Mixed Images Enhance Supervised Contrastive Learning

Anonymous submission

Paper ID

Abstract

The ABSTRACT is to be in fully justified italicized text, at the top of the left-hand column, below the author and affiliation information. Use the word "Abstract" as the title, in 12-point Times, boldface type, centered relative to the column, initially capitalized. The abstract is to be in 10-point, single-spaced type. Leave two blank lines after the Abstract, then begin the main text. Look at previous abstracts to get a feel for style and length.

1. Introduction

In the field of machine learning, deep learning models have achieved remarkable success in various tasks, such as image classification, object detection, and natural language processing. However, these models often require a large amount of labeled data to train effectively, and they may suffer from overfitting when the data is limited. Data augmentation techniques have been widely used to address these issues by generating additional training data to increase the diversity and generalization ability of the models.

Mixup [refe] is a data augmentation method that linearly interpolates between pairs of examples and their labels. By doing so, mixup creates new synthetic data points that lie on the line segment between the original data points in the feature space. This technique has been shown to improve the generalization performance of deep learning models and reduce overfitting. Mixup and its variant approaches have been successfully utilized in a diverse range of tasks and training paradigms, such as Supervised Learning, Self-Supervised Learning, Semi-Supervised Learning, NLP, Graph, and Speech [refer].

On the other hand, Supervised Contrastive Learning (SCL) [refe] is a supervised learning approach that aims to learn more generalized and robust representations by maximizing the similarity between positive pairs and minimizing the similarity between negative pairs. This can lead to improved performance on various downstream tasks, such as classification, retrieval, and clustering.

Motivated by the complementary nature of mixup and SCL, we propose to combine these two techniques to further enhance the performance of deep learning models. Specifically, we explore how to integrate mixup into the SCL framework to leverage the advantages of both methods. By combining mixup with SCL, we expect to generate more diverse and discriminative feature representations, thereby imporving the model's generalization ability and performance on various tasks.

The rest of this paper is organized as follows. In Section 2, we review the related works on mixup and SCL. In Section 3, we present our proposed method for combining mixup and SCL. Section 4 describes the experimental setup and results, and Section 5 concludes the paper with a discussion of the findings and future directions.

2. Related Works

2.1. Mixup

Mixup is a technique used for data augmentation. By performing a simple linear transformation on the data, it can imporve the generalization ability and robustness of the model. Specifically, for two samples $(\mathbf{x}_i, \mathbf{y}_i)$ and $(\mathbf{x}_j, \mathbf{y}_j)$, Mixup generates a new sample $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$, where $\tilde{\mathbf{x}} = \lambda \mathbf{x}_i + (1-\lambda)\mathbf{x}_j, \tilde{\mathbf{y}} = \lambda \mathbf{y}_i + (1-\lambda)\mathbf{y}_j$. Here, λ is a value randomly sampled from the Beta distribution, and its value is usually between $[0, 1]$.

Mixup has been widely applied in various tasks and training paradigms, including Supervised Learining, Self-Supervised Learning, Semi-Supervised Learning, NLP, Graph, and Speech []. Meanwhile, Mixup has derived various different variants. According to the research of Jin *et al.*, Mixup and its variants can be roughly divided into two categories: (1) Sample Mixup Policies. This type of sample mixing strategy mainly focuses on how to generate new samples by mixing multiple original samples, such as Mixup, Cutmix, Manifold Mixup, ResizeMix, etc. (2) Label Mixup Policies. This type of mixing strategy focuses on how to generate corresponding labels for mixed samples, such as MixCo, TransMix, GraphMix, etc. In the subsequent experiments of this paper, only some widely used and easy-to-implement methods in Sample Mixup Policies are considered.

2.2. Contrastive Learning

Contrastive learning is an unsupervised learning method aimed at learning the feature representations of data to make similar data closer in the feature space and dissimilar data farther apart. Its core idea is to construct positive and negative sample pairs and optimize the model through a contrastive loss function, enabling the model to learn discriminative feature representations. There are many popular contrastive learning models, such as SimCLR, MoCo, BYOL, SwAV, etc. Taking SimCLR as an example, its main steps are:

(1) Constructing positive and negative sample pairs: Performing two independent and random data augmentations, such as cropping, color distortion, and flipping, on the original images $\{\mathbf{x}_i, i = 1, ..., N\}$ to obtain two different views: $\{\tilde{\mathbf{x}}_i, i = 1, ..., N\}$, $\{\hat{\mathbf{x}}_i, i = 1, ..., N\}$. The sample pairs $\{(\tilde{\mathbf{x}}_i, \hat{\mathbf{x}}_i), i = 1, ..., N\}$ are regarded as positive sample pairs, while the sample pairs $\{(\tilde{\mathbf{x}}_i, \hat{\mathbf{x}}_j), i \neq j\}$, $\{(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j), i \neq j\}$, and $\{(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j), i \neq j\}$ are regarded as negative sample pairs.

(2) Feature extraction: Using an encoder f (usually a neural network) to map the views to the feature space to obtain feature representations $\tilde{\mathbf{h}}_i = f(\tilde{\mathbf{x}}_i)$, $\hat{\mathbf{h}}_i = f(\hat{\mathbf{x}}_i)$.

(3) Adding a projection head: The feature representations $\{\tilde{\mathbf{h}}_i, i = 1, ..., N\}$ and $\{\hat{\mathbf{h}}_i, i = 1, ..., N\}$ are further processed by a small multi-layer perceptron (projection head) g to improve the performance and generalization ability of the model, obtaining $\tilde{\mathbf{z}}_i = g(\tilde{\mathbf{h}}_i)$ and $\hat{\mathbf{z}}_i = g(\hat{\mathbf{h}}_i)$.

(4) Calculating the loss function: SimCLR uses InfoNCE as the loss function, and its specific form is:

$$\mathcal{L} = -\frac{1}{N}\sum_{k=1}^{N} log\left(\frac{exp(\tilde{\mathbf{z}}_k^T \cdot \hat{\mathbf{z}}_k/\tau)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\mathbb{I}_{[i\neq j]}NegDis}\right) \quad (1)$$

where $NegDis = exp(\tilde{\mathbf{z}}_i^T \cdot \tilde{\mathbf{z}}_j/\tau) + exp(\hat{\mathbf{z}}_i^T \cdot \hat{\mathbf{z}}_j/\tau)$; τ is a temperature parameter used to adjust the intensity of the contrast; $\mathbb{I}_{[i\neq j]}$ is an indicator function, and if $i \neq j$, then $\mathbb{I}_{[i\neq j]} = 1$, otherwise $\mathbb{I}_{[i\neq j]} = 1$.

By continuously reducing the loss function to update the parameters of the neural network, the neural network can learn visual feature representations with good generalization ability and has achieved significant performance improvements in various downstream tasks.

2.3. Combining Mixup with Self-Supervise Learning

According to the research of Jin *et al.*, the combination of Mixup and Self-Supervised learning is mainly reflected in the following aspects:

(1) Contrastive Learning: The application of the Mixup method in Contrastive Learning is to enhance the feature extraction ability of the model by generating ”semi-positive samples”. These ”semi-positive samples” are obtained through the Mixup technique and help the model learn more abundant feature representations. For example, the MixCo method proposes to combine mixed data with consistency regularization, enhance the generalization ability of the model by mixing multiple samples and constraining the model’s prediction consistency for the mixed samples. Whereas, i-Mix assigns a unique virtual class to each data instance and performs data mixing in the input and virtual label space to strengthen the training data.

(2) Masked Image Modeling (MIM): In MIM, the Mixup technique is used to generate mixed samples. These samples are mapped to the semantic space through a encoder and then the pixels are reconstructed in the original space through a decoder. MixMAE and MixedAE are two examples of this method, which achieve mixing by replacing the masks in MAE with patches of other images, thereby improving the feature extraction ability of the model.

(3) Semi-Supervised Learning (Semi-SL): In SSL, the Mixup technique can combine labeled and unlabeled data to improve the model’s performance by synthesizing mixed samples. For instance, the MixMatch method improves the robustness of the model by mixing labeled and unlabeled data.

Mixup has been proven to enhance the effectiveness of self-supervised learning. Now, we intend to extend it to the field of supervised contrastive learning.

2.4. Supervised Contrastive Learning

The most significant distinction between Supervised Contrastive Learning and Contrastive Learning is that the former effectively utilizes the label information of the data. In Supervised Contrastive Learning, positive sample pairs are constructed from different samples within the same category, while negative sample pairs are composed of samples between different categories. Given the data samples $\{\mathbf{x}_i, i = 1, ..., N\}$ and their corresponding labels $\{\mathbf{y}_i, i = 1, ..., N\}$, let $A(\mathbf{x}_i) = \{j : \mathbf{y}_j = \mathbf{y}_i\}$, then the contrastive loss is:

$$l_i = \sum_{j\in A(\mathbf{x}_i)} log\left(\frac{exp\left(\mathbf{x}_i^T \cdot \mathbf{x}_j/\tau\right)}{\sum_{k=1}^{N}\sum_{l=1}^{N}\mathbb{I}_{[k\neq l]}exp\left(\mathbf{x}_k^T \cdot \mathbf{x}_l/\tau\right)}\right) \quad (2)$$

$$\mathcal{L} = -\frac{1}{\sum_{i=1}^{N}|A(\mathbf{x}_i)|}\sum_{i=1}^{N} l_i \quad (3)$$

where $A(\mathbf{x}_i)$ represents the indices of the samples that belong to the same category as $\mathbf{x}_i$; $|A(\mathbf{x}_i)|$ is the number of indices in $A(\mathbf{x}_i)$. Additionally, in practical applications, just as in Contrastive Learning, we perform two independent and random data augmentations on the samples to obtain two distinct views, which ensures that each sample has at least one positive sample pair.

Compare to Contrastive Learning, Supervised Contrastive Learning can more effectively learn task-related feature representations, thereby achieving better performance in classification tasks, improving the classification accuracy, and helping the model form clear category boundaries in the feature space, thereby enhancing the model's generalization ability.

3. Method

In this section, we will explain how our method—Mixed Supervised Contrastive Learning (MixSupCon)—combines Mixup and Supervised Contrastive Learning. Our motivation is similar to that of MixCo: we use Mixup to soften the data and labels, prompting the model to learn the intrinsic relationships between "semi-positive samples" and "semi-negative samples", thereby alleviating the problem of instance discrimination.

By applying Mixup, which involves linearly interpolating between pairs of images and their corresponding labels, we create mixed samples. These mixed images are then input into the neural network, allowing it to determine the degree of similarity and dissimilarity between the images. Learning to capture such relationships is much more challenging than simply distinguishing positive and negative samples, but it significantly enhances the model's feature extraction capabilities. This approach enables the model to better understand the subtle differences in the data, leading to improved generalization and robustness.

Given the classification samples $\{(\mathbf{x}_i, \mathbf{y}_i), i = 1, ..., N\}$, where $\mathbf{x}_i$ represents the i-th image, and $\mathbf{y}_i$ is the label encoded in One-Hot form corresponding to the i-th image.

Similar to Contrastive Learning, first, two independent and random data augmentation operations are performed on the samples to obtain two different views, namely $\{(\tilde{\mathbf{x}}_i, \mathbf{y}_i), i = 1, ..., N\}$ and $\{(\hat{\mathbf{x}}_i, \mathbf{y}_i), i = 1, ..., N\}$. Then, two samples are randomly selected from these two views respectively, and the mixup method is used to generate the mixed sample data $\{(\bar{\mathbf{x}}_k, \bar{\mathbf{y}}_k), k = 1, ..., N_{mix}\}$. Specifically, $\bar{x}_k = \lambda_k \tilde{\mathbf{x}}_i + (1 - \lambda_k)\hat{\mathbf{x}}_j$, $\bar{\mathbf{y}}_k = \lambda_k \tilde{\mathbf{y}}_i + (1 - \lambda_k)\hat{\mathbf{y}}_j$, where λ_k are samples drawn from the Beta distribution.

After that, the mixed data is input into the neural network (encoder) to obtain the representation: $\mathbf{v}_k = f_{encoder}(\bar{\mathbf{x}}_k)$. The these representations are input into the projection head $g : \mathbf{z}_k = g(\mathbf{v}_k)$.

Next, the contrastive loss can be calculated:

$$\mathcal{L} = -\frac{1}{\sum_{k=1}^{N_{mix}} \Phi(\mathbf{y}_k)} \sum_{k=1}^{N_{mix}} \Phi(\mathbf{y}_k)\Psi(\mathbf{z}_k) \tag{4}$$

where

$$\Psi(\mathbf{z}_k) = log\left(\frac{exp(\mathbf{z}_k^T \cdot \mathbf{z}_k/\tau)}{\sum_{i=1}^{N_{mix}} \sum_{j=1}^{N_{mix}} \mathbb{I}_{[i\neq j]} exp(\mathbf{z}_k^T \cdot \mathbf{z}_k/\tau)}\right) \tag{5}$$

Hyperparameter	Value
Batch Size	256
Epochs	1000
Learning Rate	0.05
Weight Decay	0.0001
Learning Rate Decay Epochs	700, 800, 900
Learning Rate Decay Rate	0.1

Table 1. The hyperparameters of pretraining tasks.

and

$$\Phi(\mathbf{y}_k) = \mathbf{y}_k^T \cdot \mathbf{y}_k \tag{6}$$

Here, $\Phi(\mathbf{y}_k)$ is utilized to calculate the similarity between the mixed samples, $\Psi(\mathbf{z}_k)$ is employed to calculate the similarity between representations.

4. Experiments

4.1. Experimental Setup

4.1.1. Datasets

This study employs two recognized benchmark datasets for image classification tasks: CIFAR-10 and CIFAR-100. The CIFAR-10 dataset comprises 60,000 color images in 10 different classes, with a split of 50,000 training images and 10,000 test images. The CIFAR-100 dataset is an extension of CIFAR-10, consisting of 60,000 images across 100 classes, divided into 50,000 training and 10,000 test images. Each images in both datasets is of size 32×32 pixels.

4.1.2. Model Architectures

We have selected three ResNet models (ResNet-18, ResNet-34, ResNet-50) for this investigation. Their inclusion is predicated on their varying depths and performance profiles, which will allow us to conduct a thorough analysis of the influence of model architecture on experimental results.

4.1.3. Pretraining Tasks

The pretraining tasks selected for this study are SimCLR, SupCon, and MixSupCon. The hyperparameters for these tasks are standardized as 1. After pretraining, the models undergo fine-tuning on the datasets with a linear classifier (a fully connected layer), while the pretrained model parameters remain unchanged. The fine-tuning hyperparameters are detailed as 2.

4.1.4. Control Group: Non-Pretrained Classification Task

A control group is established to assess the performance of classification tasks without the benefit of pretraining. The hyperparameters for this control group are as 3.

Hyperparameter	Value
Batch Size	256
Epochs	100
Learning Rate	0.1
Weight Decay	0
Learing Rate Decay Epochs	60,75,90
Learning Rate Decay Rate	0.2

Table 2. The hyperparameters of fine-tuning tasks.

Hyperparameter	Value
Batch Size	256
Epochs	500
Learning Rate	0.2
Weight Decay	0.0001
Learing Rate Decay Epochs	350,400,450
Learning Rate Decay Rate	0.1

Table 3. The hyperparameters of non-pretrained classification task.

4.1.5. Experimental Phases

The experimental process is divided into four distinct phases:

1. Linear Evaluation: In this phase, models are pretrained on the datasets and subsequently fine-tuned with a linear classifier to assess the quality of the learned representations.

2. Visualization of Representations: Following pretraining, we employ dimensionality reduction techniques (t-SNE) to visualize the learned representations, providing insights into the clustering and separability of the features.

3. Ablation Study: This phase delves into the effects of various data mixing strategies and beta distribution parameters on model accuracy, systematically analyzing their impact on overall performance.

4. Comparison with SOTA: The final phase benchmarks our methods against state-of-the-art (SOTA) approaches, establishing the competitiveness and efficacy of our strategies.

4.1.6. Software and Hardware

All experiments are conducted using Python 3.10 in conjunction with the PyTorch library, version 2.0.1. The computational backbone is provided by an NVIDIA GeForce RTX 4080 GPU, ensuring the efficient execution of deep learning tasks.

4.2. Linear Evaluation

4.3. Visualization of Representations

4.4. Ablation

5. Conclusion

References