

Improving Contrastive Learning by Visualizing Feature Transformation

Rui Zhu^{1,2*}, Bingchen Zhao^{3*}, Jingen Liu^{2†}, Zhenglong Sun¹, Chang Wen Chen⁴

¹ The Chinese University of HongKong, Shenzhen ² JD AI Research ³ Tongji University ⁴ The Hong Kong Polytechnic University

ruizhu@link.cuhk.edu.cn, {zhaobc.gm, jingenliu}@gmail.com, sunzhenglong@cuhk.edu.cn, changwen.chen@polyu.edu.hk

Abstract

Contrastive learning, which aims at minimizing the distance between positive pairs while maximizing that of negative ones, has been widely and successfully applied in unsupervised feature learning, where the design of positive and negative (pos/neg) pairs is one of its keys. In this paper, we attempt to devise a feature-level data manipulation, differing from data augmentation, to enhance the generic contrastive self-supervised learning. To this end, we first design a visualization scheme for pos/neg score¹ distribution, which enables us to analyze, interpret and understand the learning process. To our knowledge, this is the first attempt of its kind. More importantly, leveraging this tool, we gain some significant observations, which inspire our novel Feature Transformation proposals including the extrapolation of positives. This operation creates harder positives to boost the learning because hard positives enable the model to be more view-invariant. Besides, we propose the interpolation among negatives, which provides diversified negatives and makes the model more discriminative. It is the first attempt to deal with both challenges simultaneously. Experiment results show that our proposed Feature Transformation can improve at least 6.0% accuracy on ImageNet-100 over MoCo baseline, and about 2.0% accuracy on ImageNet-1K over the MoCoV2 baseline. Transferring to the downstream tasks successfully demonstrate our model is less task-bias. Visualization tools and codes: <https://github.com/DTennant/CL-Visualizing-Feature-Transformation>.

1. Introduction

Finetuning from ImageNet [36] supervised pre-train networks [16, 39, 19] for down-stream tasks, such as object detection [28, 33, 34] and semantic segmentation [29, 5], is a de facto dominant approach in computer vision community. But recently self-supervised contrastive learn-

*Equally-contributed and this work is done at JD AI Research.

†Corresponding author.

¹Pos/neg score indicates cosine similarity of pos/neg pair.

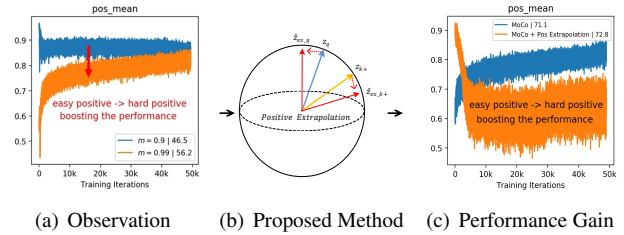


Figure 1. The motivation of visualizing the score distribution. (a) It draws the score distribution of positive pairs for m (the momentum in MoCo[14]) being 0.99 and 0.9, showing that smaller positive scores generally need longer time to converge and obtain better accuracy. (b) Inspired by (a), we apply extrapolation on positive pairs to slightly decrease the scores, generating harder positives. (c) Leveraging the extrapolation of positives, we improve the performance from 71.1% (the blue) to 72.8% (the orange). The performance increase is consistent with the change of distribution. The mean score of positive pairs changes from blue plot (before extrapolation) to orange plot (after extrapolation).

ing achieves comparable transfer performance without the human-provided annotations. One of the key issues of contrastive learning is to design positive and negative (pos/neg) pairs to learn an embedding space such that the positives stay closer in the space while the negatives are pushed away.

Most existing approaches [4, 6, 42, 7] acquire pos/neg pairs by data augmentation, which exploits various views of the same image to form positive pairs. For example, CMC[41] uses the luminance and chrominance color channel of an image as two views. InfoMin [42] demonstrates that incremental data augmentations indeed lead to decreasing mutual information between views and thus improve transfer performance. In other words, an effective positive pair prefers to convey more variance of one instance. With a series of promotions, the contrastive learning methods based on data augmentations [4, 6, 42, 7] are achieving closer to the fully supervised performance on ImageNet[6].

Most previous data augmentations (*e.g.*, cropping, color distortion) are directly sourced from human intuitions, which may lack much interpretability, thus they can not guarantee their effectiveness. We argue, however, that the feature-level data manipulation (*i.e.*, feature transformation) can provide more explainable or effective pos/neg pairs to

enhance the feature embedding. To this end, we first design a scheme to visualize the pos/neg pair score distributions during the training. We believe that, from these score distributions, we can reveal and explain how the model parameter values affect its performance. The visualization can help us trace back the training process. Moreover, it enables us to observe the characteristics of the pos/neg pairs, and then invent more effective feature transformations (FT).

Figure 1 demonstrates the motivation of score visualization. By plotting the score distributions under different momentum values of MoCo [14], we can clearly observe that the case of $m = 0.99$ has smaller positive scores while achieves better performance. A small positive score indicates less similarity between the pair, which means this positive pair actually carrying large view variance of one example. Actually, this is consistent with the goal of feature learning, which targets at a more view-invariant visual representation. Therefore, we conjecture that “hard positives” are the ones conveying large view variance of a sample. Inspired by this observation, we introduce an extrapolation operation on positive pairs to increase view variance and thus acquire hard positives. Figure 1(c) shows that the extrapolation of positives can boost the model performance from the “blue” one to the “orange” one.

Besides, to make full use of negative features, we propose the random interpolation among negatives, which intuitively provides diversified negatives for each training step and makes the model more discriminative.

Unlike the traditional data augmentation, our feature transformation does not bring additional training examples. Instead, it aims at reshaping the feature distribution by manipulating both positive and negative pairs. Basically, our feature transformation will create hard positives and diversified negatives to learn a more view-invariant (hard positive) and a more discriminative (diversified negatives) representation. It is directly driven by the performance of the learned representation, while data augmentation is kind of blind to the performance. Furthermore, our feature transformation makes the model less “task-bias”, which means we can achieve performance improvement for various downstream tasks. It has been verified by our experiments on object detection, instance segmentation, and long-tailed classification with significant improvement.

Both our visualization tool and feature transformation are generic, and can be applied to various self-supervised contrastive learning including MoCo[14], SimCLR[6], InfoMin[42], SwAv[4], SimSiam[8]. In the following sections, we employ the classic model MoCo to demonstrate our framework. To summarize, our contributions include:

- We are the first to design a visualization tool to analyze and interpret how the score distribution of pos/neg pairs affects the model’s capability. The visualization also helps us come into some significant observations.

- Inspired by the observations on the model visualization, we propose a simple yet effective feature transformation, which creates both “hard positives” and “diversified negatives” to enhance the training. The feature transformations enable to learn more “view-invariant” and discriminative representations.
- We conduct thorough experiments and our model achieves the state-of-the-art performance. In addition, the experiments on the downstream tasks successfully demonstrate our model is less task biased.

2. Related Work

Contrastive Learning: Contrastive losses have been widely used in self-supervised learning and brought significant improvements on classification [13, 1, 14, 41, 42, 6, 7, 12, 4, 18, 2, 59, 49, 52, 9, 3, 41, 45, 51, 56, 47, 46, 24] and detection [48, 53, 54, 55]. InfoMin [42] uses the lower bound of NCE to demonstrate that incremental data augmentations lead to decreasing mutual information between views and thus improve transfer performance. In other words, relatively harder data augmentation for contrastive learning boosts the transfer performance[20, 6]. We show that our proposed feature transformation can be easily adopted on current state-of-the-art models.

MixUp for contrastive learning Mixup [58] and its numerous variants [44, 57, 22] provide highly effective data augmentation strategies when paired with a cross-entropy loss for supervised and semi-supervised learning. Manifold mixup [44] is a feature-level regularization for supervised learning while Un-mix [38] proposes using mixup in the image/pixel space for self-supervised learning; And in MoChi [20] the authors propose mixing the negative sample in the embedding space for hard negatives augmentation but hurt the classification accuracy. i-Mix [25] proposed a strategy mixing instances in both input and virtual label spaces to regularize contrastive training. In this paper, we proposed to use feature transformation rather than data augmentation. Positive features are extrapolated to increase the hardness of positives, and negative features in the memory queue are interpolated to increase the diversity. Our FT provides more efficacy compared with augmentations.

Generating examples for metric learning: The idea of generating new examples for metric learning has been explored by [27, 10, 23]. The Embedding Expansion [23] work uses uniform interpolation between two positive and negative points, creates a set of synthetic points, and then selects the hardest pair as negative. [27, 10] generate new hard examples by generators and improve performance for metric learning. Different from the approaches [27, 10] for supervised metric learning, our pos/neg FTs are aiming at self-supervised learning and doesn’t require labels, extra parameters and loss terms to be optimized.

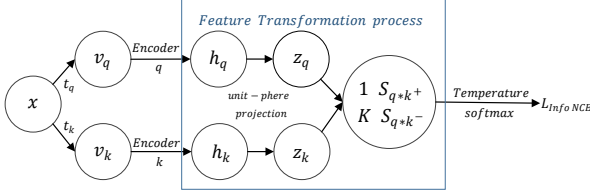


Figure 2. Feature Transformation Contrastive learning pipeline.

3. Visualization of Contrastive Learning

3.1. Preliminaries

Let us start from the basic procedures of contrastive learning, as shown in Figure 2. Each data sample x passes through two separate data augmentation pipeline t_q and t_k , which are randomly sampled from the same data augmentation pool, and two views v_q and v_k will be acquired to construct positive pairs [6, 12]. The encoder q and k ² will respectively map two views into feature embedding space. An ℓ_2 normalization is applied on feature vector h_q and h_k to project the corresponding vector h_q and h_k (i.e., $z_q = h_q / \|h_q\|_2$) onto the unit sphere and obtain z_q and z_k . Their inner product will produce the cos similarity score, namely one positive pair score $S_{q,k+}$ and K negative pair scores $S_{q,k-}$. These pair scores are input to InfoNCE loss 1 for contrastive learning:

$$\mathcal{L} = -\log \left[\frac{\exp(S_{q,k+}/\tau)}{\exp(S_{q,k+}/\tau) + \sum_K \exp(S_{q,k-}/\tau)} \right] \quad (1)$$

Here we roughly defined Feature Transformation process as certain manipulations on encoder embeddings h_q and h_k , in order to reshape the distribution of the output pos/neg pair score ($S_{q,k+}$ and $S_{q,k-}$), for better contrastive learning in the follow-up InfoNCE loss. The most common FT applied in current SOTA is the [4, 6, 42, 7, 14] unit-sphere projection of ℓ_2 normalization. We provide empirical studies of this regular FT and illustrate its importance for significant constriction of feature length (ℓ_2) in Supp F.

3.2. Score Distribution Visualization

We choose to visualize the score distribution of pos/neg pairs instead of the loss curves and transfer accuracy, as the inside training dynamics can unearth the learning capability of the model. Specifically, there are two practical reasons: (1) The basic idea of InfoNCE loss is to compare the pos/neg scores in a log-softmax manner, so visualizing the input score pairs can help study the contrastive learning process. (2) The normalized feature vectors z_q and z_k are high-dimensional, which is challenging for storage and visualization; The exponential amplification of scores is too large to observe the details of characteristics of pos/neg scores.

²Encoder q and k might be the same [6] or different network [14, 12].

m	≤ 0.5	0.6	0.7	0.8	0.9	0.99	0.999	1
acc (%)	<i>collapse</i>	21.2	32.8	39.3	46.5	56.2	53.1	31.2

Table 1. The parameter experiments of m on MoCo ($\tau = 0.07$).

However, $S_{q,k}$ is one-dimensional and limited to $[-1, 1]$, which is suitable to observe inside the contrastive process.

Notice that this practical visualization tool is offline and doesn't affect training speed with negligible computation. Even with larger datasets and batch size, it's still feasible. The details of the visualization tool are present in Supp A.

3.3. Visualization Examples with MoCo

We choose the computationally-efficient model, MoCo [14] as an example to demonstrate our visualization design.

Momentum Update Mechanism: Memory queue [51] is an initial approach for solving the large batch computational burden which stores K negative features in the memory that will be updated using the output of the encoder at each training step. However, the rapid change of the encoder (f_q and f_k) could bring inconsistency into the memory queue which usually contains outdated features. MoCo solves the inconsistency issue by leveraging a momentum update mechanism [40] where only f_q is updated by back-propagation and the f_k is updated by momentum mechanism:

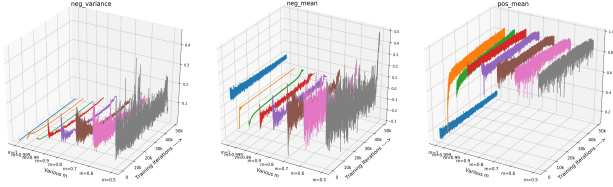
$$\theta_{f_k} \leftarrow m\theta_{f_k} + (1 - m)\theta_{f_q} \quad (2)$$

where $m \in [0, 1)$ is the momentum coefficient and has a huge influence to the final transfer accuracy. The memory queue is then updated using the features from f_k because the momentum update of f_k brings a smoother change of features that could reduce the inconsistency in memory queue.

In the following sections, we provide thorough experiments and visualization analysis to show how the parameter m affects the contrastive learning process. We attempt various m for MoCo on ImageNet-100 (denoted as IN-100) [41] with linear readout protocol for evaluation (details in Supp B). As the Tab 1 shown, with the decrease of m (increasing the update speed of encoder f_k), the accuracy presents an inverse U-shape and the max 56.24% locates at $m = 0.99$ and the model collapse³ when $m \leq 0.5$. The trend of these results is similar with BYOL [12].

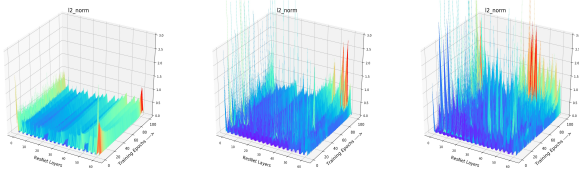
We choose three non-trivial statistics to visualize the score distribution: the mean of pos/neg scores (indicating the approximate average of the pos/neg pair distance) and the variance of negative scores (indicating the fluctuation degree of the negative samples in the memory queue). As shown in Fig 3(a), when m becomes smaller, the update speed of encoder k is increasing, leading to incremental differences of features among training steps, which is reflected as the growing variance of negative scores of the queue,

³Model collapse means that the transfer accuracy with linear readout protocol can not achieve the accuracy of training from random initialization, i.e., 15.90%, indicating the negative effect brought by pre-train.



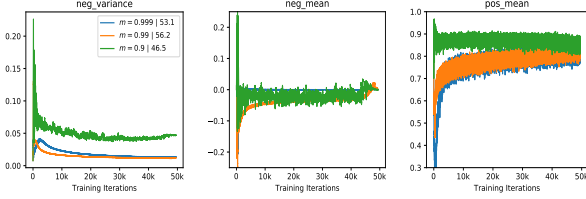
(a) Var of neg scores (b) Mean of neg scores (c) Mean of pos scores

Figure 3. Pos/neg score statistics of various m in MoCo training



(a) $m = 0.99$ | 56.2% (b) $m = 0.6$ | 21.2% (c) $m = 0.5$ | collapse

Figure 4. Gradient (ℓ_2 norm) landscape of various m



(a) Var of neg scores (b) Mean of neg scores (c) Mean of pos scores

Figure 5. 2D view of pos/neg score statistics of various m

namely the inconsistency. Specifically, when $m = 1$ (no update of f_k during training), the variance is closed to zero (blue line) while the variance of $m = 0.9$ (red) is larger but relatively unstable. $m = 0.5$ (grey) brings more violent fluctuations/inconsistency in the memory queue, leading to a poor transfer accuracy even model collapse.

Inside Analysis of Model Collapse: The model collapse is caused by various reasons. Small m (fast update speed of f_k) brings not only the inconsistency, but also the confusion of negative scores. For the mean of neg scores (lines in Fig 3(b)), the volatility degree of $m = 0.6$ (pink) and $m = 0.5$ (grey) is much sharper than the best model $m = 0.99$ (green). The mean of neg scores reflects the approximate score for all the negative pairs in the memory queue. If it becomes drastically volatile with the training process, the corresponding loss value and gradient will fluctuate violently, resulting in bad convergence. As shown in Fig 4, the smooth and stable gradient landscape of $m = 0.99$ (Fig 4(a)) becomes sharp and messy with the decrease of m (Fig 4(b) for $m = 0.6$ and Fig 4(c) for $m = 0.5$). Details of gradient landscape are put in Supp C. *Basically, to learn a better pre-trained model, we need to prepare negative pairs that can maintain the stability and smoothness of score distribution and gradient for the training process, which is similar to supervised learning [37].*

α_{ex}	-	0.2	0.4	0.6	1.4	1.6	2.0
acc (%)	71.1	71.6	71.8	71.9	72.7	72.4	72.8

Table 2. Various α_{ex} for positive extrapolation, the best result is marked in bold. We employ ResNet-50 [16] for the results. '-' indicates MoCo baseline without using extrapolation.

Hard Positive Boosts Performance: Small m not only indicates the faster update speed, but also more similarity between encoder f_k and f_q , i.e., in an extreme case, when $m = 0$, the parameters θ_k is completely the same with θ_q in each training step. The increasing similarity of encoder f_q and f_k will reduce the dissimilarity between z_q and z_{k+} , and only the view variance brought by data augmentations remains, leading to a higher positive score. Fig 3(c) shows that high positive scores of $m \leq 0.9$ will produce easy positive pairs with the close distance and little view variance in feature space.

However, in Fig 5(c), when we increase m from 0.9 (green) to 0.99 (orange), the easy pos pair becomes hard pos pair (from very similar 0.9 to less similar 0.7), leading to a higher transfer accuracy (46.5% v.s 56.2%, 9.7% increased). Note that this observation (converting easy positive to hard one) could be explained by InfoMin principle [42]: Raising the view variance between z_q and z_{k+} corresponds to increasing the mutual information for contrastive learning, which forces the encoder learns a more robust embedding and thus improves the transfer accuracy.

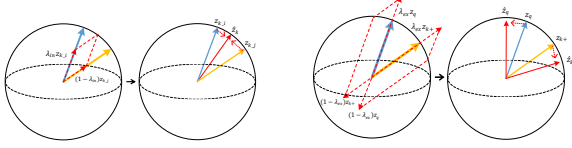
In the guarantee of stable and smooth score distribution and gradient, we can adopt some feature transformation methods which create hard ones by decreasing easy positive scores. Thus, we propose a positive feature extrapolation method to improve transfer accuracy in section 4.1.

4. Proposed Feature Transformation Method

The learning objective of Info-NCE is to draw the positive pair (z_q and z_{k+}) closer while pushing away negative pairs (z_q and all the z_{k-} in memory queue) in the embedding space. Therefore, we could directly apply feature transformation on the pos/neg features, in order to provide appropriate regularization [44] or make the learning harder [42]. Specifically, we develop positive extrapolation to transform the original positive pair to be further to increase the hardness and negative interpolation of memory queue to increase the diversity of negative samples, as Fig 6 shown. Notably, our method does not change the loss terms because it only replaces original pair scores with the new transformed pos/neg scores for calculating loss term.

4.1. Positive Extrapolation

Following the discussion in Sec 3.3 which indicates that lowering the easy positive pair scores to create hard positive pairs during training could be beneficial for the final transfer performance. Thus we would like to explore a way to



(a) Negative Interpolation (b) Positive Extrapolation

Figure 6. The process of our proposed negative interpolation and positive extrapolation. For the negative interpolation, we randomly interpolate two features in memory queue to produce a new negative. For positive extrapolation, the two positive features are pushed away from each other using extrapolation, changing easy positives to hard positives, which is better for contrastive learning.

manipulate the positive features z_q and z_{k+} to increase the view variance between them during training.

First, we simply adopt weighted addition for the two positive features to generate new feature:

$$\begin{aligned}\hat{z}_q &= \lambda_{ex} z_q + (1 - \lambda_{ex}) z_{k+} \\ \hat{z}_{k+} &= \lambda_{ex} z_{k+} + (1 - \lambda_{ex}) z_q\end{aligned}\quad (3)$$

where \hat{z}_q and \hat{z}_{k+} are the transformed new features. Meanwhile, considering the design principle of mixup [44, 58], we make sure that the summation of weights equals to 1. More importantly, we should guarantee that the transformed pos score $\hat{S}_{q,k+}$ is smaller than the original pos score $S_{q,k+}$, namely $\hat{z}_q \hat{z}_{k+} \leq z_q z_{k+}$. Take Equation 3 into the transformed score:

$$\hat{S}_{q,k+} = 2\lambda_{ex}(1 - \lambda_{ex})(1 - S_{q,k+}) + S_{q,k+} \leq S_{q,k+} \quad (4)$$

Because $S_{q,k+} \in [-1, 1]$ and thus $(1 - S_{q,k+}) \geq 0$. To make sure the lower score $\hat{S}_{q,k+} \leq S_{q,k+}$, we need to set $\lambda_{ex} \geq 1$ to let $2 \cdot \lambda_{ex}(1 - \lambda_{ex}) \leq 0$. So we choose $\lambda_{ex} \sim \text{Beta}(\alpha_{ex}, \alpha_{ex}) + 1$ ⁴ is sampled from a beta distribution and then adding 1 results in a range of (1, 2). And the range of transformed pos score will be $\hat{S}_{q,k+} \in [-4 + 5S_{q,k+}, S_{q,k+}]$.

Intuitively, it can be seemed as a simple approach to push away z_q and z_{k+} in feature space. After extrapolation, the distance between the extrapolated feature vector is enlarged. Therefore the extrapolation can serve as a feature transformation to create hard positives from easy ones. As shown in Fig 6(b), it brings a minor direction change for two positive vectors and meanwhile conveying a larger view variance of a sample for better contrastive learning. The visualization of lowering pos score by extrapolation is shown in Fig 1(c).

We evaluate the efficacy of positive extrapolation on IN-100 and attempt various α_{ex} in Tab 2. The positive extrapolation with various α_{ex} consistently improves the accuracy from the baseline MoCo (71.1%), which clearly

⁴We choose to set the two parameter α_{ex} of the beta distribution to be the same, because the two mixed features are symmetrical. And the same applies to the negative feature interpolation.

Method	α_{ex}	pos interpolation/extrapolation
MoCo	0.2	69.1 / 71.6
(baseline: 71.1)	2.0	67.4 / 72.8

Table 3. Positive extrapolation v.s. interpolation. Interpolation hurts the performance while extrapolation improves.

demonstrates the efficacy of positive extrapolation. It is interesting that $\alpha_{ex} > 1$ will get better results than those of $\alpha_{ex} < 1$. Because the beta distribution with $\alpha_{ex} < 1$ provides extreme large or small λ_{ex} with high probability, e.g., 1.1 or 1.9, while the beta distribution with $\alpha_{ex} > 1$ gives neutral $\lambda_{ex} = 1.5$ with high probability⁵. According to Equation 4, extreme λ_{ex} will bring too much/little hardness, so the corresponding performance is not robust as the neutral one.

What if Positive Interpolation? To further verify our conjecture that extrapolation can create hard positives while interpolation won't, we also conduct experiments for the interpolation of positive features, shown in Tab 3. We can observe a clear performance drop (5.4% drop for neutral $\alpha_{ex} = 2$) for this experiment. The reason is that the interpolation between positive features pulls the positive pairs together thus reducing the hardness in the training process. In other words, the view variance of positive pairs is decreasing, and thus easy to cause non-robust features.

4.2. Negative Interpolation

Previous contrastive models [6, 14] do not make full use of negative samples. e.g., In MoCo, there are many repetitive negative features stores in the memory queue iteration by iteration. Thus we could design a new strategy to fully utilize negative features and increase the diversity of the memory queue. With sufficient randomness, we propose the negative interpolation in memory queue, which intuitively provides diversified negatives for each training step.

Specifically, we denote the negative memory queue of MoCo as $Z_{neg} = \{z_1, z_2, \dots, z_K\}$ where K is the size of the memory queue, and Z_{perm} as the random permutation of Z_{neg} . We propose to use a simple interpolation between two memory queue to create a new queue $\hat{Z}_{neg} = \{\hat{z}_1, \hat{z}_2, \dots, \hat{z}_K\}$:

$$\hat{Z}_{neg} = \lambda_{in} \cdot Z_{neg} + (1 - \lambda_{in}) \cdot Z_{perm} \quad (5)$$

where $\lambda_{in} \sim \text{Beta}(\alpha_{in}, \alpha_{in})$ is in the range of (0, 1), as Fig 6(a) shown. The transformed memory queue \hat{Z}_{neg} provides fresh interpolated negatives for contrastive loss iteration by iteration, where the random permutation and λ_{in} ensure the diversity of \hat{Z}_{neg} of each training step. The diversity makes the model to compare with much more linear combinations

⁵The beta distribution with $\alpha_{ex} > 1$ shows an inverted U shape which samples 0.5 with a greater probability and thus making λ_{ex} to have a greater chance to be 1.5.

α_{in}	-	0.2	0.4	0.6	1.4	1.6	2.0
acc (%)	71.1	73.3	74.1	74.2	73.5	74.6	74.1

Table 4. Various α_{in} for negative interpolation, the best result is marked in bold. We employ ResNet-50 [16] for the results. ‘-’ indicates MoCo baseline without using negative interpolation.

Method	α_{in}	Z_n	queue size	Acc
moco+ original queue	-	Z_{neg}	K	71.10
moco+ original queue	-	Z_{neg}	$2K$	71.40
moco+ Neg FT queue	1.6	\hat{Z}_{neg}	K	74.64
moco+ Neg FT+original	1.6	\tilde{Z}_{neg}	$2K$	74.73

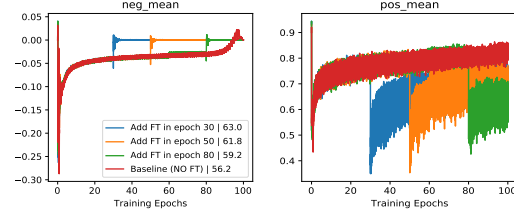
Table 5. Ablation results for using different queue of negative features (Res50). The transformed queue \hat{Z}_{neg} can completely replace the extended queue \tilde{Z}_{neg} with small computations.

of previous negatives in each training step. Positive extrapolation increases the view variance between two pos features while the negative interpolation similarly boosts the “sample variance” (diversity) of the memory queue. We conjecture that original queue Z_{neg} provides discrete distribution of negative samples but our method can fill in the incomplete sample points of the distribution by random interpolation, leading to a more discriminative model. We evaluate the efficacy of negative interpolation on IN-100 and attempt various α_{in} in Tab 4. The neg interpolation is fairly robust with various α_{in} , with the improvement of 2.2%-3.5% from the baseline (71.1%). More interesting discussions about negative feature transformation (hard negatives & negative extrapolation) are shown in Supp G.

Previous works have explored the method leveraging image-level [38] and feature-level [20] mixing in contrastive learning. Our method differs from the previous works in three ways, first is the motivation, we are motivated by our observation in Sec 3.2 to propose the feature transformation strategies. Second, the way we extrapolate between two positive features is novel and outperforms the other two methods on several experiments in Tab 8 and 9. Third, the negative interpolation aims at fully utilizing negative samples in each training step. Both FT methods focus on exploring an effective way to perform feature transformation, not simply extending hard negatives to memory queue [20], neither the image-level mixup [38]. In the following sections, we provide inside discussions for the proposed FT, including (1) What if extending memory queue instead of FT. (2) When to add FT? (3) Dimension-level mixing rather than linear mixup. (4) Could the gains brought by FT vanish if training longer?

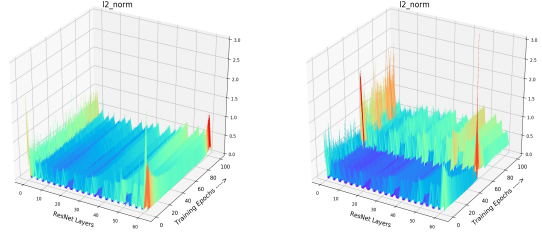
4.3. Discussions

Extending memory queue instead of FT: Previous works [14, 6] show that increasing the number of negative example (K) in contrastive learning could be beneficial for the final performance, thus they either uses a memory



(a) Mean of neg scores

(b) Mean of pos scores



(c) Baseline MoCo landscape

(d) Adding FT in 50th epoch

Figure 7. Visualization of when to add FT, including score distribution and Gradient (ℓ_2 norm) landscape.

FT begin epoch	0	2	30	50	80	-
Res18 acc (%)	62.6	63.3	62.9	61.8	59.2	56.2
Res50 acc (%)	76.9	76.4	75.9	74.0	72.2	71.1

Table 6. When to add feature transformation. We employ Res-18 (total 100 epochs) and Res-50 (total 200 epochs) on IN-100 for the results. ‘-’ indicates MoCo baseline without using any FT.

queue [14] or a large batchsize [6] to obtain more negative examples. Specifically, [31, 17, 42] shows that increasing K will improve the lower bound of the mutual information. The negative interpolation method could also be leveraged to enlarge the number of negative examples: We use the union queue of original negatives and interpolated queue, $\tilde{Z}_{neg} = \hat{Z}_{neg} \cup Z_{neg}$, which contains twice the number of negative examples ($2K$) than \hat{Z}_{neg} .

We compare the performance of using only the interpolated queue \hat{Z}_{neg} , original Z_{neg} with $K/2K$ negative samples, and their combination \tilde{Z}_{neg} , in Tab 5. We found that using the combination queue shows negligible improvement over the performance (74.73%) of using the interpolated queue alone (74.64%). We consider that the interpolated negative features contain sufficient diversified negatives compared with the original queue. So even the double negative samples (more mutual information) of the extended queue (\tilde{Z}_{neg}) cannot boost the performance. Notably, the extended queue requires double times computation for each contrastive loss. Thus we recommend feature transformations with less computation but more efficacy rather than feature augmentation.

When to add feature transformation? Here we present the efficacy of FT by analysis of starting FT in various training stages. As shown in Tab 6, starting FT (pos extrapolation + neg interpolation) from various epoch can consis-

tently boost the accuracy of baseline, and starting from earlier can improve more (7.1%/5.8% boosts with Res-18/Res-50). With the visualizations of score distribution and gradient landscape in Fig 7, we can see that our FT brings hard positives (lowering pos scores in Fig 7(b)) and hard negatives (rising neg scores in Fig 7(a)) simultaneously when the combined FT is inserted in various stages. Besides, with the comparison of the gradient (ℓ_2 norm) landscape, we can observe that our FT brings a greater gradient for the training, which makes the model escape from the local minima and avoid over-fitting. These analyses indicate our FT is a plug-and-play method and brings persistent view-invariance and discrimination for the training of contrastive models. More detailed discussions and visualizations are put in Supp D.

How about Dimension-level mixing: Besides the proposed linear feature interpolation and extrapolation on the feature-level (128-d vector), we also extend the transformation to a dimension-level where the parameter λ is a vector rather than a scalar number, this dimension-level mixing can be described as follows:

$$\hat{z}_{new} = \lambda \odot z_i + (1 - \lambda) \odot z_j \quad (6)$$

where \odot stands for Hadamard product, and $\lambda \in [0, 1]^n$ is a vector with the same dimension as the feature vector. The value of each dimension of λ is randomly sampled from a beta distribution $\lambda_i \sim \text{Beta}(\alpha, \alpha)$. This formulation is used for negative interpolation; For positive, λ is added 1 to perform extrapolation. For neg/pos features, the dimension-level mixing could introduce more diversity/more view variance (hardness) because every dimension is performed with transformation. Experiments of dimension-level mixing on IN-100 shows improvement over the feature-level mixing (the 5th row in Tab 7).

Could the gains brought by FT vanish if training longer? Simply training longer leads to significant performance boost for contrastive pre-train. So here we provide the results of MoCov2/MoCov2+FT (500 epoch) on IN-100: 80.7%->81.5%. Compared with 200 epoch results (75.6%->78.3% in Tab 7), longer training actually minimizes the improvement over the baseline. More training epochs can lead to comparing much more pos/neg pairs to increase the diversity. However, our proposed FT accelerates this process by providing diversity and results in fast convergence, which responds to the motivation of learning diversified and discriminative representations.

5. Experiments

In this section, we evaluate our Feature Transformation methods from four perspectives: (1) Ablation studies (2) FT on various contrastive models. (3) Evaluating the representation on ImageNet-1k. (4) Finetuning on various downstream tasks. We keep the fairness of the experiments, especially when compared with other methods. Notice that the

Method	MoCov1	MoCov2	simCLR	Infomin	swav	SimSiam
baseline*	71.10	75.61	74.32	81.9	82.1	77.1
+pos FT	72.80	76.22	75.80	-	-	77.8
+neg FT	74.64	77.12	76.71	-	-	
+both	76.87	78.33	78.25	83.2	83.2	
+both _{dim}	77.21	79.21	78.81	-	-	

Table 7. Ablation studies of proposed methods on various contrastive models. The models are pre-trained for 200 epochs with Res50 on IN-100. * indicates reproduced baseline results.

pre-train	IN-1k inat-18 CUB200 FGVC-aircraft			
supervised	76.1	66.1	81.9*	82.6*
mocov1[14]	60.6	65.6	82.8*	83.5*
mocov1+ours	61.9	67.3	83.2	84.0
mocov2[7]	67.5	66.8*	82.9*	83.6*
mocov2+ours	69.6	67.7	83.1	84.1
mocov2+MoChi[20]	68.0	-	-	-
mocov2+UnMix[38]	68.6	-	-	-

Table 8. Classification results. * indicates our reproduced results.

data augmentations are followed with the baseline methods. Details of experiments and datasets are put in Supp B.

5.1. Ablation study

We adopt the linear readout protocol [14] to compare performance for image classification on IN-100, where we freeze the features and train a supervised linear classifier using softmax. Tab 7 summarizes the results of ablation studies. We observe that the positive extrapolation and negative interpolation components are complementary which can improve the top-1 accuracy by 5.77%/2.72% when combined on MoCoV1/MoCoV2. The dimension-level mix also shows improvement based on the already high performance of both components. The performance-boosting of ablation studies over MoCo shows the efficacy of our FT. Notice that the transformed features are not necessarily on the unit sphere (*i.e.*, has a norm of 1), we did not need to re-perform ℓ_2 norm for transformed features, because the performance difference is negligible (76.87% *v.s* post-norm 76.68%). More discussions about ℓ_2 for vector length are put in Supp F. Here we strongly recommend to re-perform ℓ_2 norm for the transformed features on all the datasets, for the sake of contrasting all the scores on the unit-sphere.

5.2. FT on various contrastive models

We apply our FT to various contrastive models in Tab 7. It presents that our FT brings 5.77%, 3.93%, 1.3%, 1.1%, and 0.7% improvement over MoCo [14], SimCLR [6], InfoMin [42], SWAV [4] and SimSiam [8], respectively on IN-100 dataset (200 epoch). It is worthy to point out that the series of ablation studies of our FT can boost the Sim-

pre-train	IN-1k	Faster [35] R50-C4 VOC			Mask R-CNN [15] R50-C4 COCO					
	Top-1	AP	AP ₅₀	AP ₇₅	AP ^{bb}	AP ₅₀ ^{bb}	AP ₇₅ ^{bb}	AP ^{mk}	AP ₅₀ ^{mk}	AP ₇₅ ^{mk}
random init*	-	33.8	60.2	33.1	26.4	44.0	27.8	29.3	46.9	30.8
supervised*	76.1	53.5	81.3	58.8	38.2	58.2	41.2	33.3	54.7	35.2
infomin*	70.1	57.6	82.7	64.6	39.0	58.5	42.0	34.1	55.2	36.3
mocoV1[14]	60.6	55.9	81.5	62.6	38.5	58.3	41.6	33.6	54.8	35.6
mocoV1+ours	61.9	56.1	82.0	62.0	39.0	58.7	42.1	34.1	55.1	36.0
mocoV2[7]	67.5	57.0	82.4	63.6	39.0	58.6	41.9	34.2	55.4	36.2
mocoV2+ours	69.6	58.1	83.3	65.1	39.5	59.2	42.1	34.6	55.6	36.5
mocoV2+mochi[20]	68.0	57.1	82.7	64.1	39.4	59.0	42.7	34.5	55.7	36.7
DetCo[53]	68.6	57.8	82.6	64.2	39.4	59.2	42.3	34.4	55.7	36.6
InsLoc[55]	-	57.9	82.9	65.3	39.5	59.1	42.7	34.5	56.0	36.8

Table 9. Object detection. All model are pre-trained for 200 epochs on ImageNet-1k. * means that the results are followed from respective papers [14, 42]. The COCO results of mocoV2 is from [20]. Our results are reported using the average of 5 runs.

CLR model. The experiments shows our FT is generic and robust for various contrastive models.

5.3. Evaluating the representation on ImageNet-1k

After ablations on IN-100 dataset, we use the best settings of α_{in} and α_{ex} to train a model on ImageNet-1k (IN-1K). Note that the dimension-level mix is not used for the experiments on IN-1K due to computational constraints. We apply our method on the baseline MoCo [14] and MoCoV2 [7], which are both trained on IN-1K with 200 epochs. The results and comparison are summarized in Tab 8. Our method improves MoCoV1 and MoCoV2 by 1.3% and 2.1% on Top-1 accuracy respectively which are significant on a large dataset like IN-1K. UnMix [38] and MoChi [20] are the methods that also leverage mixup to better aid the contrastive learning process. Notably, we can observe that our method with MoCoV2 can provide larger performance gain than UnMix and MoChi respectively.

5.4. Downstream Tasks

Fine-grained image classification We evaluate the efficacy on real world fine-grained classification datasets, *e.g.*, large scale long-tail iNaturalist2018 [43], CUB-200 [50] and FGVC-aircraft [30]. As shown in Tab 8, our FT significantly boosts the transfer performance on iNat-18, with 1.7% and 0.9% improvement based on MoCo and MoCoV2. Besides, our FT brings consistent improvement on CUB-200 and FGVC-aircraft.

Object Detection Recent works [48, 53, 54, 55, 59] have shown that the transfer accuracy of state-of-the-arts (SOTAs) models [4, 6, 42, 7, 14] on classification and detection are inconsistent and have low correlation, denoted as “task-bias”. One important reason is that pre-tasks of SOTA are specifically designed and optimized for classification, such as instance discrimination [51, 14] and clustering [4], leading to substantial enhancement on classification but slight

gain for detection. Therefore we evaluate our FT on detection/instance segmentation tasks. As summarized in Tab 9, our FT can boosts the baseline model MoCo-V2 on various datasets and metrics respectively. Our FT strongly improves the transfer accuracy on VOC [11] and MSCOCO [26]. Besides, our FT with MoCo-V2 can get slightly better accuracy than those contrastive models specifically designed for detection tasks, *e.g.*, DetCo[53] and InsLoc [55]. Moreover, our FT can get much better classification results than DetCo. Notice that our FT is not aiming at the local information during pre-task design, but more invariance from feature transformation. These experiments indicate that our FT is less task-bias than the pre-task-based contrastive models. The performance boosts suggest the efficacy and robustness of our proposed FT, and enable us to learn more “view-invariant” and discriminative representations.

6. Conclusions

In this work, we have developed a visualization tool to visualize the score distributions of positive and negative pairs. Leveraging this visualization tool, we can understand the inside of the contrastive learning process. More specifically, we discover significant observations inspiring our novel Feature Transformation, including positive extrapolation such that more hard positives are created for the training. Besides, we propose the interpolation among negatives, which makes full use of negatives and provides diversified negatives. The feature transformations enable to learn more view-invariant and discriminative representations. Experiments show that our proposed Feature Transformation can improve at least 6.0% accuracy on ImageNet-100 over MoCo, and about 2.0% accuracy on ImageNet-1K over the MoCoV2 baseline. Transferring to the downstream tasks successfully demonstrate our model is less task-bias. In our future work, we will explore more feature manipulation strategies with the help of our visualization tool.

References

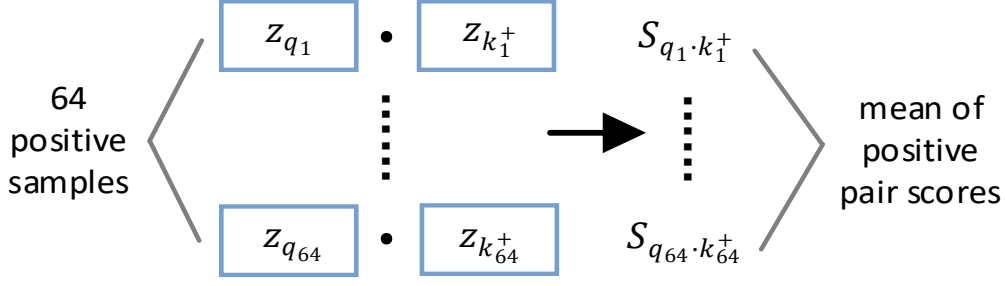
- [1] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In *Advances in Neural Information Processing Systems*, pages 15509–15519, 2019. 2
- [2] Qi Cai, Yu Wang, Yingwei Pan, Ting Yao, and Tao Mei. Joint contrastive learning with infinite possibilities. *NeurIPS*, 2020. 2
- [3] Yue Cao, Zhenda Xie, Bin Liu, Yutong Lin, Zheng Zhang, and Han Hu. Parametric instance classification for unsupervised visual feature learning. *NeurIPS*, 2020. 2
- [4] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *NeurIPS*, 2020. 1, 2, 3, 7, 8
- [5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014. 1
- [6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 1, 2, 3, 5, 6, 7, 8, 12, 13, 17
- [7] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 1, 2, 3, 7, 8, 13
- [8] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *CVPR*, 2021. 2, 7
- [9] Ching-Yao Chuang, Joshua Robinson, Lin Yen-Chen, Antonio Torralba, and Stefanie Jegelka. Debaised contrastive learning. *NeurIPS*, 2020. 2
- [10] Yueqi Duan, Wenzhao Zheng, Xudong Lin, Jiwen Lu, and Jie Zhou. Deep adversarial metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2780–2789, 2018. 2
- [11] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 8, 14
- [12] Jean-Bastien Grill, Florian Strub, Florent Althé, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doherty, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *NeurIPS*, 2020. 2, 3, 17
- [13] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304, 2010. 2
- [14] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020. 1, 2, 3, 5, 6, 7, 8, 12, 13, 14, 17
- [15] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 8, 14
- [16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 4, 6
- [17] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *ICLR*, 2019. 6
- [18] Qianjiang Hu, Xiao Wang, Wei Hu, and Guo-Jun Qi. Adco: Adversarial contrast for efficient learning of unsupervised representations from self-trained negative adversaries. 2021. 2
- [19] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *CVPR*, 2017. 1
- [20] Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. Hard negative mixing for contrastive learning. *Advances in Neural Information Processing Systems*, 33, 2020. 2, 6, 7, 8
- [21] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschiot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*, 2020. 17
- [22] Sungnyun Kim, Gihun Lee, Sangmin Bae, and Se-Young Yun. Mixco: Mix-up contrastive learning for visual representation. *arXiv preprint arXiv:2010.06300*, 2020. 2
- [23] Byungsoo Ko and Geonmo Gu. Embedding expansion: Augmentation in embedding space for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7255–7264, 2020. 2
- [24] Soroush Abbasi Koohpayegani, Ajinkya Tejankar, and Hamed Pirsiavash. Mean shift for self-supervised learning. *arXiv preprint arXiv:2105.07269*, 2021. 2
- [25] Kibok Lee, Yian Zhu, Kihyuk Sohn, Chun-Liang Li, Jinwoo Shin, and Honglak Lee. i-mix: A strategy for regularizing contrastive representation learning. In *International Conference on Learning Representations*, 2021. 2
- [26] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 8, 14
- [27] Xudong Lin, Yueqi Duan, Qiyuan Dong, Jiwen Lu, and Jie Zhou. Deep variational metric learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 689–704, 2018. 2
- [28] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 1
- [29] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 1
- [30] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013. 8

- [31] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 6
- [32] Chao Peng, Tete Xiao, Zeming Li, Yuning Jiang, Xiangyu Zhang, Kai Jia, Gang Yu, and Jian Sun. Megdet: A large mini-batch object detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6181–6189, 2018. 14
- [33] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 1
- [34] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 1, 14
- [35] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 8
- [36] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015. 1
- [37] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *NeurIPS*, 2018. 4
- [38] Zhiqiang Shen, Zechun Liu, Zhuang Liu, Marios Savvides, Trevor Darrell, and Eric Xing. Un-mix: Rethinking image mixtures for unsupervised visual representation learning. *arXiv preprint arXiv:2003.05438*, 2020. 2, 6, 7, 8, 13
- [39] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1
- [40] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204, 2017. 3
- [41] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *ECCV*, 2019. 1, 2, 3, 17
- [42] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning. *arXiv preprint arXiv:2005.10243*, 2020. 1, 2, 3, 4, 6, 7, 8, 13, 14
- [43] Grant Van Horn, Oisín Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8769–8778, 2018. 8
- [44] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning*, pages 6438–6447. PMLR, 2019. 2, 4, 5
- [45] Feng Wang, Huaping Liu, Di Guo, and Fuchun Sun. Unsupervised representation learning by invariance propagation. *NeurIPS*, 2020. 2
- [46] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, pages 9929–9939. PMLR, 2020. 2
- [47] Xudong Wang, Ziwei Liu, and Stella X Yu. Unsupervised feature learning by cross-level instance-group discrimination. In *CVPR*, 2021. 2
- [48] Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li. Dense contrastive learning for self-supervised visual pre-training. *CVPR*, 2021. 2, 8
- [49] Chen Wei, Huiyu Wang, Wei Shen, and Alan Yuille. Co2: Consistent contrast for unsupervised visual representation learning. *ICLR*, 2021. 2
- [50] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-ucsd birds 200. 2010. 8
- [51] 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. 2, 3, 8
- [52] Tete Xiao, Xiaolong Wang, Alexei A Efros, and Trevor Darrell. What should not be contrastive in contrastive learning. *ICLR*, 2021. 2
- [53] Enze Xie, Jian Ding, Wenhai Wang, Xiaohang Zhan, Hang Xu, Zhenguo Li, and Ping Luo. Detco: Unsupervised contrastive learning for object detection. *arXiv preprint arXiv:2102.04803*, 2021. 2, 8
- [54] Zhenda Xie, Yutong Lin, Zheng Zhang, Yue Cao, Stephen Lin, and Han Hu. Propagate yourself: Exploring pixel-level consistency for unsupervised visual representation learning. *CVPR*, 2021. 2, 8
- [55] Ceyuan Yang, Zhirong Wu, Bolei Zhou, and Stephen Lin. Instance localization for self-supervised detection pretraining. *CVPR*, 2021. 2, 8
- [56] Mang Ye, Xu Zhang, Pong C Yuen, and Shih-Fu Chang. Unsupervised embedding learning via invariant and spreading instance feature. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6210–6219, 2019. 2
- [57] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6023–6032, 2019. 2
- [58] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *ICLR*, 2018. 2, 5
- [59] Nanxuan Zhao, Zhirong Wu, Rynson WH Lau, and Stephen Lin. What makes instance discrimination good for transfer learning? *NeurIPS*, 2020. 2, 8

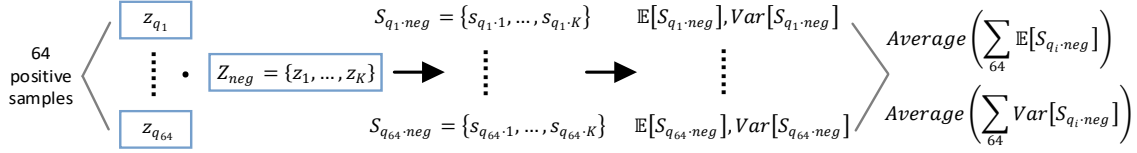
Appendix

Contents

1. Introduction	1
2. Related Work	2
3. Visualization of Contrastive Learning	3
3.1. Preliminaries	3
3.2. Score Distribution Visualization	3
3.3. Visualization Examples with MoCo	3
4. Proposed Feature Transformation Method	4
4.1. Positive Extrapolation	4
4.2. Negative Interpolation	5
4.3. Discussions	6
5. Experiments	7
5.1. Ablation study	7
5.2. FT on various contrastive models	7
5.3. Evaluating the representation on ImageNet-1k	8
5.4. Downstream Tasks	8
6. Conclusions	8
A Details of the Visualization Tool	12
B Experimental Details of each Part in the Paper	13
B.1. Data Augmentations	13
B.2. Implementation of Visualization experiments	13
B.3. Implementation on ImageNet-100	13
B.4. Implementation on ImageNet-1k	13
B.5. Implementation on Fine-grained Classification	13
B.6. Object detection on PASCAL VOC	14
B.7. Object detection and Instance segmentation on MSCOCO	14
C Details of the Gradient Landscape	14
D Discussion of when to Add the Feature Transformation	15
D.1. Effectiveness of our FT	15
D.2. FT in the Early Training Stage	15
E Details of Comparison to other Methods	17
E.1. Additional experiments of our proposed feature transformation methods on SimCLR	17
E.2. Apply Positive Extrapolation on Non-contrastive Models on IN-1K	17
F. Discussion of the feature normalization for FT	17
F.1. Importance of Unit-sphere Projection	17
F.2. Whether to add ℓ_2 Normalization after FT	18
G Discussion of the Negative Feature Transformation	19
G.1. Negative Extrapolation in Memory Queue	19
G.2. Creating Hard Negatives	20



(a) Mean of positive pair scores.



(b) Mean and variance of negative pair scores.

Figure 8. Illustrations of pos/neg pair score visualization. (a) Mean of positive pair scores. (b) Mean and variance of negative pair scores.

A. Details of the Visualization Tool

We choose three non-trivial statistics to visualize the score distribution: the mean of pos/neg scores (denoted as $Mean(pos)/Mean(neg)$, indicating the approximate average of the pos/neg pair distance) and the variance of negative scores (denoted as $Var(neg)$, indicating the fluctuation degree of the negative samples in the memory queue).

Without loss of generality, we randomly choose 64 samples⁶ in one batch to calculate the statistics data and perform visualization. (1) For the positive pair score: As shown in Fig 8(a), we denote the z_{q_i} , ($i = 1, 2, 3 \dots 64$) as the 64 query samples. And $z_{k_i^+}$, are the corresponding positive features of z_{q_i} . Then we can get 64 positive score $S_{q_i \cdot k_i^+}$, by inner product. Finally, we retain the mean value of these 64 positive scores as $Mean(pos)$. (2) For the negative pair scores: As shown in Fig 8(b), we denote the $Z_{neg} = \{z_1, z_2, \dots, z_K\}$ where K is the size of the memory queue⁷. Each z_{q_i} combining Z_{neg} will create K negative pair scores in a set, named $S_{q_i \cdot neg} = \{s_{q_i \cdot 1}, s_{q_i \cdot 2}, \dots, s_{q_i \cdot K}\}$. To keep all the $64 \times K$ negative scores is challenging (about 4TB storage for the pair scores), so for each $S_{q_i \cdot neg}$, we retain their mean and variance to show the distribution of K negative sample scores corresponding to z_{q_i} . More generally, we further average these 64 means and variances to show the statistical characteristics of these K negative samples ($Mean(neg)$ and $Var(neg)$). These statistics are recorded at each training step to track the score distribution in the training process.

Our visualization is very practical. It is offline, which almost does not affect the training speed. Instead of storing K (65536) pair scores, we save their statistical mean & variance to represent the scores' distribution. As a result, it only takes about 20MB storage and 5 minutes extra time for a 256 batch-size 100 epoch training. Even with larger datasets and batch size, it's still feasible.

⁶We usually apply batch-size 256 on 4-GPU servers. Here we collect one batch 64 on a single GPU for statistics.

⁷ K is a large number, e.g., 65536 in MoCo [14] and the largest 8192×2 in one batch for SimCLR [6]. We use $K = 65536$ in all the MoCo experiments.

B. Experimental Details of each Part in the Paper

The experiments are mainly implemented using the code from InfoMin [42]⁸. The transfer experiments on object detection and instance segmentation are implemented using Detectron2⁹. We keep the fairness of the experiments, especially when compared with other methods. The code of our proposed methods and visualization tools will be made public.

B.1. Data Augmentations

For the experiments of combining our feature transformation module with other contrastive learning methods, we use the same image-level data augmentation strategies as the respective methods. Specifically, for our visualization experiments and other experiments using MoCo, we use the same data augmentation strategies with MoCo which contains *Random Resized Crop*, *Horizontal Flip*, *ColorJitter*, and *Random Gray Scale*. For the experiments on MoCoV2 [7] and SimCLR [6], the data augmentation strategies are the same which contains *Random Resized Crop*, *Horizontal Flip*, *ColorJitter*, *Random Gray Scale*, and *Gaussian Blur*.

B.2. Implementation of Visualization experiments

For training All the visualization experiments are carried on ImageNet-100 and ResNet-18 for fast evaluation and parameters-tuning experiments. For the visualization experiments (including Table 1, Table 6 (2nd row), figure 1(a), 3, 4, 5, 7 in the paper and Table 11 (2nd row), 14, figure 9, 10, 11 in supplementary materials), we apply a mini-batch size of 256 is used with 4-GPUs, where the number of negative examples is set to 65,536, with initial learning of 0.03. And we use $256/4 = 64$ samples to perform visualizations. For the fast grid experiments, the model is trained for only 100 epochs with the learning rate multiplied by 0.1 at 60 and 80 epochs. We use SGD as the optimizer, the weight decay of SGD is 0.0001 and the momentum of SGD is 0.9. And for various unit-sphere projection experiments, we apply 200 epochs training to perform visualization.

For testing we use the linear readout protocol to evaluate the trained representation on the validation set by fixing the learned representation and train a supervised linear classifier on the representations, the single-crop top-1 accuracy on the validation set is reported. An initial learning rate of 10 and weight decay 0. The classifier is trained with 100 epochs and the learning rate is multiplied by 0.1 at 60, and 80 epochs.

B.3. Implementation on ImageNet-100

For training we use ResNet-50 for ImageNet-100 implementations. And momentum parameter is set to be 0.99 for our experiments. (including Table 2, 3, 4, 5, 6 (2nd row), 7 in the paper and Table 11 (1st row), 12, 15, 16 in supplementary materials). A mini-batch size of 256 is used with 8-GPUs, where the number of negative examples is set to 65,536, with initial learning of 0.03. The model is trained for 200 epochs with the learning rate multiplied by 0.1 at 120 and 160 epochs. We use SGD as the optimizer, the weight decay of SGD is 0.0001 and the momentum of SGD is 0.9.

For testing we use the linear readout protocol to evaluate the trained representation on the validation set by fixing the learned representation and train a supervised linear classifier on the representations, the single-crop top-1 accuracy on the validation set is reported. We use an initial learning rate of 10 and weight decay 0. The classifier is trained with 60 epochs and the learning rate is multiplied by 0.1 at 30, 40, and 50 epochs following [38].

B.4. Implementation on ImageNet-1k

For training The momentum update parameter m for the experiments on ImageNet-1k is set to 0.999, other parameters are set to the same as the experiments on ImageNet-100. ResNet-50 is used as an encoder. (including Table 8 in the paper and Table 13 in supp material). We can observe that the best result for positive extrapolation and negative interpolation is achieved when α_{in} and α_{ex} are set to 1.6 and 2.0 respectively. Thus we use this value for the other experiments. Except otherwise stated, other hyper-parameters are set to be the same with MoCo [14] and MoCoV2[7].

For testing The same linear readout protocol is used where the linear classifier is trained for 100 epochs and the initial learning rate is 30 which are multiplied by 0.1 at 60, 80 epochs.

B.5. Implementation on Fine-grained Classification

In addition to object detection and instance segmentation tasks, we also provide a study of fine-grained classification. We choose three challenging fine-grained datasets to conduct the experiments, iNaturalist 2018 dataset, CUB-200 dataset,

⁸<https://github.com/HobbitLong/PyContrast>

⁹<https://github.com/facebookresearch/detectron2>

and FGVC-aircraft dataset. (1) The iNaturalist 2018 has 437k images and 8142 classes, this dataset is commonly used for fine-grained classification and long-tailed recognition, and is used by several papers for evaluating the transfer performance of self-supervised representations [14]. (2) The CUB-200 dataset contains 6033 images belong to 200 bird species and is used for fine-grained classification. (3) The FGVC-aircraft dataset has 10,200 images of aircraft, with 100 images for each of 102 different aircraft model variants, most of which are airplanes. When transferring to these datasets, the pre-trained model is fine-tuned with 100 epochs, the learning rate is set to 5e-3 with cosine decay.

B.6. Object detection on PASCAL VOC

The main goal of self-supervised pre-training is to obtain representation that can be beneficial for downstream tasks. We choose to use PASCAL VOC [11] and COCO [26] as our benchmark for testing the transfer performance of the representation to object detection and instance segmentation tasks following previous works [14]. For PASCAL VOC dataset, we use the `trainval07+12` split for fine-tuning, and the `test2007` split for evaluating. The image scale is set to [480, 800] pixels for training and 800 for testing. For COCO dataset, we use the `train2017` split (118k images) for fine-tuning, the `val2017` split for evaluating. The image scale is set the same with PASCAL VOC.

When transferring to detection tasks, feature normalization has been shown to be crucial during fine-tuning [14]. Therefore, the pre-trained backbone is fine-tuned with Synchronized BN (SyncBN) [32] and add SyncBN to the FPN layer following [14]. We use Faster R-CNN [34] with R50-C4 architectures for object detection on the PASCAL VOC dataset. All layers of the model are fine-tuned with 24,000 iterations with each batch consisting of 16 images. The initial learning is set to 0.02 and is multiplied by 0.1 at 18,000 and 22,000 iterations. Other hyper-parameters are set to be the same with [14].

B.7. Object detection and Instance segmentation on MSCOCO

We also tested the transferring abilities of the pre-trained model using the instance segmentation tasks on MS COCO dataset. We use a Mask R-CNN [15] R50-FPN pipeline following [42]. The batch size is set to 16 with the learning rate as 0.02, the model is trained with 1x and 2x schedules, for 1x schedules, the model is trained for 90,000 iterations on the MS COCO datasets with the learning rate multiplied by 0.1 at 60,000 and 80,000 iterations, for the 2x schedules, we use 180,000 iterations with the learning rate multiplied by 0.1 at 120,000 and 160,000 iterations. The transfer results of the 2x schedule is provided in Tab 10. Other hyper-parameters are set to be the same with [14].

Method	Performance					
	AP ^{bb}	AP ^{bb} ₅₀	AP ^{bb} ₇₅	AP ^{mk}	AP ^{mk} ₅₀	AP ^{mk} ₇₅
mocov1	40.7	60.5	44.1	35.4	57.3	37.6
mocov1+ours	41.5	61.0	44.5	35.9	57.7	38.0
mocov2	40.9	60.7	44.4	35.5	57.5	37.9
mocov2+ours	41.3	60.9	44.8	35.7	57.8	38.1

Table 10. COCO object detection and instance segmentation based on Mask-RCNN-FPN with 2x learning rate schedule. Our results are reported using the average of 3 runs.

C. Details of the Gradient Landscape

We provide the details of our gradient landscape Figure 4 of various m in the paper. As shown in Fig 9, we provide ℓ_2 norm for each layer of the encoder (ResNet-18) with the training process. X axis indicates the layers of the encoder, while Y axis indicates the 100 training epochs. And Z axis means the value of ℓ_2 norm. We choose the ℓ_2 norm of this layer (total gradient ℓ_2 norm of this layer) because the ℓ_2 norm of gradient is very obvious to show the smoothness of gradient landscape. We can see that small $m = 0.6$ and 0.5 brings drastic volatility with the training process. The corresponding loss value and the gradient will fluctuate violently, resulting in bad convergence. As shown in Fig 9, the smooth and stable gradient landscape of $m = 0.99$ (Fig 9(a)) becomes sharp and messy with the decrease of m (Fig 9(b) for $m = 0.6$ and Fig 9(c) for $m = 0.5$). Therefore, to learn a better pre-trained model, we need to prepare negative pairs that can maintain the stability and smoothness of score distribution and gradient for the training process. It seems that the gradient landscape looks spiky: 1) Across Y axis indicating the training epochs. 2) Across X axis representing the ResNet layers, it shows the gradients of all layers including the BatchNorm layer whose gradient is small. But the gradient of Convolution layer is large, thus it seems to be spiky across X axis. The spiky gradient on X axis doesn't influence the training, while the smooth gradient on Y axis matters.

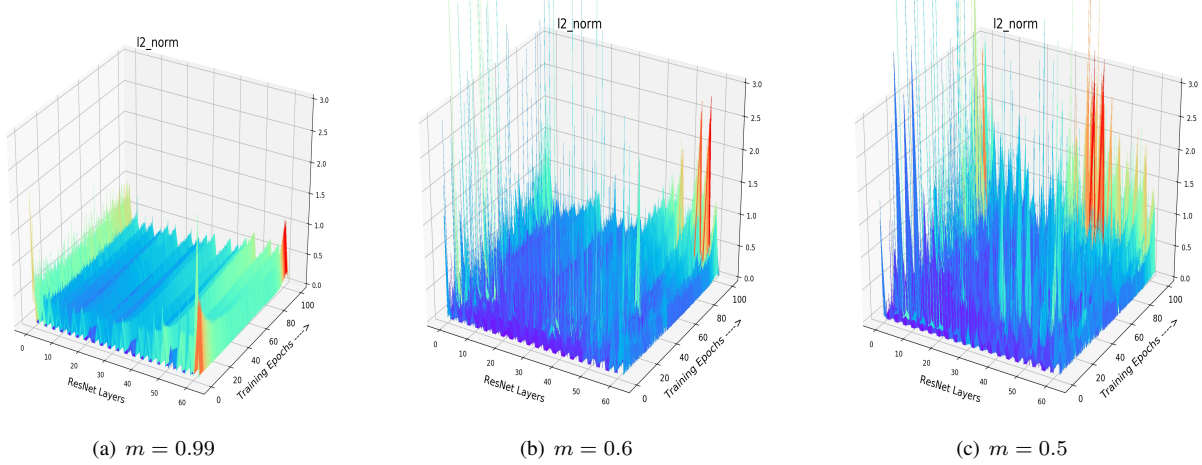


Figure 9. Gradient (ℓ_2 norm) landscape of various m . We provide ℓ_2 norm for each layer of the encoder (ResNet-18) with the training process. Across Y axis indicating the training epochs. Across X axis representing the ResNet layers, it shows the gradients of all layers including the BatchNorm layer whose gradient is small. But the gradient of Convolution layer is large, thus it seems to be spiky across X axis. And Z axis means the value of ℓ_2 norm. The spiky gradient on X axis doesn't influence the training, while the smooth gradient on Y axis matters. We can see that small $m = 0.6$ and 0.5 brings drastic volatility with the training process. The corresponding loss value and the gradient will fluctuate violently, resulting in bad convergence.

D. Discussion of when to Add the Feature Transformation

D.1. Effectiveness of our FT

We present the efficacy of FT by analysis of starting FT in various training stages. As shown in Tab 11, starting FT (pos extrapolation + neg interpolation) from various epoch can boost the accuracy of baseline, and starting from earlier can improve more (7.1%/5.8% boosts with Res-18/Res-50). It is worthy to note that even adding FT in the 80th epoch can bring 3% improvement compared with the MoCo baseline (No FT in training). With the visualizations of score distribution Fig 10, we can see that our FT not only brings hard positives (lowering pos scores in Fig 10(d)) and hard negatives (rising neg scores in Fig 10(c)) simultaneously when the combined FT is inserted in various stages. The combination of positive extrapolation and negative interpolation can help rise the neg scores in the training process. Besides, with the comparison of the Gradient (ℓ_2 norm) landscape, we can observe that our FT brings a greater gradient for the training (Adding FT in the 30th epoch Fig 10(h) and 50th epoch Fig 10(i)), which makes the model escape from the local minima and avoid over-fitting. These analyses indicate our FT is a plug-and-play method and brings persistent view-invariance and discrimination for the training of contrastive models.

D.2. FT in the Early Training Stage

Due to the memory queue is initialized by random vectors at the start of training, the positive score and negative score have confusion, as shown in the visualizations in the early training stage (Fig 10(a) and Fig 10(b)). We provide the visualizations in the first 10 epoch to see the score distribution: (1) Adding FT from the 0th epoch will bring negative pairs whose score is very high (blue line in Fig 10(a), 0.8 negative score, which is too large for negative pairs), indicating the feature transformations for the random vectors will hurt the pair score distribution. From the perspective of gradient landscape in Fig 10(f), the initial gradient brought by FT is too sharp and not smooth for training compared with the baseline MoCo in Fig 10(e). (2) Adding FT from the 2nd epoch (In the 2nd epoch, the memory queue is filled by the semantic features from training data rather than the random vectors) will relieve solve too high negative scores (orange line in Fig 10(a), normal negative score) and meanwhile lower the positive score from easy positive to hard one (orange line in Fig 10(b), decreasing the positive score). The gradient (Fig 10(g)) seems more smooth and stable compare with starting FT from 0th epoch (Fig 10(f)). More importantly, in Tab 11, starting from the 2nd epoch (63.3%) can achieve slightly better accuracy than that at the beginning (62.6%). However, in the final experiments of imagenet-1K, we still use the strategy of starting FT from the 0th epoch. Because there seems no obvious performance difference in the ResNet-50 backbone in Tab 11. Future work will focus more on this issue.

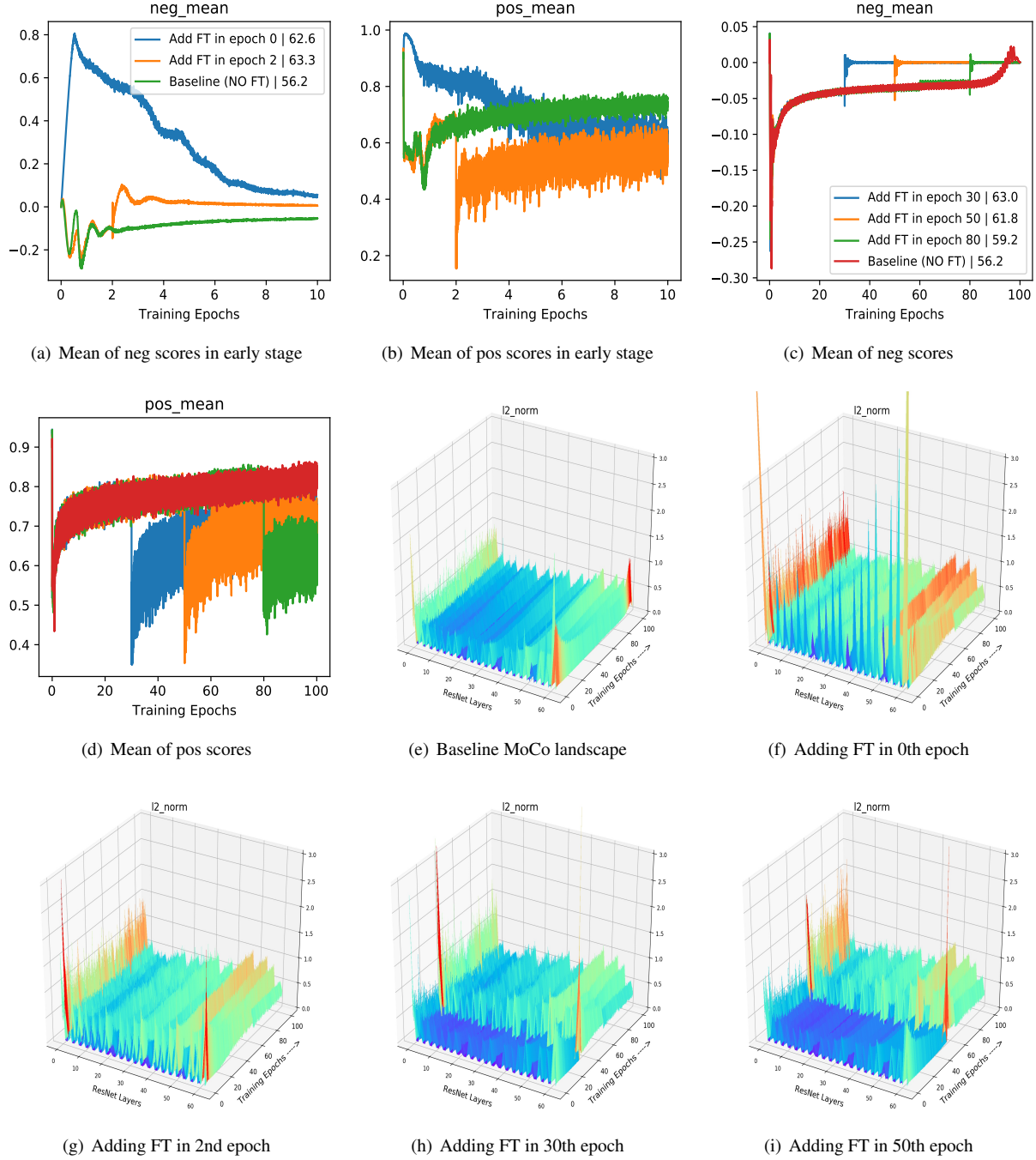


Figure 10. Visualization of when to add FT, including score distribution and Gradient (ℓ_2 norm) landscape.

FT begin epoch	0	2	30	50	80	-
Res18 acc (%)	62.6	63.3	62.9	61.8	59.2	56.2
Res50 acc (%)	76.9	76.4	75.9	74.0	72.2	71.1

Table 11. When to add feature transformation. We employ Res-18 (total 100 epochs) and Res-50 (total 200 epochs) on IN-100 for the results. '-' indicates MoCo baseline without using any FT.

method	arch	acc
SimCLR	r50	74.32
SimCLR+pos extrapolation	r50	75.80
SimCLR+neg interpolation	r50	76.71
SimCLR+both	r50	78.25
SimCLR+both _{dimension}	r50	78.81

Table 12. Performance comparison of our proposed two feature transformation module on imagenet-100 with SIMCLR, the model are trained for 200 epochs. the last line with both_{dimension} is mixing the feature (both pos/neg) using the dimension-level mixing, which shows improvements over the feature-level mixing.

Method	SimSiam	BYOL
baseline*	68.1	66.5
+pos extrapolation	68.7	67.2

Table 13. Comparison studies of proposed methods with non-contrastive methods. The models are pre-trained for 100 epochs with Res50 on IN-1K. * indicates reproduced baseline results.

E. Details of Comparison to other Methods

In this section, we discuss the details of how to apply our feature transformation to other self-supervised methods. We evaluate the performance of feature transformation on three representative methods, namely InfoMin, SwAV, and SimSiam.

For feature transformation on InfoMin, we perform both positive extrapolation and negative interpolation. Note that we perform the feature transformation on both branches of the InfoMin method, i.e. the original branch and the JigSaw branch. For feature transformation on SwAV, we only transform the two features of the input image by positive extrapolation, the rest of the SwAV pipeline is left unchanged. For SimSiam, as the method only uses positive pairs for training, so we only apply the positive extrapolation as the feature transformation. All the other hyperparameters are set to be the same as the original paper of each self-supervised method.

E.1. Additional experiments of our proposed feature transformation methods on SimCLR

To demonstrate the effectiveness of our feature transformation methods (Negative feature interpolation and Positive feature extrapolation), we also provide the experimental results on ImageNet-100 [41] of applying our method on another classic contrastive learning model, SimCLR [6]. Instead of using two encoders for encoding q and k like in MoCo [14], SimCLR directly uses a single network to encode the two views and contrast them against other negative examples. Because both MoCo and SimCLR are contrastive-based methods, the negative interpolation and positive extrapolation strategies can also be applied to SimCLR. We show the results of combining negative interpolation and positive extrapolation in Tab 12.

E.2. Apply Positive Extrapolation on Non-contrastive Models on IN-1K

Here we complement the results of applying positive extrapolation on non-contrastive models [12, 6]. The models are pre-trained for 100 epochs on IN-1K with the same data augmentation setting of the original paper. As shown in Table 13, we provide the IN-1k results (100ep) of BYOL/BYOL+posFT (66.5% -> 67.2%) and SimSiam/SimSiam+posFT (68.1% -> 68.7%) indicating pos extrapolation alone can help BYOL and SimSiam. Notice that we didn't perform the parameter experiments (not the optimal extrapolation parameter α_{ex}), so the improvement is slight.

F. Discussion of the feature normalization for FT

Here we provide additional visualization and analysis on the regular Feature Transformation (feature normalization, ℓ_2 normalization) due to its significant constriction (unit-sphere projection) and Whether to add ℓ_2 Normalization after our proposed FT.

F.1. Importance of Unit-sphere Projection

Unit-sphere projection (ℓ_2 norm) constricts the feature vector length from unbounded to 1, in the meanwhile retains the vector direction. Thus the pair scores $S_{q,k}$ can be limited to $[-1, 1]$. Recent paper [21] concludes that unit sphere projection plays a key role in ensuring the large gradients of hard positives and negatives from the loss gradient properties. However,

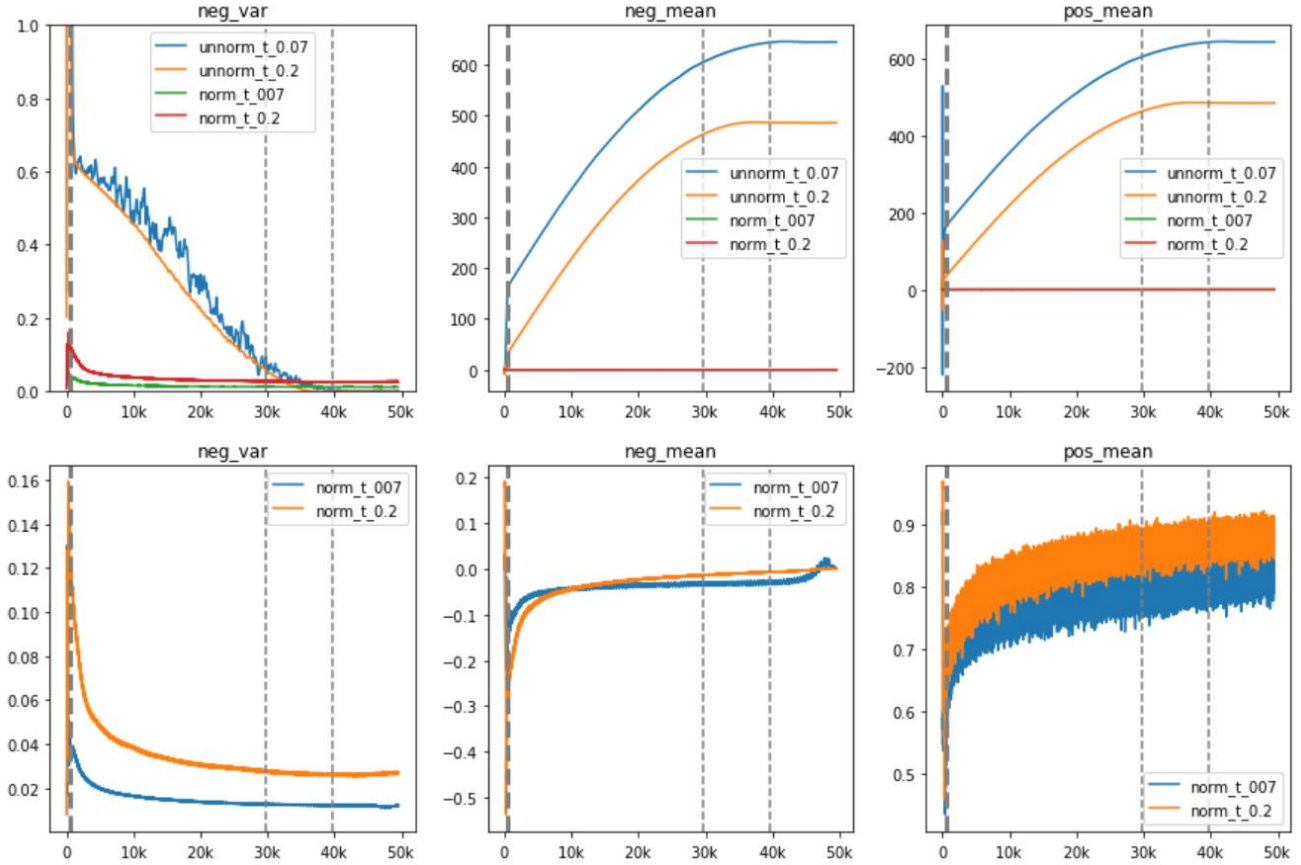


Figure 11. Pos/neg pair score distribution of unit sphere projection on ImageNet-100

Method	τ	lr	Acc%
MoCo w/ unit sphere proj	0.07	0.03	65.04
MoCo w/ unit sphere proj	0.2	0.03	63.06
MoCo w/o unit sphere proj	0.07	0.03/10	<i>collapse</i>
MoCo w/o unit sphere proj	0.2	0.03/10	<i>collapse</i>

Table 14. The experiments for unit sphere projection on ImageNet-100

without the unit sphere projection, the feature vector length lost the constriction to $[-1, 1]$, and the too-large score distribution leads to bad contrastive learning and poor transfer performance (65.04% *v.s.* *modelcollapse*). As shown in our empirical study (Fig 11 and Table 14) of this significant FT, the mean of positive pair score is similar to the mean of negative pair score when we removed unit-sphere projection, which will lead to an awful contrastive learning process: confusing the pos/neg pairs and bad gradient landscape brought by too large score distribution. Meanwhile, with the constrictions of unit-sphere projection, the mean of pos/neg pair scores are as expected: $neg \in [-0.2, 0]$ and $pos \in [0.6, 0.9]$, which can be discriminated by the log-softmax loss function. This limited small score distribution benefits the later contrastive learning and brought a stable training process. Finally, the variance of the negative pair score shows that model with unit-sphere projection will provide less volatile negative pairs, which is better for contrastive learning.

F.2. Whether to add ℓ_2 Normalization after FT

In this section, we provide empirical studies about whether re-perform the ℓ_2 norm for the transformed features after FT. As shown in Tab 15, the performance difference is negligible for the model with/without re-performing the ℓ_2 normalization, (74.64% *v.s.* post-norm 74.82% for negative interpolation, 72.80% *v.s.* post-norm 72.45% for positive extrapolation, 76.87%

Method (MoCov1)	Acc%
baseline*	71.10
+pos extrapolation	72.80
+pos extrapolation _{norm}	72.45
+neg interpolation	74.64
+neg interpolation _{norm}	74.82
+both	76.87
+both _{norm}	76.68
+neg extrapolation	71.84
+neg extrapolation _{norm}	71.95

Table 15. Ablation studies of proposed methods on various contrastive models. The model are pre-trained for 200 epochs with Res50 on IN-100. The line with *norm* is re-normalizing the transformed feature to the unit sphere, which show no improvements. * indicates reproduced baseline results.

Method (MoCov1)	Beta parameter	Acc%
baseline*	-	71.10
+neg interpolation	$Beta(1.6, 1.6)$	74.64
+neg extrapolation	$Beta(2.0, 2.0)$	71.84
+hard negative	$Beta(2.0, 1.0)$	73.45
+hard negative	$Beta(5.0, 2.0)$	74.32

Table 16. Ablation studies of proposed methods on various contrastive models. The model are pre-trained for 200 epochs with Res50 on IN-100. The line with *norm* is normalizing the transformed feature to the unit sphere, which show no improvements. * indicates reproduced baseline results.

v.s post-norm 76.68% for combined FT). So we conclude that the transformed features are not necessarily on the unit sphere (*i.e.* has a norm of 1) due to the negligible performance difference. And in the final experiments of imagenet-1K, we do not re-perform the ℓ_2 norm after feature transformations. **However, we strongly recommend to re-perform ℓ_2 norm for the transformed features on all the datasets, for the sake of contrasting all the scores on the unit-sphere.**

G. Discussion of the Negative Feature Transformation

In this section, we provide more discussions about the feature manipulation of the negative examples. We have discussed negative interpolation to fully utilize negative features and increase the diversity of the memory queue. Here we provide the situation about negative extrapolation in memory queue and creating hard negatives.

G.1. Negative Extrapolation in Memory Queue

We have explored the negative interpolation to fully utilize negative features and increase the diversity of the memory queue. Then how about the negative extrapolation in the memory queue? Will the extrapolated negatives still be effective to increase the diversity of the memory queue and the performance?

Specifically, we denote the negative memory queue of MoCo as $Z_{neg} = \{z_1, z_2, \dots, z_K\}$ where K is the size of the memory queue, and Z_{perm} as the random permutation of Z_{neg} . We propose to use a simple extrapolation between two memory queue to create a new queue $\hat{Z}_{neg}^{ex} = \{\hat{z}_1^{ex}, \hat{z}_2^{ex}, \dots, \hat{z}_K^{ex}\}$:

$$\hat{Z}_{neg}^{ex} = \lambda_{ex} \cdot Z_{neg} + (1 - \lambda_{ex}) \cdot Z_{perm} \quad (7)$$

where $\lambda_{ex} \sim Beta(\alpha_{ex}, \alpha_{ex}) + 1$ is in the range of (1, 2). The transformed memory queue \hat{Z}_{neg} provides fresh extrapolated negatives for contrastive loss iteration by iteration. As shown in Tab 16, the negative extrapolation brings slight improvement over baseline (71.84% v.s. 71.10, 0.74% improved), while negative interpolation significantly improves to 74.64%. Both the negative interpolation and extrapolation can increase the diversity of the memory queue, but why extrapolation cannot boost the performance? We conjecture that the original queue Z_{neg} provides discrete distribution of negative samples but our method can fill in the incomplete sample points of the distribution by random interpolation, leading to a more discriminative model. But the extrapolated sample points may not stay in the previous manifold/distribution. Future work will focus more on this discussion.

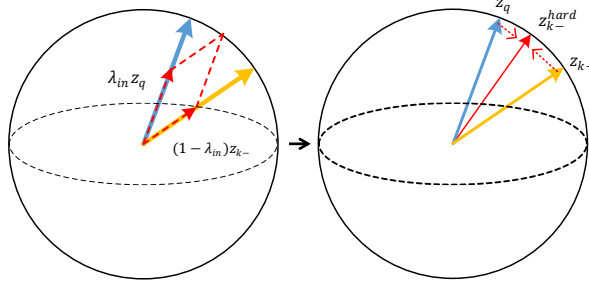


Figure 12. The process of creating hard negatives. The distance between the pos/neg feature vector is lowered, changing easy negatives to hard negatives, which is better for contrastive learning.

G.2. Creating Hard Negatives

The negative interpolation and extrapolation are both performed in the memory queue to increase the diversity. Another feature transformation for negative features is to increase the hardness during training, like the way of positive extrapolation. Our goal is to increasing the easy negative pair scores (similarity) to create hard negative pairs during training could be beneficial for the final transfer performance. Specifically, we use interpolation between z_q and all the negatives in the memory queue $Z_{neg} = \{z_1, z_2, \dots, z_K\}$ to create a hard negative queue $Z_{neg}^{hard} = \{z_1^{hard}, z_2^{hard}, \dots, z_K^{hard}\}$.

$$Z_{neg}^{hard} = \lambda_{in} \cdot z_q + (1 - \lambda_{in}) \cdot Z_{neg} \quad (8)$$

This equation indicates that each negative sample in the memory queue Z_{neg} will be interpolated with z_q to create hard negative queue Z_{neg}^{hard} . And $\lambda_{in} \sim \text{Beta}(\alpha_{in}, \alpha_{in})$ is in the range of $(0, 1)$. By this transformation, we can guarantee that the transformed neg score $S_{q,k-}^{hard}$ is larger than the original pos score $S_{q,k-}$, namely $z_q z_{k-}^{hard} \geq z_q z_{k-}$, which means we create a hard negative queue. Intuitively, it can be seemed like a simple approach to draw z_q and z_{k-} closer in feature space. After interpolation, the distance between the pos/neg feature vector is lowered. Therefore this interpolation can serve as a feature transformation to create hard negatives from easy ones. As shown in Fig 12, it brings a minor direction change for positive/negative vectors. As shown in Tab 16, our hard negatives can bring consistent boosts over the baseline (74.32% v.s. (71.10%, 3.22% improved), indicating that this hard negative is effective for the contrastive learning. Future work will focus more on this topic. However, we choose the negative interpolation rather than the hard negative strategy in the final experiments of IN-1K. Because the computation of hard negative strategy is too large (Each z_q needs a new hard negative queue, so it takes time for one large batch to produce hard negative queue.).