03 ± 0.07	61.35 ± 0.1
MIX'EM dominant component	82.19 ± 0.21	71.35 ± 0.27	67.15 ± 0.32	39.19 ± 0.44	43.59 ± 0.14	26.67 ± 0.12	68.44 ± 0.44	64.08 ± 0.1	54.66 ± 0.09

tions in a downstream task. The standard procedure to gauge a self-supervised representation encoder's performance is to estimate the accuracy of a linear classifier trained on the frozen features generated by the encoder [37,10,11]. Similarly, we train a linear classifier on the frozen base SimCLR encoder, SimCLR, and various forms of MIX'EM. According to table 3, the mixture module neither improves nor hurts the representation quality for the downstream supervised task. This means that the representation learned using SimCLR contains rich information to easily train a supervised linear classifier without further disentanglement of the representation. However, for the unsupervised setting, category-level disentanglement of the representation is essential, as we observed a significant boost in clustering accuracy using MIX'EM.

Table 3: MIX'EM does not disrupt contrastive learning objective while imposing category-level disentanglement on representations. (evaluated on STL10)

	Model	**Supervised linear classifier accuracy**
(0)	base SimCLR encoder	86.1
(1)	SimCLR	87.21 ± 0.1
	MIX'EM	
(2)	+ entropy maximization	87.25 ± 0.05
(3)	+ component entropy minimization	87.15 ± 0.06
(4)	+ associative embedding loss	87.22 ± 0.09

5.4 Effect of using test data in training

We investigate three scenarios regarding data splits used for training of MIX'EM and SimCLR and K-means; (1) using both train and test splits for training. This is the standard setting and legitimate, as we do not use the labels [25,39]. (2) only using the train split for training; (3) using the train and unlabeled splits (on STL10 only) for training. Note that we always evaluate on the test split. The results are presented in table 4.

Scenario (1) vs (2) Using test split in training consistently improves performance, having a more significant impact on STL10. We argue that this is due

Table 4: How splits used for training influence MIX'EM or SimCLR when K-means is used for clustering. All evaluations are on test split.

Dataset	Training splits	Method	ACC	NMI	ARI
STL10	train+unlabled	SimCLR	67.44 ± 0.71	64.90 ± 0.1	51.26 ± 0.18
		MIX'EM	71.04 ± 1.13	62.56 ± 0.85	52.29 ± 1.41
	train	SimCLR	65.57 ± 0.4	63.72 ± 0.2	50.50 ± 0.38
		MIX'EM	74.20 ± 0.06	65.19 ± 0.06	55.89 ± 0.1
	train+test	SimCLR	69.95 ± 0.78	67.05 ± 0.33	55.37 ± 0.78
		MIX'EM	77.76 ± 0.08	68.03 ± 0.07	61.35 ± 0.1
CIFAR10	train	SimCLR	77.74 ± 0.08	67.21 ± 0.15	58.54 ± 0.16
		MIX'EM	79.51 ± 0.41	68.29 ± 0.28	63.29 ± 0.44
	train+test	SimCLR	79.72 ± 0.22	69.56 ± 0.28	62.06 ± 0.43
		MIX'EM	81.87 ± 0.23	70.85 ± 0.26	66.59 ± 0.37

to the size and visual difficulty of STL10. CIFAR10 has 50k training and 10k test images. But, on STL10 there is only 5k training and 8k test images. Hence, on STL10, using test split in training means 160% additional data, while on CIFAR10 it is just a 20% addition. In the future, a more controlled experiment by progressively removing fractions of training data should help make a more informed conclusion. Additionally, STL10 is a subset of ImageNet and is visually more complex. On CIFAR100-20 trend is quite similar to CIFAR10.

Scenario (2) vs. (3) Unlabeled split of STL10 contains 100k images; however, we do not know the distribution of the categories, and it contains unknown distractor categories. Therefore, despite increasing training data by a large factor, performance drops in this scenario. MIX'EM presumes access to the expected number of categories, which does not hold for the unlabeled set. We believe this is the reason why the accuracy of K-means on SimCLR does not drop as much in this case. Nevertheless, MIX'EM still is significantly more accurate.

6 Conclusion

We presented MIX'EM, a novel solution for unsupervised image classification. MIX'EM incorporates a mixture of embeddings module in SimCLR in order to impose semantic structure on the representations. To successfully train MIX'EM, we introduce various loss terms. Our model sets a new, significantly high, state-of-the-art for research on unsupervised image classification. We also show that applying K-means itself on normalized representations from SimCLR results in impressively high accuracy. We believe this can be used as a new, and more challenging, procedure for evaluating the quality of self-supervised representation learning methods.

The results we publish here could be further improved by using the latest findings on the best settings for contrastive visual representation learning [51]. In the future, we would like to explore the impact of our model on image retrieval and instance segmentation tasks. Moreover, studying the theoretical aspects of MIX'EM could provide insight for further improvements.

References

1. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 770–778 1, 9
2. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. (2012) 1097–1105 1
3. Sun, C., Shrivastava, A., Singh, S., Gupta, A.: Revisiting unreasonable effectiveness of data in deep learning era. In: Proceedings of the IEEE international conference on computer vision. (2017) 843–852 1
4. He, K., Girshick, R., Dollár, P.: Rethinking imagenet pre-training. In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 4918–4927 1
5. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision, Springer (2014) 740–755 2
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition, Ieee (2009) 248–255 2
7. Bachman, P., Hjelm, R.D., Buchwalter, W.: Learning representations by maximizing mutual information across views. In: Advances in Neural Information Processing Systems. (2019) 15535–15545 2, 4
8. Gidaris, S., Singh, P., Komodakis, N.: Unsupervised representation learning by predicting image rotations. arXiv preprint arXiv:1803.07728 (2018) 2, 4
9. Wu, Z., Xiong, Y., Yu, S.X., Lin, D.: Unsupervised feature learning via non-parametric instance discrimination. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 3733–3742 2, 5, 6
10. Oord, A.v.d., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748 (2018) 2, 13
11. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. arXiv preprint arXiv:2002.05709 (2020) 2, 3, 4, 5, 6, 10, 13
12. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2020) 9729–9738 2, 5
13. Gutmann, M., Hyvärinen, A.: Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. (2010) 297–304 2, 4, 6
14. Asano, Y.M., Rupprecht, C., Vedaldi, A.: A critical analysis of self-supervision, or what we can learn from a single image. arXiv preprint arXiv:1904.13132 (2019) 2
15. Greff, K., Kaufman, R.L., Kabra, R., Watters, N., Burgess, C., Zoran, D., Matthey, L., Botvinick, M., Lerchner, A.: Multi-object representation learning with iterative variational inference. arXiv preprint arXiv:1903.00450 (2019) 3
16. Chen, M., Artières, T., Denoyer, L.: Unsupervised object segmentation by redrawing. In: Advances in Neural Information Processing Systems. (2019) 12726–12737 3
17. Li, C., Lee, G.H.: Generating multiple hypotheses for 3d human pose estimation with mixture density network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 9887–9895 3

18. Lee, S., Purushwalkam, S., Cogswell, M., Crandall, D., Batra, D.: Why m heads are better than one: Training a diverse ensemble of deep networks. arXiv preprint arXiv:1511.06314 (2015) 3, 7
19. Ye, Q., Kim, T.K.: Occlusion-aware hand pose estimation using hierarchical mixture density network. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 801–817 3
20. Makansi, O., Ilg, E., Cicek, O., Brox, T.: Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 7144–7153 3
21. Varamesh, A., Tuytelaars, T.: Mixture dense regression for object detection and human pose estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2020) 13086–13095 3
22. McLachlan, G.J., Basford, K.E.: Mixture models: Inference and applications to clustering. Volume 84. M. Dekker New York (1988) 3, 7
23. Jordan, M.I., Jacobs, R.A.: Hierarchical mixtures of experts and the em algorithm. Neural computation **6** (1994) 181–214 3
24. Bishop, C.M.: Mixture density networks. (1994) 3
25. Ji, X., Henriques, J.F., Vedaldi, A.: Invariant information clustering for unsupervised image classification and segmentation. In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 9865–9874 3, 4, 5, 10, 13
26. Huang, J., Gong, S., Zhu, X.: Deep semantic clustering by partition confidence maximisation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2020) 8849–8858 3, 4, 5, 13
27. Van Gansbeke, W., Vandenhende, S., Georgoulis, S., Proesmans, M., Van Gool, L.: Learning to classify images without labels. arXiv preprint arXiv:2005.12320 (2020) 3, 4, 5, 12, 13
28. Newell, A., Deng, J.: Pixels to graphs by associative embedding. In: Advances in neural information processing systems. (2017) 2171–2180 3, 8
29. Newell, A., Huang, Z., Deng, J.: Associative embedding: End-to-end learning for joint detection and grouping. In: Advances in Neural Information Processing Systems. (2017) 2277–2287 3, 8
30. Chen, X., Fan, H., Girshick, R., He, K.: Improved baselines with momentum contrastive learning. arXiv preprint arXiv:2003.04297 (2020) 4, 6
31. Hjelm, R.D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., Bengio, Y.: Learning deep representations by mutual information estimation and maximization. arXiv preprint arXiv:1808.06670 (2018) 4
32. Donahue, J., Krähenbühl, P., Darrell, T.: Adversarial feature learning. arXiv preprint arXiv:1605.09782 (2016) 4
33. Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M., Courville, A.: Adversarially learned inference. arXiv preprint arXiv:1606.00704 (2016) 4
34. Doersch, C., Gupta, A., Efros, A.A.: Unsupervised visual representation learning by context prediction. In: Proceedings of the IEEE international conference on computer vision. (2015) 1422–1430 4
35. Caron, M., Bojanowski, P., Joulin, A., Douze, M.: Deep clustering for unsupervised learning of visual features. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 132–149 4, 5, 7, 13
36. Noroozi, M., Favaro, P.: Unsupervised learning of visual representations by solving jigsaw puzzles. In: European Conference on Computer Vision, Springer (2016) 69–84 4

37. Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. In: European conference on computer vision, Springer (2016) 649–666 4, 13
38. Asano, Y.M., Rupprecht, C., Vedaldi, A.: Self-labelling via simultaneous clustering and representation learning. arXiv preprint arXiv:1911.05371 (2019) 5, 7
39. Yang, J., Parikh, D., Batra, D.: Joint unsupervised learning of deep representations and image clusters. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 5147–5156 5, 10, 13
40. Yan, X., Misra, I., Gupta, A., Ghadiyaram, D., Mahajan, D.: Clusterfit: Improving generalization of visual representations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2020) 6509–6518 5
41. Zhan, X., Xie, J., Liu, Z., Ong, Y.S., Loy, C.C.: Online deep clustering for unsupervised representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2020) 6688–6697 5
42. Sanchez, E.H., Serrurier, M., Ortner, M.: Learning disentangled representations via mutual information estimation. arXiv preprint arXiv:1912.03915 (2019) 5
43. Von Luxburg, U.: A tutorial on spectral clustering. Statistics and computing **17** (2007) 395–416 9
44. Coates, A., Ng, A., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics. (2011) 215–223 9
45. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. (2009) 9
46. Han, K., Rebuffi, S.A., Ehrhardt, S., Vedaldi, A., Zisserman, A.: Automatically discovering and learning new visual categories with ranking statistics. arXiv preprint arXiv:2002.05714 (2020) 9
47. Maaten, L.v.d., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9** (2008) 2579–2605 10
48. Xie, J., Girshick, R., Farhadi, A.: Unsupervised deep embedding for clustering analysis. In: International conference on machine learning. (2016) 478–487 13
49. Chang, J., Wang, L., Meng, G., Xiang, S., Pan, C.: Deep adaptive image clustering. In: Proceedings of the IEEE international conference on computer vision. (2017) 5879–5887 13
50. Haeusser, P., Plapp, J., Golkov, V., Aljalbout, E., Cremers, D.: Associative deep clustering: Training a classification network with no labels. In: German Conference on Pattern Recognition, Springer (2018) 18–32 13
51. Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., Isola, P.: What makes for good views for contrastive learning. arXiv preprint arXiv:2005.10243 (2020) 14