Improving Contrastive Learning by Visualizing Feature Transformation

Warley Vital Barbosa

June 7, 2024

Overview

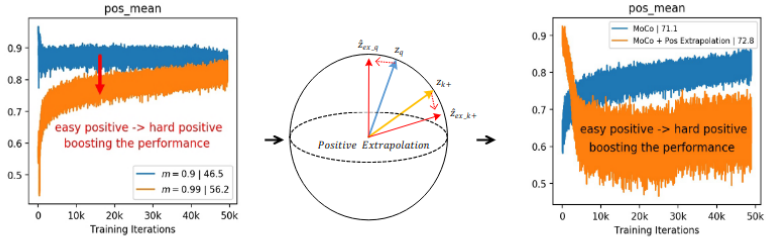
Overview

1. Introduction

Introduction

- **Contrastive Learning**
  - Representation learning: positive pairs closer than negative pairs.
    - Self-supervised Contrastive Learning – comparable transfer performance without annotations.
- The **importance** of positive and negative pairs
  - How to best **design** these pairs?
  - Traditional methods primarily use **data augmentation** techniques.
- **Limitations** of data augmentation
  - Human intuition
  - **Lack of a deeper understanding** of the feature space transformations
  - Most **informative or challenging pairs** aren't necessarily generated
- This work enhances contrastive learning via feature-level manipulation, not data augmentation.

Motivation of Visualizing the Score Distribution



(a) Observation (b) Proposed Method (c) Performance Gain

Figure: From the authors. (a) It draws the score distribution of positive pairs for m (the momentum in MoCo[14]). (b) Inspired by (a), they apply extrapolation on positive pairs to slightly decrease the scores, generating harder positives. (c) Leveraging the extrapolation of positives, they improve the performance. 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).

Key Contributions

- **Visualization Tool**
  - First tool to analyze contrastive learning using score distributions.
  - *VISUALIZATION* $\rightarrow$ *INSIGHTS*
- **Feature Transformation**
  - **Positive Extrapolation** - new harder positives by increasing view variance.
  - **Negative Interpolation** - diversifies negatives to enhance model discriminativeness.
- **Experimental Validation**
  - Significant accuracy improvements on ImageNet-100 and ImageNet-1K.
  - Less task-bias in downstream tasks.

Overview

2. Related Work

Related Work

Contrastive Learning

- **Instance Discrimination**:
  - *Methods*: MoCo (He et al., 2020), SimCLR (Chen et al., 2020).
  - *Focus*: Distinguishing different instances for robust representations.
- **Lower Bound of NCE**:
  - *Method*: InfoMin (Tian et al., 2020).
  - *Insight*: Increased data augmentations decrease mutual information, enhancing transfer performance.
- **Self-supervised Learning**:
  - *Methods*: BYOL (Grill et al., 2020), SimSiam (Chen & He, 2021).
  - *Achievement*: High performance without negative pairs.

Related Work

MixUp for Contrastive Learning

- **Manifold MixUp**:
  - *Proposed by*: Verma et al. (2019).
  - *Concept*: Feature-level regularization for supervised learning.
- **Un-mix**:
  - *Proposed by*: Shen et al. (2020).
  - *Concept*: Mixup in image/pixel space for self-supervised learning.
- **MoChi**:
  - *Proposed by*: Kalantidis et al. (2020).
  - *Concept*: Mixup in the embedding space to generate hard negatives, with mixed impact on classification accuracy.

Related Work

Additional Relevant Techniques

- **Propagate Yourself**: Xie et al., 2021.
- **DetCo**: Xie et al., 2021.
- **Mean Shift for Self-Supervised Learning**: Koohpayegani et al., 2021.

Overview

3. Visualization of Contrastive Learning

Learning Pipeline

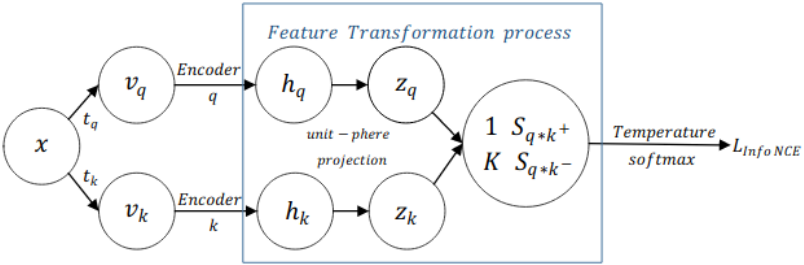


Figure: From the authors. Feature Transformation Contrastive Learning Pipeline.

Preliminaries

- Each data sample x undergoes **two separate data augmentations**, resulting in two views v_q and v_k.
- The encoder maps these **views** into a feature embedding space.
- Feature vectors are **normalized** onto the **unit sphere**.
- Positive pair score $S_{q \cdot k^+}$ and negative pair scores $S_{q \cdot k^-}$ are computed as **cosine similarities**.
- These scores are used in the **InfoNCE loss** for contrastive learning:

$$L = -\log \left[\frac{\exp(S_{q \cdot k^+}/\tau)}{\exp(S_{q \cdot k^+}/\tau) + \sum_K \exp(S_{q \cdot k^-}/\tau)} \right]$$

Score Distribution Visualization

- **Motivation**:
  - Score distributions provide deeper **insights into the learning dynamics** than loss curves or transfer accuracy alone.
  - Cosine similarity scores $S_{q \cdot k}$ are one-dimensional, limited to $[-1, 1]$, and **suitable for detailed observation**.
- **Visualization Method**:
  - **Offline Tool**: **Negligible computation overhead**, does not affect training speed.
  - **Visualized Metrics**:
    - Mean of positive scores.
    - Mean and variance of negative scores.

Visualization Examples with MoCo

- **Momentum Update Mechanism**:
  - MoCo uses a momentum update mechanism to reduce **inconsistency in the memory queue**.
  - Encoder f_k is updated using:

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

  - m (momentum coefficient) **affects** final transfer accuracy and model consistency.

Visualization Examples with MoCo

- **Momentum Update Mechanism**:
  - MoCo uses a momentum update mechanism to reduce **inconsistency in the memory queue**.
  - Encoder f_k is updated using:

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

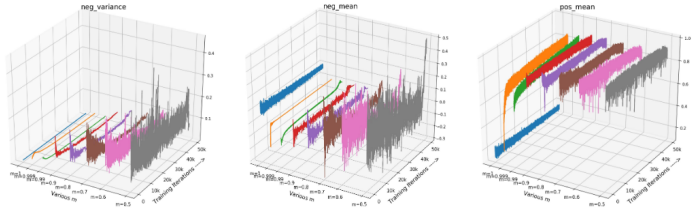
  - m (momentum coefficient) **affects** final transfer accuracy and model consistency.

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

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

Figure: From the authors. Linear readout protocol for evaluation on ImageNet-100.

Visualization Examples with MoCo

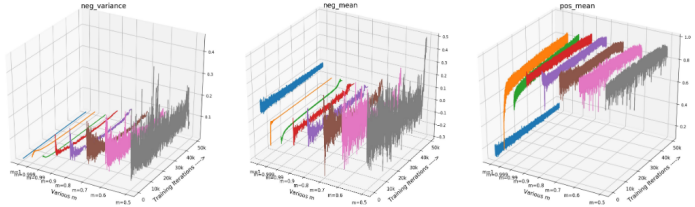


(a) Var of neg scores (b) Mean of neg scores (c) Mean of pos scores

Figure: From the authors. Pos/neg score statistics of various m in MoCo training.

- **Effect of Momentum** m:
  - Smaller m leads to higher variance in negative scores, causing **inconsistencies**.
  - Optimal $m = 0.99$ provides a **balance**, achieving the highest accuracy.

Visualization Examples with MoCo

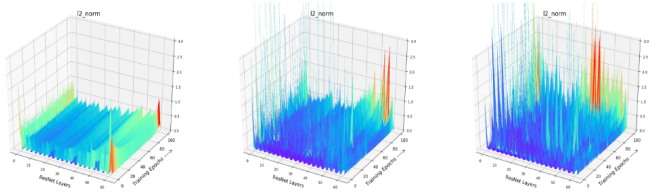


(a) Var of neg scores (b) Mean of neg scores (c) Mean of pos scores

Figure: From the authors. Pos/neg score statistics of various *m* in MoCo training.

- **Score Distribution**:
  - Lower positive scores (hard positives) indicate **larger view variance**, which enhances transfer performance.
  - Variance in negative scores should be moderate; too high variance disrupts **learning stability**.

Visualization Examples with MoCo

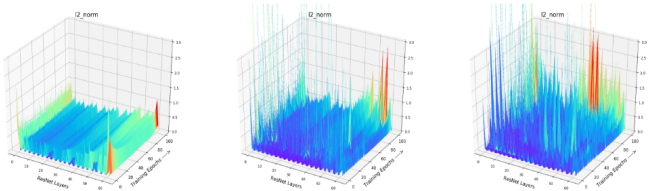


(a) $m = 0.99 \mid 56.2\%$ (b) $m = 0.6 \mid 21.2\%$ (c) $m = 0.5 \mid \text{collapse}$

Figure: From the authors. Gradient (l_2 norm) landscape of various m.

- **Gradient Landscape Analysis**:
  - Visualizing gradients helps understand the **stability of the learning process**.
  - **Smooth gradient landscapes** correlate with better performance and stable learning.

Visualization Examples with MoCo



(a) $m = 0.99 \mid 56.2\%$ (b) $m = 0.6 \mid 21.2\%$ (c) $m = 0.5 \mid \text{collapse}$

Figure: From the authors. Gradient (l_2 norm) landscape of various m.

- **Reasons for Model Collapse**:
  - Fast update speed of f_k (small m) causes **inconsistency** in the memory queue.
  - High volatility in negative scores leads to **unstable loss and gradients**.
  - Ensuring **stable score distribution and gradients** is crucial for effective contrastive learning.

Key Takeaways

- To learn a better pre-trained model, we need **correctly design** the negative pairs
  - **Stability and smoothness** of score distribution and gradient landscape
- To achieve stable and smooth score distribution and gradient, we can adapt some **feature transformation** methods.
  - Hard positives from **decreasing** easy positive scores.
  - Hard negatives from **diversifying** easy negative scores.

Overview

4. Proposed Feature Transformation Method

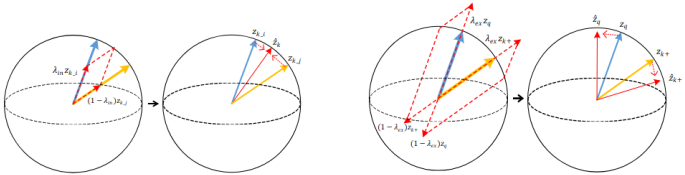
Proposed Feature Transformation Method

Learning Objective

- **Info-NCE Loss**: Aim to bring positive pairs (z_q and z_k^+) closer while pushing negative pairs (z_q and all z_k^- in memory queue) apart in the embedding space.
- **Feature Transformation**: Apply transformations on pos/neg features to provide **appropriate regularization** and make the **learning process harder**.

Proposed Feature Transformation Method

Process



(a) Negative Interpolation (b) Positive Extrapolation

Figure: From the authors. The process of the proposed negative interpolation and positive extrapolation. For the negative interpolation, they 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.

Positive Extrapolation

Concept

- **Observation**: Lowering positive pair scores creates **hard positives**, beneficial for transfer performance.
- **Goal**: Manipulate positive features (z_q and z_k^+) to increase **view variance** during training.

Positive Extrapolation

Formulation: transformation

$$\hat{z}_q = \lambda_{ex} z_q + (1 - \lambda_{ex}) z_{k^+}$$

$$\hat{z}_{k^+} = \lambda_{ex} z_{k^+} + (1 - \lambda_{ex}) z_q$$

Positive Extrapolation

Formulation: transformation

$$\hat{z}_q = \lambda_{ex} z_q + (1 - \lambda_{ex}) z_{k^+}$$
$$\hat{z}_{k^+} = \lambda_{ex} z_{k^+} + (1 - \lambda_{ex}) z_q$$

- Ensure that the summation of weights **equals to 1**.

Positive Extrapolation

Formulation: transformation

$$\hat{z}_q = \lambda_{ex} z_q + (1 - \lambda_{ex}) z_{k^+}$$

$$\hat{z}_{k^+} = \lambda_{ex} z_{k^+} + (1 - \lambda_{ex}) z_q$$

- Ensure that the summation of weights **equals to 1**.
- But we also need to make sure the transformed positive score $\hat{S}_{q \cdot k^+}$ is **smaller than the original positive score** $S_{q \cdot k^+}$...

Positive Extrapolation

Formulation: score transformation

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

Positive Extrapolation

Formulation: score transformation

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

- Set $\lambda_{ex} \geq 1$ to ensure $2 \cdot \lambda_{ex}(1 - \lambda_{ex}) \leq 0$.

Positive Extrapolation

Formulation: score transformation

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

- Set $\lambda_{ex} \geq 1$ to ensure $2 \cdot \lambda_{ex}(1 - \lambda_{ex}) \leq 0$.
- **Sample** $\lambda_{ex} \sim \text{Beta}(\alpha_{ex}, \alpha_{ex}) + 1$, resulting in a range of (1, 2).

Positive Extrapolation

Formulation: score transformation

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

- Set $\lambda_{ex} \geq 1$ to ensure $2 \cdot \lambda_{ex}(1 - \lambda_{ex}) \leq 0$.
- **Sample** $\lambda_{ex} \sim \text{Beta}(\alpha_{ex}, \alpha_{ex}) + 1$, resulting in a range of (1, 2).
- Transformed positive scores range:

$$\hat{S}_{q \cdot k^+} \in [-4 + 5S_{q \cdot k^+}, S_{q \cdot k^+}]$$

Positive Extrapolation

Formulation: score transformation

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

- Set $\lambda_{ex} \geq 1$ to ensure $2 \cdot \lambda_{ex}(1 - \lambda_{ex}) \leq 0$.
- **Sample** $\lambda_{ex} \sim \text{Beta}(\alpha_{ex}, \alpha_{ex}) + 1$, resulting in a range of (1, 2).
- Transformed positive scores range:

$$\hat{S}_{q \cdot k^+} \in [-4 + 5S_{q \cdot k^+}, S_{q \cdot k^+}]$$

- The transformation **increases the distance** between extrapolated feature vectors, creating **hard positives** from easy ones.

Positive Extrapolation

Experimental Results

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.

Figure: From the authors.

- **ImageNet-100**: Positive extrapolation **improves** accuracy from 71.1% to 72.8%.
- **Conclusion**: Positive extrapolation consistently demonstrates efficacy by **improving** accuracy over baseline MoCo.

Negative Interpolation

Concept

- **Problem**: Previous methods do not make **full use of negative** samples.

Negative Interpolation

Concept

- **Problem**: Previous methods do not make **full use of negative** samples.
- **Goal**: Fully utilize negative samples by creating **diversified negatives** for each training step.

Negative Interpolation

Formulation: transformation

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

- Z_{neg}: negative memory queue of MoCo - $\{z_1, z_2, \ldots, z_K\}$
- Z_{perm}: Random permutation of negative memory queue Z_{neg}.
- $\lambda_{in} \sim \text{Beta}(\alpha_{in}, \alpha_{in})$

Negative Interpolation

Formulation: transformation

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

- Z_{neg}: negative memory queue of MoCo - $\{z_1, z_2, \ldots, z_K\}$
- Z_{perm}: Random permutation of negative memory queue Z_{neg}.
- $\lambda_{in} \sim \text{Beta}(\alpha_{in}, \alpha_{in})$
- The transformation provides **fresh interpolated negatives**, enhancing **diversity** and making the model more **discriminative**.

negative interpolation

Experimental Results

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.

Figure: From the authors.

- **ImageNet-100**: Negative interpolation **improves** accuracy from 71.1% to 74.6%.
- **Conclusion**: Negative interpolation significantly **improves** accuracy, demonstrating effectiveness and robustness across various α_{in}.

Comparison with previous methods

- Previous works have explored **image-level** and **feature-level** mixing
- This work differs in three ways
  - Motivated by **empirical observations**
  - Positive extrapolation: **novel** and **outperforms** other methods
  - Negative interpolation: **fully utilizes the negative samples**

Discussions

- **Memory Queue vs. Feature Transformation**
  - Negative interpolation achieves similar improvements with less computational cost.
- **Timing of Feature Transformation**
  - Earlier introduction leads to greater accuracy improvements.
  - Provides stable and smooth training dynamics.
- **Dimension-level Mixing**
  - Introduces more diversity and view variance, improving performance.
- **Longer Training**
  - Gains from feature transformation diminish over time but accelerate convergence.

Discussion

Extending Memory Queue Instead of Feature Transformation

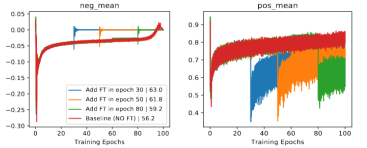
- **Previous Approaches**:
  - Increasing the number of negative examples in the memory queue (K) is beneficial for performance.
  - Methods like MoCo use memory queues or large batch sizes to acquire more negative examples.
  - Increasing K improves the lower bound of mutual information.
- **This Approach**:
  - Negative interpolation can also enlarge the number of negative examples.
  - **Experiment**: Use union queue of original negatives and interpolated negatives.
    - Combination shows negligible improvement over using interpolated queue alone.
    - **Conclusion**: Interpolated negatives provide sufficient diversity without the need for double negatives.
  - **Recommendation**: Feature transformations are more computationally efficient and effective than simply increasing the number of negative examples.

Discussion

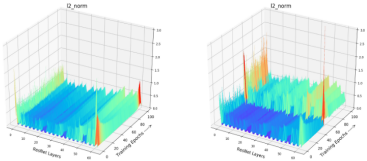
When to Add Feature Transformation

- **Experiment**:
  - Adding Feature Transformation (FT) at different training stages consistently boosts accuracy.
  - Earlier implementation of FT shows more significant improvement.
- **Analysis**:
  - FT brings hard positives (lowering positive scores) and hard negatives (rising negative scores).
  - Greater gradient observed with FT helps the model escape local minima and avoid overfitting.
  - **Conclusion**: FT is a plug-and-play method that enhances view-invariance and discrimination during contrastive model training.

When to Add Feature Transformation



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



(c) Baseline MoCo landscape (d) Adding FT in 50th epoch

Figure: From the authors. Visualization of when to add FT, including score distribution and Gradient (l_2 norm) landscape.

Discussion

Dimension-Level Mixing

- **Concept**:
  - Extend transformation to dimension-level mixing.
  - Perform transformations on each dimension of the feature vector.
- **Formulation**:

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

  - $\odot$ denotes the Hadamard product.
  - λ is a vector with the same dimension as the feature vector, sampled from a beta distribution.
- **Benefits**:
  - Introduces more diversity in negative interpolation.
  - Increases view variance in positive extrapolation.
  - **Results**: Dimension-level mixing shows improvement over feature-level mixing.

Discussion

Gains from Feature Transformation Over Longer Training

- **Observation**:
  - Simply training longer improves contrastive pre-train performance.
  - MoCoV2 with 500 epochs shows that FT maintains its benefits, but the improvement margin reduces with extended training.
- **Conclusion**:
  - Longer training naturally increases diversity by comparing more positive/negative pairs.
  - FT accelerates this process, leading to faster convergence.
  - **Recommendation**: FT is highly beneficial in the early stages of training, providing rapid improvements in diversity and discriminativeness.

Overview

5. Experiments

Experimental Results: ImageNet-100

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.

Figure: From the authors.

Experimental Results: ImageNet-1K

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[36]	68.6	-	-	-

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

Figure: From the authors.

Experimental Results: Downstream Tasks

pre-train	IN-1k Top-1	Faster [33] R50-C4 VOC AP	AP_{50}	AP_{75}	Mask R-CNN [15] R50-C4 COCO AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
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[51]	68.6	57.8	82.6	64.2	39.4	59.2	42.3	34.4	55.7	36.6
InsLoc[53]	-	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, 40]. The COCO results of mocoV2 is from [20]. Our results are reported using the average of 5 runs.

Figure: From the authors.

Overview

6. Conclusion

Conclusion

- **Novel Approach**: Visualization tool and feature transformations enhance contrastive learning.
- **Effective Methods**: Positive extrapolation and negative interpolation improve view-invariance and discriminativeness.
- **Robustness**: Demonstrated significant accuracy improvements and reduced task-bias.
- **Future Work**: Explore additional feature manipulation strategies using the visualization tool.

Overview

7. Potential Directions

Potential Directions

- Evaluate proposed feature transformations in the context of kinship recognition.
- Simplify kinship datasets to evaluate training dynamics.
- Propose new feature transformations.

The End