Towards Domain-Agnostic Contrastive Learning

Vikas Verma [1 2] **Minh-Thang Luong** [1] **Kenji Kawaguchi** [3] **Hieu Pham** [1] **Quoc V. Le** [1]

Abstract

Limitations of existing contrastive based SSL

Despite recent successes, most contrastive self-supervised learning methods are domain-specific, relying heavily on data augmentation techniques that require knowledge about a particular domain, such as image cropping and rotation. To overcome such limitation, we propose a domain-agnostic approach to contrastive learning, named *DACL*, that is applicable to problems where domain-specific data augmentations are not readily available. Key to our approach is the use of *Mixup noise* to create similar and dissimilar examples by mixing data samples differently either at the input or hidden-state levels. We theoretically analyze our method and show advantages over the Gaussian-noise based contrastive learning approach. To demonstrate the effectiveness of DACL, we conduct experiments across various domains such as tabular data, images, and graphs. Our results show that DACL not only outperforms other domain-agnostic noising methods, such as Gaussian-noise, but also combines well with domain-specific methods, such as SimCLR, to improve self-supervised visual representation learning.

Proposed solution

1. Introduction

One of the core objectives of deep learning is to discover useful representations from the raw input signals without explicit labels provided by human annotators. Recently, self-supervised learning methods have emerged as one of the most promising classes of methods to accomplish this objective with strong performances across various domains such as computer vision (Oord et al., 2018; He et al., 2020; Chen et al., 2020b; Grill et al., 2020), natural language processing (Dai & Le, 2015; Howard & Ruder, 2018; Peters et al., 2018; Radford et al., 2019; Clark et al., 2020), and speech recognition (Schneider et al., 2019; Baevski et al., 2020). These self-supervised methods learn useful representations without explicit annotations by reformulating the unsupervised representation learning problem into a supervised learning problem. This reformulation is done by defining a pretext task. The pretext tasks defined in these methods are based on certain domain-specific *regularities* and would generally differ from domain to domain (more discussion about this is in the related work, Section 6).

How we went from unsupervised to self-supervised?

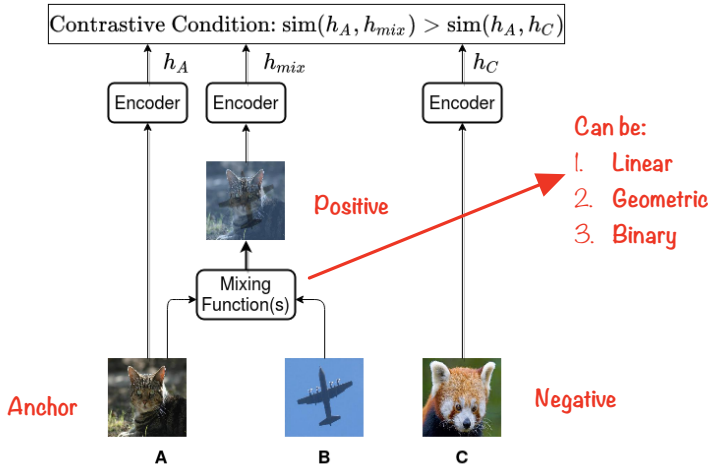

Figure 1. For a given sample A, we create a positive sample by mixing it with another random sample B. The mixing function can be either of the form of Equation 3 (Linear-Mixup), 5 (Geometric-Mixup) or 6 (Binary-Mixup), and the mixing coefficient is chosen in such a way that the mixed sample is closer to A than B. Using another randomly chosen sample C, the contrastive learning formulation tries to satisfy the condition $\text{sim}(\boldsymbol{h_A}, \boldsymbol{h_{mix}}) > \text{sim}(\boldsymbol{h_A}, \boldsymbol{h_C})$, where sim is a measure of similarity between two vectors.

Among various pretext tasks defined for self-supervised learning, contrastive learning, e.g. (Chopra et al., 2005; Hadsell et al., 2006; Oord et al., 2018; Hénaff et al., 2019; He et al., 2020; Chen et al., 2020b; Tian et al., 2020; Cai et al., 2020; Wang & Isola, 2020), is perhaps the most popular approach that learns to distinguish semantically similar examples over dissimilar ones. Despite its general applicability, contrastive learning requires a way, often by means of data augmentations, to create semantically similar and dissimilar examples in the domain of interest for it to work. For

What's one thing that isn't nice about Contrastive learning?

[*]Equal contribution [1]Google Research, Brain Team. [2]Aalto University, Finland. [3]Harvard University. Correspondence to: Vikas Verma <vikas.verma@aalto.fi>, Minh-Thang Luong <thangluong@google.com>, Kenji Kawaguchi <kkawaguchi@fas.harvard.edu>, Hieu Pham <hyhieu@google.com>, Quoc V. Le <qvl@google.com>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

When it comes to data augmentation, IMHO, tabular data is the hardest one. Requires much more creativity compared to anything else.

example, in computer vision, semantically similar samples can be constructed using semantic-preserving augmentation techniques such as flipping, rotating, jittering, and cropping. These semantic-preserving augmentations, however, require domain-specific knowledge and may not be readily available for other modalities such as graph or tabular data.

Why Gaussian noise isn't enough!

How to create semantically similar and dissimilar samples for new domains remains an open problem. As a simplest solution, one may add a sufficiently small random noise (such as Gaussian-noise) to a given sample to construct examples that are similar to it. Although simple, such augmentation strategies do not exploit the underlying structure of the data manifold. In this work, we propose DACL, which stands for Domain-Agnostic Contrastive Learning, an approach that utilizes *Mixup-noise* to create similar and dissimilar examples by mixing data samples differently either at the input or hidden-state levels. A simple diagrammatic depiction of how to apply DACL in the input space is given in Figure 1. Our experiments demonstrate the effectiveness of DACL across various domains, ranging from tabular data, to images and graphs; whereas, our theoretical analysis sheds light on why Mixup-noise works better than Gaussian-noise.

In summary, the contributions of this work are as follows:

- We propose Mixup-noise as a way of constructing positive and negative samples for contrastive learning and conduct theoretical analysis to show that Mixup-noise has better generalization bounds than Gaussian-noise.
- We show that using other forms of data-dependent noise (geometric-mixup, binary-mixup) can further improve the performance of DACL.
- We extend DACL to domains where data has a non-fixed topology (for example, graphs) by applying Mixup-noise in the hidden states.
- We demonstrate that Mixup-noise based data augmentation is complementary to other image-specific augmentations for contrastive learning, resulting in improvements over SimCLR baseline for CIFAR10, CIFAR100 and ImageNet datasets.

2. Contrastive Learning : Problem Definition

Contrastive learning can be formally defined using the notions of "anchor", "positive" and "negative" samples. Here, positive and negative samples refer to samples that are semantically similar and dissimilar to anchor samples. Suppose we have an encoding function $h : \boldsymbol{x} \mapsto \boldsymbol{h}$, an anchor sample $\boldsymbol{x}$ and its corresponding positive and negative samples, $\boldsymbol{x}^+$ and $\boldsymbol{x}^-$. The objective of contrastive learning is to bring the anchor and the positive sample closer in the embedding space than the anchor and the negative sample.

Formally, contrastive learning seeks to satisfy the following condition, where sim is a measure of similarity between two vectors:

$$\text{sim}(\boldsymbol{h}, \boldsymbol{h}^+) > \text{sim}(\boldsymbol{h}, \boldsymbol{h}^-) \tag{1}$$

Ways to formulate equation !

While the above objective can be reformulated in various ways, including max-margin contrastive loss in (Hadsell et al., 2006), triplet loss in (Weinberger & Saul, 2009), and maximizing a metric of local aggregation (Zhuang et al., 2019), in this work we consider InfoNCE loss because of its adaptation in multiple current state-of-the-art methods (Sohn, 2016; Oord et al., 2018; He et al., 2020; Chen et al., 2020b; Wu et al., 2018). Let us suppose that $\{\boldsymbol{x}_k\}_{k=1}^N$ is a set of N samples such that it consists of a sample $\boldsymbol{x}_i$ which is semantically similar to $\boldsymbol{x}_j$ and dissimilar to all the other samples in the set. Then the InfoNCE tries to maximize the similarity between the positive pair and minimize the similarity between the negative pairs, and is defined as:

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\boldsymbol{h}_i, \boldsymbol{h}_j))}{\sum_{k=1}^N 1_{[k \neq i]} \exp(\text{sim}(\boldsymbol{h}_i, \boldsymbol{h}_k))} \tag{2}$$

3. Domain-Agnostic Contrastive Learning with Mixup

IMHO this won't work in tabular data if you aren't careful enough

For domains where natural data augmentation methods are not available, we propose to apply Mixup (Zhang et al., 2018) based data interpolation for creating positive and negative samples. Given a data distribution $\mathcal{D} = \{\boldsymbol{x}_k\}_{k=1}^K$, a positive sample for an anchor $\boldsymbol{x}$ is created by taking its random interpolation with another randomly chosen sample $\tilde{\boldsymbol{x}}$ from $\mathcal{D}$:

$$\boldsymbol{x}^+ = \lambda \boldsymbol{x} + (1 - \lambda)\tilde{\boldsymbol{x}} \tag{3}$$

where λ is a coefficient sampled from a random distribution such that $\boldsymbol{x}^+$ is closer to $\boldsymbol{x}$ than $\tilde{\boldsymbol{x}}$. For instance, we can sample λ from a uniform distribution $\lambda \sim U(\alpha, 1.0)$ with high values of α such as 0.9. Similar to SimCLR (Chen et al., 2020b), positive samples corresponding to other anchor samples in the training batch are used as the negative samples for $\boldsymbol{x}$.

Creating positive samples using Mixup in the input space (Eq. 3) is not feasible in domains where data has a non-fixed topology, such as sequences, trees, and graphs. For such domains, we create positive samples by mixing fixed-length hidden representations of samples (Verma et al., 2019a). Formally, let us assume that there exists an encoder function $h : \mathcal{I} \mapsto \boldsymbol{h}$ that maps a sample $\mathcal{I}$ from such domains to a representation $\boldsymbol{h}$ via an intermediate layer that has a fixed-length hidden representation $\boldsymbol{v}$, then we create positive sample in the intermediate layer as:

$$\boldsymbol{v}^+ = \lambda \boldsymbol{v} + (1 - \lambda)\tilde{\boldsymbol{v}} \tag{4}$$

Here v is the output of an intermediate hidden layer

The above Mixup based method for constructing positive samples can be interpreted as adding noise to a given sample

in the direction of another sample in the data distribution. We term this as Mixup-noise. One might ask how Mixup-noise is a better choice for contrastive learning than other forms of noise? The central hypothesis of our method is that a network is forced to learn better features if the noise captures the structure of the data manifold rather than being independent of it. Consider an image $\boldsymbol{x}$ and adding Gaussian-noise to it for constructing the positive sample: $\boldsymbol{x}^+ = \boldsymbol{x} + \boldsymbol{\delta}$, where $\boldsymbol{\delta} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}^2\boldsymbol{I})$. In this case, to maximize the similarity between $\boldsymbol{x}$ and $\boldsymbol{x}^+$, the network can learn just to take an average over the neighboring pixels to remove the noise, thus bypassing learning the semantic concepts in the image. Such kind of trivial feature transformation is not possible with Mixup-noise, and hence it enforces the network to learn better features. In addition to the aforementioned hypothesis, in Section 4, we formally conduct a theoretical analysis to understand the effect of using Gaussian-noise vs Mixup-noise in the contrastive learning framework.

Valid question

Awesome example!

For experiments, we closely follow the encoder and projection-head architecture, and the process for computing the "normalized and temperature-scaled InfoNCE loss" from SimCLR (Chen et al., 2020b). Our approach for Mixup-noise based Domain-Agnostic Contrastive Learning (DACL) in the input space is summarized in Algorithm 1. Algorithm for DACL in hidden representations can be easily derived from Algorithm 1 by applying mixing in Line 8 and 14 instead of line 7 and 13.

Algorithm 1 Mixup-noise Domain-Agnostic Contrastive Learning.

1: **input:** batch size N, temperature τ, encoder function h, projection-head g, hyperparameter α.
2: **for** sampled minibatch $\{\boldsymbol{x}_k\}_{k=1}^N$ **do**
3: **for all** $k \in \{1, \ldots, N\}$ **do**
4: # Create first positive sample using Mixup Noise
5: $\lambda_1 \sim U(\alpha, 1.0)$ # sample mixing coefficient
6: $\boldsymbol{x} \sim \{\boldsymbol{x}_k\}_{k=1}^N - \{\boldsymbol{x}_k\}$
7: $\tilde{\boldsymbol{x}}_{2k-1} = \lambda_1 \boldsymbol{x}_k + (1-\lambda_1)\boldsymbol{x}$
8: $\boldsymbol{h}_{2k-1} = h(\tilde{\boldsymbol{x}}_{2k-1})$ # apply encoder
9: $\boldsymbol{z}_{2k-1} = g(\boldsymbol{h}_{2k-1})$ # apply projection-head
10: # Create second positive sample using Mixup Noise
11: $\lambda_2 \sim U(\alpha, 1.0)$ # sample mixing coefficient
12: $\boldsymbol{x} \sim \{\boldsymbol{x}_k\}_{k=1}^N - \{\boldsymbol{x}_k\}$
13: $\tilde{\boldsymbol{x}}_{2k-1} = \lambda_2 \boldsymbol{x}_k + (1-\lambda_2)\boldsymbol{x}$
14: $\boldsymbol{h}_{2k} = h(\tilde{\boldsymbol{x}}_{2k})$ # apply encoder
15: $\boldsymbol{z}_{2k} = g(\boldsymbol{h}_{2k})$ # apply projection-head
16: **end for**
17: **for all** $i \in \{1, \ldots, 2N\}$ and $j \in \{1, \ldots, 2N\}$ **do**
18: $s_{i,j} = \boldsymbol{z}_i^\top \boldsymbol{z}_j / (\|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|)$ # pairwise similarity
19: **end for**
20: **define** $\ell(i,j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} 1_{[k \neq i]} \exp(s_{i,k}/\tau)}$
21: $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$
22: update networks h and g to minimize $\mathcal{L}$
23: **end for**
24: **return** encoder function $h(\cdot)$, and projection-head $g(\cdot)$

3.1. Additional Forms of Mixup-Based Noise

We have thus far proposed the contrastive learning method using the linear-interpolation Mixup. Other forms of Mixup-noise can also be used to obtain more diverse samples for contrastive learning. In particular, we explore "Geometric-Mixup" and "Binary-Mixup" based noise. In Geometric-Mixup, we create a positive sample corresponding to a sample $\boldsymbol{x}$ by taking its weighted-geometric mean with another randomly chosen sample $\tilde{\boldsymbol{x}}$:

$$\boldsymbol{x}^+ = \boldsymbol{x}^{\lambda} \odot \tilde{\boldsymbol{x}}^{(1-\lambda)} \tag{5}$$

Similar to Linear-Mixup in Eq.3 , λ is sampled from a uniform distribution $\lambda \sim U(\beta, 1.0)$ with high values of β.

In Binary-Mixup (Beckham et al., 2019), the elements of $\boldsymbol{x}$ are swapped with the elements of another randomly chosen sample $\tilde{\boldsymbol{x}}$. This is implemented by sampling a binary mask $\mathbf{m} \in \{0,1\}^k$ (where k denotes the number of input features) and performing the following operation:

$$\boldsymbol{x}^+ = \boldsymbol{x} \odot \mathbf{m} + \tilde{\boldsymbol{x}} \odot (1 - \mathbf{m}) \tag{6}$$

where elements of $\mathbf{m}$ are sampled from a Bernoulli(ρ) distribution with high ρ parameter.

We extend the DACL procedure with the aforementioned additional Mixup-noise functions as follows. For a given sample $\boldsymbol{x}$, we randomly select a noise function from Linear-Mixup, Geometric-Mixup, and Binary-Mixup, and apply this function to create both of the positive samples corresponding to $\boldsymbol{x}$ (line 7 and 13 in Algorithm 1). The rest of the details are the same as Algorithm 1. We refer to this procedure as DACL+ in the following experiments.

Apply every type of mixup chosen randomly at each iteration!

4. Theoretical Analysis

In this section, we mathematically analyze and compare the properties of Mixup-noise and Gaussian-noise based contrastive learning for a binary classification task. We first prove that for both Mixup-noise and Gaussian-noise, optimizing hidden layers with a contrastive loss is related to minimizing classification loss with the last layer being optimized using labeled data. We then prove that the proposed method with Mixup-noise induces a different regularization effect on the classification loss when compared with that of Gaussian-noise. The difference in regularization effects shows the advantage of Mixup-noise over Gaussian-noise when the data manifold lies in a low dimensional subspace.

1. Why optimising for contrastive loss is equivalent to optimising classification loss in the last layer using labelled data?
2. Why Mixup induces a better regularisation than simple Gaussian noise?

Intuitively, our theoretical results show that contrastive learning with Mixup-noise has implicit data-adaptive regularization effects that promote generalization.

To compare the cases of Mixup-noise and Gaussian-noise, we focus on linear-interpolation based Mixup-noise and unify the two cases using the following observation. For Mixup-noise, we can write $\boldsymbol{x}^+_{\text{mix}} = \lambda\boldsymbol{x}+(1-\lambda)\tilde{\boldsymbol{x}} = \boldsymbol{x}+\alpha\delta(\boldsymbol{x},\tilde{\boldsymbol{x}})$ with $\alpha = 1-\lambda > 0$ and $\delta(\boldsymbol{x},\tilde{\boldsymbol{x}}) = (\tilde{\boldsymbol{x}} - \boldsymbol{x})$ where $\tilde{\boldsymbol{x}}$ is drawn from some (empirical) input data distribution. For Gaussian-noise, we can write $\boldsymbol{x}^+_{\text{gauss}} = \boldsymbol{x} + \alpha\delta(\boldsymbol{x},\tilde{\boldsymbol{x}})$ with $\alpha > 0$ and $\delta(\boldsymbol{x},\tilde{\boldsymbol{x}}) = \tilde{\boldsymbol{x}}$ where $\tilde{\boldsymbol{x}}$ is drawn from some Gaussian distribution. Accordingly, for each input $\boldsymbol{x}$, we can write the positive example pair $(\boldsymbol{x}^+, \boldsymbol{x}^{++})$ and the negative example $\boldsymbol{x}^-$ for both cases as: $\boldsymbol{x}^+ = \boldsymbol{x} + \alpha\delta(\boldsymbol{x},\tilde{\boldsymbol{x}})$, $\boldsymbol{x}^{++} = \boldsymbol{x} + \alpha'\delta(\boldsymbol{x},\tilde{\boldsymbol{x}}')$, and $\boldsymbol{x}^- = \bar{\boldsymbol{x}} + \alpha''\delta(\bar{\boldsymbol{x}},\tilde{\boldsymbol{x}}'')$, where $\bar{\boldsymbol{x}}$ is another input sample. Using this unified notation, we theoretically analyze our method with the standard contrastive loss ℓ_{ctr} defined by $\ell_{\text{ctr}}(\boldsymbol{x}^+, \boldsymbol{x}^{++}, \boldsymbol{x}^-) = -\log \frac{\exp(\text{sim}[h(\boldsymbol{x}^+),h(\boldsymbol{x}^{++})])}{\exp(\text{sim}[h(\boldsymbol{x}^+),h(\boldsymbol{x}^{++})])+\exp(\text{sim}[h(\boldsymbol{x}^+),h(\boldsymbol{x}^-)])}$, where $h(\boldsymbol{x}) \in \mathbb{R}^d$ is the output of the last hidden layer and $\text{sim}[q, q'] = \frac{q^\top q'}{\|q\|\|q'\|}$ for any given vectors q and q'. This contrastive loss ℓ_{ctr} without the projection-head g is commonly used in practice and captures the essence of contrastive learning. Theoretical analyses of the benefit of the projection-head g and other forms of Mixup-noise are left to future work.

Usual Definition and formula for CL

This section focuses on binary classification with $y \in \{0, 1\}$ using the standard binary cross-entropy loss: $\ell_{\text{cf}}(q, y) = -y\log(\hat{p}_q(y = 1)) - (1 - y)\log(\hat{p}_q(y = 0))$ with $\hat{p}_q(y = 0) = 1 - \hat{p}_q(y = 1)$ where $\hat{p}_q(y = 1) = \frac{1}{1+\exp(-q)}$. We use $f(\boldsymbol{x}) = h(\boldsymbol{x})^\top w$ to represent the output of the classifier for some w; i.e., $\ell_{\text{cf}}(f(\boldsymbol{x}), y)$ is the cross-entropy loss of the classifier f on the sample $(\boldsymbol{x}, y)$. Let $\phi : \mathbb{R} \to [0, 1]$ be any Lipschitz function with constant L_ϕ such that $\phi(q) \geq 1_{[q\leq 0]}$ for all $q \in \mathbb{R}$; i.e., ϕ is an smoothed version of 0-1 loss. For example, we can set ϕ to be the hinge loss. Let $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y}$ be the input and output spaces as $\boldsymbol{x} \in \mathcal{X}$ and $y \in \mathcal{Y}$. Let $c_{\boldsymbol{x}}$ be a real number such that $c_{\boldsymbol{x}} \geq (\boldsymbol{x}_k)^2$ for all $\boldsymbol{x} \in \mathcal{X}$ and $k \in \{1, \ldots, d\}$.

Urgghh... simple made complex

As we aim to compare the cases of Mixup-noise and Gaussian-noise accurately (without taking loose bounds), we first prove an exact relationship between the contrastive loss and classification loss. That is, the following theorem shows that optimizing hidden layers with contrastive loss $\ell_{\text{ctr}}(\boldsymbol{x}^+, \boldsymbol{x}^{++}, \boldsymbol{x}^-)$ is related to minimising classification loss $\ell_{\text{cf}}(f(\boldsymbol{x}^+), y)$ with the error term $\mathbb{E}_y[(1 - \bar{\rho}(y))\mathcal{E}_y]$, where the error term increases as the probability of the negative example x^- having the same label as that of the positive example x^+ increases:

Theorem 1. *Let $\mathcal{D}$ be a probability distribution over $(\boldsymbol{x}, y)$ as $(\boldsymbol{x}, y) \sim \mathcal{D}$, with the corresponding marginal distribution $\mathcal{D}_x$ of $\boldsymbol{x}$ and conditional distribution $\mathcal{D}_y$ of $\boldsymbol{x}$ given a y. Let $\bar{\rho}(y) = \mathbb{E}_{(\boldsymbol{x}',y')\sim\mathcal{D}}[1_{[y'\neq y]}]$ $(= \Pr(y' \neq y \mid y) > 0)$. Then, for any distribution pair $(\mathcal{D}_{\tilde{\boldsymbol{x}}}, \mathcal{D}_\alpha)$ and function δ, the following holds:*

$$\mathbb{E}_{\substack{\boldsymbol{x},\bar{\boldsymbol{x}}\sim\mathcal{D}_x,\\ \tilde{\boldsymbol{x}},\tilde{\boldsymbol{x}}',\tilde{\boldsymbol{x}}''\sim\mathcal{D}_{\tilde{x}},\\ \alpha,\alpha',\alpha''\sim\mathcal{D}_\alpha}} [\ell_{\text{ctr}}(\boldsymbol{x}^+, \boldsymbol{x}^{++}, \boldsymbol{x}^-)]$$
$$= \mathbb{E}_{\substack{(\boldsymbol{x},y)\sim\mathcal{D},\bar{\boldsymbol{x}}\sim\mathcal{D}_{\bar{y}},\\ \tilde{\boldsymbol{x}},\tilde{\boldsymbol{x}}',\tilde{\boldsymbol{x}}''\sim\mathcal{D}_{\tilde{x}},\\ \alpha,\alpha',\alpha''\sim\mathcal{D}_\alpha}} \left[\rho(y)\ell_{\text{cf}}\left(f(\boldsymbol{x}^+), y\right)\right] + \mathbb{E}_y[(1 - \bar{\rho}(y))\mathcal{E}_y]$$

where

$$\mathcal{E}_y = \mathbb{E}_{\substack{\boldsymbol{x},\bar{\boldsymbol{x}}\sim\mathcal{D}_y,\\ \tilde{\boldsymbol{x}},\tilde{\boldsymbol{x}}',\tilde{\boldsymbol{x}}''\sim\mathcal{D}_{\tilde{x}},\\ \alpha,\alpha',\alpha''\sim\mathcal{D}_\alpha}} \left[\log\left(1 + e^{-\frac{h(\boldsymbol{x}^+)^\top}{\|h(\boldsymbol{x}^+)\|}\left(\frac{h(\boldsymbol{x}^{++})}{\|h(\boldsymbol{x}^{++})\|} - \frac{h(\boldsymbol{x}^-)}{\|h(\boldsymbol{x}^-)\|}\right)}\right)\right],$$

$f(\boldsymbol{x}^+) = h(\boldsymbol{x}^+)^\top\tilde{w}$, $\bar{y} = 1 - y$, $\tilde{w} = \|h(\boldsymbol{x}^+)\|^{-1}(\|h(\pi_{y,1}(\boldsymbol{x}^{++}, \boldsymbol{x}^-))\|^{-1}h(\pi_{y,1}(\boldsymbol{x}^{++}, \boldsymbol{x}^-)) - \|h(\pi_{y,0}(\boldsymbol{x}^{++}, \boldsymbol{x}^-))\|^{-1}h(\pi_{y,0}(\boldsymbol{x}^{++}, \boldsymbol{x}^-)))$, *and* $\pi_{y,y'}(\boldsymbol{x}^{++}, \boldsymbol{x}^-) = 1_{[y=y']}\boldsymbol{x}^{++} + (1 - 1_{[y=y']})\boldsymbol{x}^-$.

All the proofs are presented in Appendix B. Theorem 1 proves the exact relationship for *training* loss when we set the distribution $\mathcal{D}$ to be an empirical distribution with Dirac measures on training data points: see Appendix A for more details. In general, Theorem 1 relates optimizing the contrastive loss $\ell_{\text{ctr}}(\boldsymbol{x}^+, \boldsymbol{x}^{++}, \boldsymbol{x}^-)$ to minimizing the classification loss $\ell_{\text{cf}}(f(\boldsymbol{x}^+), y_i)$ at the perturbed sample $\boldsymbol{x}^+$. The following theorem then shows that it is approximately minimizing the classification loss $\ell_{\text{cf}}(f(\boldsymbol{x}), y_i)$ at the original sample $\boldsymbol{x}$ with additional regularization terms on $\nabla f(\boldsymbol{x})$:

Theorem 2. *Let $\boldsymbol{x}$ and w be vectors such that $\nabla f(\boldsymbol{x})$ and $\nabla^2 f(\boldsymbol{x})$ exist. Assume that $f(\boldsymbol{x}) = \nabla f(\boldsymbol{x})^\top\boldsymbol{x}$, $\nabla^2 f(\boldsymbol{x}) = 0$, and $\mathbb{E}_{\tilde{\boldsymbol{x}}\sim\mathcal{D}_{\tilde{x}}}[\tilde{\boldsymbol{x}}] = 0$. Then, if $yf(\boldsymbol{x}) + (y - 1)f(\boldsymbol{x}) \geq 0$, the following two statements hold for any $\mathcal{D}_{\tilde{x}}$ and $\alpha > 0$:*

(i) (Mixup) *if $\delta(\boldsymbol{x},\tilde{\boldsymbol{x}}) = \tilde{\boldsymbol{x}} - \boldsymbol{x}$,*

$$\mathbb{E}_{\tilde{\boldsymbol{x}}\sim\mathcal{D}_{\tilde{x}}}[\ell_{\text{cf}}(f(\boldsymbol{x}^+), y)] \tag{7}$$
$$= \ell_{\text{cf}}(f(\boldsymbol{x}), y) + c_1(\boldsymbol{x})\|\nabla f(\boldsymbol{x})\| + c_2(\boldsymbol{x})\|\nabla f(\boldsymbol{x})\|^2 + c_3(\boldsymbol{x})\|\nabla f(\boldsymbol{x})\|^2_{\mathbb{E}_{\tilde{\boldsymbol{x}}\sim\mathcal{D}_{\tilde{x}}}[\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}^\top]} + O(\alpha^3),$$

(ii) (Gaussian-noise) *if $\delta(\boldsymbol{x},\tilde{\boldsymbol{x}}) = \tilde{\boldsymbol{x}} \sim \mathcal{N}(0, \sigma^2 I)$,*

$$\mathbb{E}_{\tilde{\boldsymbol{x}}\sim\mathcal{N}(0,\sigma^2 I)}[\ell_{\text{cf}}\left(f(\boldsymbol{x}^+), y\right)] \tag{8}$$
$$= \ell_{\text{cf}}(f(\boldsymbol{x}), y) + \sigma^2 c_3(\boldsymbol{x})\|\nabla f(\boldsymbol{x})\|^2 + O(\alpha^3),$$

where $c_1(\boldsymbol{x}) = \alpha|\cos(\nabla f(\boldsymbol{x}), \boldsymbol{x})|\|y - \psi(f(\boldsymbol{x}))\|\|\boldsymbol{x}\| \geq 0$, $c_2(\boldsymbol{x}) = \frac{\alpha^2|\cos(\nabla f(\boldsymbol{x}),\boldsymbol{x})|^2\|\boldsymbol{x}\|}{2}|\psi'(f(\boldsymbol{x}))| \geq 0$, and $c_3(\boldsymbol{x}) = \frac{\alpha^2}{2}|\psi'(f(\boldsymbol{x}))| > 0$. Here, ψ is the logic function

as $\psi(q) = \frac{\exp(q)}{1+\exp(q)}$ *(ψ' is its derivative),* $\cos(a, b)$ *is the cosine similarity of two vectors* a *and* b*, and* $\|v\|_M^2 = v^\top M v$ *for any positive semidefinite matrix* M.[1]

The assumptions of $f(\boldsymbol{x}) = \nabla f(\boldsymbol{x})^\top \boldsymbol{x}$ and $\nabla^2 f(\boldsymbol{x}) = 0$ in Theorem 2 are satisfied by feedforward deep neural networks with ReLU and max pooling (without skip connections) as well as by linear models. The condition of $yf(\boldsymbol{x}) + (y-1)f(\boldsymbol{x}) \geq 0$ is satisfied whenever the training sample $(\boldsymbol{x}, y)$ is classified correctly. In other words, Theorem 2 states that when the model classifies a training sample $(\boldsymbol{x}, y)$ correctly, a training algorithm implicitly minimizes the additional regularization terms for the sample $(\boldsymbol{x}, y)$, which partially explains the benefit of training after correct classification of training samples.

In Eq. (7)–(8), we can see that both the Mixup-noise and Gaussian-noise versions have different regularization effects on $\|\nabla f(\boldsymbol{x})\|$ — the Euclidean norm of the gradient of the model f with respect to input $\boldsymbol{x}$. In the case of the linear model, we know from previous work that the regularization on $\|\nabla f(\boldsymbol{x})\| = \|w\|$ indeed promotes generalization:

Remark 1. *(Bartlett & Mendelson, 2002)* Let $\mathcal{F}_b = \{\boldsymbol{x} \mapsto w^\top \boldsymbol{x} : \|w\|^2 \leq b\}$. Then, for any $\delta > 0$, with probability at least $1-\delta$ over an i.i.d. draw of n examples $((\boldsymbol{x}_i, y_i))_{i=1}^n$, the following holds for all $f \in \mathcal{F}_b$:

$$\mathbb{E}_{(\boldsymbol{x},y)}[1_{[(2y-1)\neq \mathrm{sign}(f(\boldsymbol{x}))]}] - \frac{1}{n}\sum_{i=1}^n \phi((2y_i-1)f(\boldsymbol{x}_i)) \leq 4L_\phi \sqrt{\frac{b c_{\boldsymbol{x}} d}{n}} + \sqrt{\frac{\ln(2/\delta)}{2n}}. \quad (9)$$

By comparing Eq. (7)–(8) and by setting $\mathcal{D}_{\tilde{x}}$ to be the input data distribution, we can see that the Mixup-noise version has additional regularization effect on $\|\nabla f(\boldsymbol{x})\|_{\Sigma_X}^2 = \|w\|_{\Sigma_X}^2$, while the Gaussian-noise version does not, where $\Sigma_X = \mathbb{E}_x[\boldsymbol{x}\boldsymbol{x}^\top]$ is the input covariance matrix. The following theorem shows that this implicit regularization with the Mixup-noise version can further reduce the generalization error:

Theorem 3. *Let* $\mathcal{F}_b^{(\mathrm{mix})} = \{\boldsymbol{x} \mapsto w^\top \boldsymbol{x} : \|w\|_{\Sigma_X}^2 \leq b\}$. *Then, for any* $\delta > 0$*, with probability at least* $1-\delta$ *over an iid draw of* n *examples* $((\boldsymbol{x}_i, y_i))_{i=1}^n$*, the following holds for all* $f \in \mathcal{F}_b^{(\mathrm{mix})}$*:*

$$\mathbb{E}_{(\boldsymbol{x},y)}[1_{[(2y-1)\neq \mathrm{sign}(f(\boldsymbol{x}))]}] - \frac{1}{n}\sum_{i=1}^n \phi((2y_i-1)f(\boldsymbol{x}_i)) \leq 4L_\phi \sqrt{\frac{b\,\mathrm{rank}(\Sigma_X)}{n}} + \sqrt{\frac{\ln(2/\delta)}{2n}}. \quad (10)$$

[1] We use this notation for conciseness without assuming that it is a norm. If M is only positive semidefinite instead of positive definite, $\|\cdot\|_M$ is not a norm since this does not satisfy the definition of the norm for positive definiteness; i.e., $\|v\| = 0$ does not imply $v = 0$.

Comparing Eq. (9)–(10), we can see that the proposed method with Mixup-noise has the advantage over the Gaussian-noise when the input data distribution lies in low dimensional manifold as $\mathrm{rank}(\Sigma_X) < d$. In general, our theoretical results show that the proposed method with Mixup-noise induces the implicit regularization on $\|\nabla f(\boldsymbol{x})\|_{\Sigma_X}^2$, which can reduce the complexity of the model class of f along the data manifold captured by the covariance Σ_X. See Appendix A for additional discussions on the interpretation of Theorems 1 and 2 for neural networks.

The proofs of Theorems 1 and 2 hold true also when we set $\boldsymbol{x}$ to be the output of a hidden layer and by redefining the domains of h and f to be the output of the hidden layer. Therefore, by treating $\boldsymbol{x}$ to be the output of a hidden layer, our theory also applies to the contrastive learning with positive samples created by mixing the hidden representations of samples. In this case, Theorems 1 and 2 show that the contrastive learning method implicitly regularizes $\|\nabla f(\boldsymbol{x}^{(l)})\|_{\mathbb{E}[\tilde{\boldsymbol{x}}^{(l)}(\tilde{\boldsymbol{x}}^{(l)})^\top]}$ — the norm of the gradient of the model f with respect to the output $\boldsymbol{x}^{(l)}$ of the l-th hidden layer in the direction of data manifold. Therefore, contrastive learning with Mixup-noise at the input space or a hidden space can promote generalization in the data manifold in the input space or the hidden space.

5. Experiments

We present results on three different application domains: tabular data, images, and graphs. For all datasets, to evaluate the learned representations under different contrastive learning methods, we use the linear evaluation protocol (Bachman et al., 2019; Hénaff et al., 2019; He et al., 2020; Chen et al., 2020b), where a linear classifier is trained on top of a frozen encoder network, and the test accuracy is used as a proxy for representation quality. Similar to SimCLR, we discard the projection-head during linear evaluation.

For each of the experiments, we give details about the architecture and the experimental setup in the corresponding section. In the following, we describe common hyperparameter search settings. For experiments on tabular and image datasets (Section 5.1 and 5.2), we search the hyperparameter α for linear mixing (Section 3 or line 5 in Algorithm 1) from the set $\{0.5, 0.6, 0.7, 0.8, 0.9\}$. To avoid the search over hyperparameter β (of Section 3.1), we set it to same value as α. For the hyperparameter ρ of Binary-Mixup (Section 3.1), we search the value from the set $[0.1, 0.3, 0.5]$. For Gaussian-noise based contrastive learning, we chose the mean of Gaussian-noise from the set $\{0.05, 0.1, 0.3, 0.5\}$ and the standard deviation is set to 1.0. For all experiments, the hyperparameter temperature τ (line 20 in Algorithm 1) is searched from the set $\{0.1, 0.5, 1.0\}$. For each of the experiments, we report the best values of aforementioned hyperparameters in the Appendix C .

For experiments on graph datasets (Section 5.3), we fix the value of α to 0.9 and value of temperature τ to 1.0.

5.1. Tabular Data

Why?? 😕😕

For tabular data experiments, we use Fashion-MNIST and CIFAR-10 datasets as a proxy by permuting the pixels and flattening them into a vector format. We use No-pretraining and Gaussian-noise based contrastive leaning as baselines. Additionally, we report supervised learning results (training the full network in a supervised manner).

We use a 12-layer fully-connected network as the base encoder and a 3-layer projection head, with ReLU non-linearity and batch-normalization for all layers. All pre-training methods are trained for 1000 epochs with a batch size of 4096. The linear classifier is trained for 200 epochs with a batch size of 256. We use LARS optimizer (You et al., 2017) with cosine decay schedule without restarts (Loshchilov & Hutter, 2017), for both pre-training and linear evaluation. The initial learning rate for both pre-training and linear classifier is set to 0.1.

Results: As shown in Table 1, DACL performs significantly better than the Gaussian-noise based contrastive learning. DACL+ , which uses additional Mixup-noises (Section 3.1), further improves the performance of DACL. More interestingly, our results show that the linear classifier applied to the representations learned by DACL gives better performance than training the full network in a supervised manner.

Method	Fashion-MNIST	CIFAR10
No-Pretraining	66.6	26.8
Gaussian-noise	75.8	27.4
DACL	81.4	37.6
DACL+	**82.4**	**39.7**
Full network supervised training	79.1	35.2

Table 1. Results on tabular data with a 12-layer fully-connected network.

5.2. Image Data

We use three benchmark image datasets: CIFAR-10, CIFAR-100, and ImageNet. For CIFAR-10 and CIFAR-100, we use No-Pretraining, Gaussian-noise based contrastive learning and SimCLR (Chen et al., 2020b) as baselines. For ImageNet, we use recent contrastive learning methods e.g. (Gidaris et al., 2018; Donahue & Simonyan, 2019; Bachman et al., 2019; Tian et al., 2019; He et al., 2020; Hénaff et al., 2019) as additional baselines. SimCLR+DACL refers to the combination of the SimCLR and DACL methods, which is implemented using the following steps: (1) for each training batch, compute the SimCLR loss and DACL loss separately and (2) pretrain the network using the sum of SimCLR and DACL losses.

Basically add another loss for DACL

For all experiments, we closely follow the details in SimCLR (Chen et al., 2020b), both for pre-training and linear evaluation. We use ResNet-50(x4) (He et al., 2016) as the base encoder network, and a 3-layer MLP projection-head to project the representation to a 128-dimensional latent space.

Pre-training: For SimCLR and SimCLR+DACL pretraining, we use the following augmentation operations: random crop and resize (with random flip), color distortions, and Gaussian blur. We train all models with a batch size of 4096 for 1000 epochs for CIFAR10/100 and 100 epochs for ImageNet.[2] We use LARS optimizer with learning rate 16.0 ($= 1.0 \times$ Batch-size$/256$) for CIFAR10/100 and 4.8 ($= 0.3 \times$ Batch-size$/256$) for ImageNet. Furthermore, we use linear warmup for the first 10 epochs and decay the learning rate with the cosine decay schedule without restarts (Loshchilov & Hutter, 2017). The weight decay is set to 10^{-6}.

LARS is 🔥🔥

Linear evaluation: To stay domain-agnostic, we do not use any data augmentation during the linear evaluation of No-Pretraining, Gaussian-noise, DACL and DACL+ methods in Table 2 and 3. For linear evaluation of SimCLR and SimCLR+DACL, we use random cropping with random left-to-right flipping, similar to (Chen et al., 2020b). For CIFAR10/100, we use a batch-size of 256 and train the model for 200 epochs, using LARS optimizer with learning rate 1.0 ($= 1.0 \times$ Batch-size$/256$) and cosine decay schedule without restarts. For ImageNet, we use a batch size of 4096 and train the model for 90 epochs, using LARS optimizer with learning rate 1.6 ($= 0.1 \times$ Batch-size$/256$) and cosine decay schedule without restarts. For both the CIFAR10/100 and ImageNet, we do not use weight-decay and learning rate warm-up.

Results: We present the results for CIFAR10/CIFAR100 and ImageNet in Table 2 and Table 3 respectively. We observe that DACL is better than Gaussian-noise based contrastive learning by a wide margin and DACL+ can improve the test accuracy even further. However, DACL falls short of methods that use image augmentations such as SimCLR (Chen et al., 2020b). This shows that the invariances learned using the image-specific augmentation methods (such as cropping, rotation, horizontal flipping) facilitate learning better representations than making the representations invariant to Mixup-noise. This opens up a further question: are the invariances learned from image-specific augmentations complementary to the Mixup-noise based invariances? To answer this, we combine DACL with SimCLR (Sim-

Amazing! 👌👌💥💥

[2]Our reproduction of the results of SimCLR for ImageNet in Table 3 differs from (Chen et al., 2020b) because our experiments are run for 100 epochs vs their 1000 epochs.

CLR+DACL in Table 2 and Table 3) and show that it can improve the performance of SimCLR across all the datasets. This suggests that Mixup-noise is complementary to other image data augmentations for contrastive learning.

Method	CIFAR-10	CIFAR-100
No-Pretraining	43.1	18.1
Gaussian-noise	56.1	29.8
DACL	81.3	46.5
DACL+	83.8	52.7
SimCLR	93.4	73.8
SimCLR+DACL	**94.3**	**75.5**

DACL fails to beat SimCLR

Table 2. Results on CIFAR10/100 with ResNet50(4×)

Method	Architecture	Param(M)	Top 1	Top 5
Rotation (Gidaris et al., 2018)	ResNet50 (4×)	86	55.4	-
BigBiGAN (Donahue & Simonyan, 2019)	ResNet50 (4×)	86	61.3	81.9
AMDIM (Bachman et al., 2019)	Custom-ResNet	626	68.1	-
CMC (Tian et al., 2019)	ResNet50 (2×)	188	68.4	88.2
MoCo (He et al., 2020)	ResNet50 (4×)	375	68.6	-
CPC v2 (Hénaff et al., 2019)	ResNet161	305	71.5	90.1
BYOL (300 epochs) (Grill et al., 2020)	ResNet50 (4×)	375	72.5	90.8
No-Pretraining	ResNet50 (4×)	375	4.1	11.5
Gaussian-noise	ResNet50 (4×)	375	10.2	23.6
DACL	ResNet50 (4×)	375	24.6	44.4
SimCLR (Chen et al., 2020b)	ResNet50 (4×)	375	73.4	91.6
SimCLR+DACL	ResNet50 (4×)	375	**74.4**	**92.2**

The gap 😳😳😳

Table 3. Accuracy of linear classifiers trained on representations learned with different self-supervised methods on the ImageNet dataset.

5.3. Graph-Structured Data

We present the results of applying DACL to graph classification problems using six well-known benchmark datasets: MUTAG, PTC-MR, REDDIT-BINARY, REDDIT-MULTI-5K, IMDB-BINARY, and IMDB-MULTI (Simonovsky & Komodakis, 2017; Yanardag & Vishwanathan, 2015). For baselines, we use No-Pretraining and InfoGraph (Sun et al., 2020). InfoGraph is a state-of-the-art contrastive learning method for graph classification problems, which is based on maximizing the mutual-information between the global and node-level features of a graph by formulating this as a contrastive learning problem.

For applying DACL to graph structured data, as discussed in Section 3, it is required to obtain fixed-length representations from an intermediate layer of the encoder. For graph neural networks, e.g. Graph Isomorphism Network (GIN) (Xu et al., 2018), such fixed-length representation can be obtained by applying global pooling over the node-level representations at any intermediate layer. Thus, the Mixup-noise can be applied to any of the intermediate layer by adding an auxiliary feed-forward network on top of such intermediate layer. However, since we follow the encoder and projection-head architecture of SimCLR, we can also apply the Mixup-noise to the output of the encoder. In this work, we present experiments with Mixup-noise applied to the output of the encoder and leave the experiments with Mixup-noise at intermediate layers for future work.

We closely follow the experimental setup of InfoGraph (Sun et al., 2020) for a fair comparison, except that we report results for a linear classifier instead of the Support Vector Classifier applied to the pre-trained representations. This choice was made to maintain the coherency of evaluation protocol throughout the paper as well as with respect to the previous state-of-the-art self-supervised learning papers. [3] For all the pre-training methods in Table 4, as graph encoder network, we use GIN (Xu et al., 2018) with 4 hidden layers and node embedding dimension of 512. The output of this encoder network is a fixed-length vector of dimension 4×512. Further, we use a 3-layer projection-head with its hidden state dimension being the same as the output dimension of a 4-layer GIN (4×512). Similarly for InfoGraph experiments, we use a 3-layer discriminator network with hidden state dimension 4×512.

For all experiments, for pretraining, we train the model for 20 epochs with a batch size of 128, and for linear evaluation, we train the linear classifier on the learned representations for 100 updates with full-batch training. For both pre-training and linear evaluation, we use Adam optimizer (Kingma & Ba, 2014) with an initial learning rate chosen from the set $\{10^{-2}, 10^{-3}, 10^{-4}\}$. We perform linear evaluation using 10-fold cross-validation. Since these datasets are small in the number of samples, the linear-evaluation accuracy varies significantly across the pre-training epochs. Thus, we report the average of linear classifier accuracy over the last five pre-training epochs. All the experiments are repeated five times.

Results: In Table 4 we see that DACL closely matches the performance of InfoGraph, with the classification accuracy of these methods being within the standard deviation of each other. In terms of the classification accuracy mean, DACL outperforms InfoGraph on four out of six datasets. This result is particularly appealing because we have used no domain knowledge for formulating the contrastive loss, yet achieved performance comparable to a state-of-the-art graph contrastive learning method.

6. Related Work

Self-supervised learning: Self-supervised learning methods can be categorized based on the pretext task they seek

[3] Our reproduction of the results for InfoGraph differs from (Sun et al., 2020) because we apply a linear classifier instead of Support Vector Classifier on the pre-trained features.

Dataset	MUTAG	PTC-MR	REDDIT-BINARY	REDDIT-M5K	IMDB-BINARY	IMDB-MULTI
No. Graphs	188	344	2000	4999	1000	1500
No. classes	2	2	2	5	2	3
Avg. Graph Size	17.93	14.29	429.63	508.52	19.77	13.00
			Method			
No-Pretraining	81.70 ± 2.58	53.07 ± 1.27	55.13 ± 1.86	24.27 ± 0.93	52.67 ± 2.08	33.72 ± 0.80
InfoGraph (Sun et al., 2020)	$\mathbf{86.74 \pm 1.28}$	57.09 ± 1.52	63.52 ± 1.66	$\mathbf{42.89 \pm 0.62}$	63.97 ± 2.05	39.28 ± 1.43
DACL	85.31 ± 1.34	$\mathbf{59.24 \pm 2.57}$	$\mathbf{66.92 \pm 3.38}$	42.86 ± 1.11	$\mathbf{64.71 \pm 2.13}$	$\mathbf{40.16 \pm 1.50}$

Table 4. Classification accuracy using a linear classifier trained on representations obtained using different self-supervised methods on 6 benchmark graph classification datasets.

to learn. For instance, in (de Sa, 1994), the pretext task is to minimize the disagreement between the outputs of neural networks processing two different modalities of a given sample. In the following, we briefly review various pretext tasks across different domains. In the natural language understanding, pretext tasks include, predicting the neighbouring words (word2vec (Mikolov et al., 2013)), predicting the next word (Dai & Le, 2015; Peters et al., 2018; Radford et al., 2019), predicting the next sentence (Kiros et al., 2015; Devlin et al.), predicting the masked word (Devlin et al.; Yang et al., 2019; Liu et al.; Lan et al., 2020)), and predicting the replaced word in the sentence (Clark et al., 2020). For computer vision, examples of pretext tasks include rotation prediction (Gidaris et al., 2018), relative position prediction of image patches (Doersch et al., 2015), image colorization (Zhang et al., 2016), reconstructing the original image from the partial image (Pathak et al., 2016; Zhang et al., 2017), learning invariant representation under image transformation (Misra & van der Maaten, 2020), and predicting an odd video subsequence in a video sequence (Fernando et al., 2017). For graph-structured data, the pretext task can be predicting the context (neighbourhood of a given node) or predicting the masked attributes of the node (Hu et al., 2020). Most of the above pretext tasks in these methods are domain-specific, and hence they cannot be applied to other domains. Perhaps a notable exception is the language modeling objectives, which have been shown to work for both NLP and computer vision (Dai & Le, 2015; Chen et al., 2020a).

Contrastive learning: Contrastive learning is a form of self-supervised learning where the pretext task is to bring positive samples closer than the negative samples in the representation space. These methods can be categorized based on how the positive and negative samples are constructed. In the following, we will discuss these categories and the domains where these methods cannot be applied: (a) this class of methods use domain-specific augmentations (Chopra et al., 2005; Hadsell et al., 2006; Ye et al., 2019; He et al., 2020; Chen et al., 2020b; Caron et al., 2020) for creating positive and negative samples. These methods are state-of-the-art for computer vision tasks but can not be applied to domains where semantic-preserving data augmentation does not exist, such as graph-data or tabular data. (b) another class of methods constructs positive and negative samples by defining the local and global context in a sample (Hjelm et al., 2019; Sun et al., 2020; Veličković et al., 2019; Bachman et al., 2019; Trinh et al., 2019). These methods can not be applied to domains where such global and local context does not exist, such as tabular data. (c) yet another class of methods uses the ordering in the sequential data to construct positive and negative samples (Oord et al., 2018; Hénaff et al., 2019). These methods cannot be applied if the data sample cannot be expressed as an ordered sequence, such as graphs and tabular data. Thus our motivation in this work is to propose a contrastive learning method that can be applied to a wide variety of domains.

Mixup based methods: Mixup-based methods allow inducing inductive biases about how a model's predictions should behave in-between two or more data samples. Mixup(Zhang et al., 2018; Tokozume et al., 2017) and its numerous variants(Verma et al., 2019a; Yun et al., 2019; Faramarzi et al., 2020) have seen remarkable success in supervised learning problems, as well other problems such as semi-supervised learning (Verma et al., 2019b; Berthelot et al., 2019), unsupervised learning using autoencoders (Beckham et al., 2019; Berthelot et al., 2019), adversarial learning (Lamb et al., 2019; Lee et al., 2020; Pang et al., 2020), graph-based learning (Verma et al., 2021; Wang et al., 2020), computer vision (Yun et al., 2019; Jeong et al., 2020; Panfilov et al., 2019), natural language (Guo et al., 2019; Zhang et al., 2020) and speech (Lam et al., 2020; Tomashenko et al., 2018). In contrastive learning setting, Mixup-based methods have been recently explored in (Shen et al., 2020; Kalantidis et al., 2020; Kim et al., 2020b). Our work differs from aforementioned works in important aspects: unlike these methods, we theoretically demonstrate why Mixup-noise based directions are better than Gaussian-noise for constructing positive pairs, we propose other forms of Mixup-noise and show that these forms are complementary to linear Mixup-noise, and experimentally validate our method across different domains. We also note that Mixup based contrastive learning methods such

as ours and (Shen et al., 2020; Kalantidis et al., 2020; Kim et al., 2020b) have advantage over recently proposed adversarial direction based contrastive learning method (Kim et al., 2020a) because the later method requires additional gradient computation.

7. Discussion and Future Work

In this work, with the motivation of designing a domain-agnostic self-supervised learning method, we study Mixup-noise as a way for creating positive and negative samples for the contrastive learning formulation. Our results show that the proposed method DACL is a viable option for the domains where data augmentation methods are not available. Specifically, for tabular data, we show that DACL and DACL+ can achieve better test accuracy than training the neural network in a fully-supervised manner. For graph classification, DACL is on par with the recently proposed mutual-information maximization method for contrastive learning (Sun et al., 2020). For the image datasets, DACL falls short of those methods which use image-specific augmentations such as random cropping, horizontal flipping, color distortions, etc. However, our experiments show that the Mixup-noise in DACL can be used as complementary to image-specific data augmentations. As future work, one could easily extend DACL to other domains such as natural language and speech. From a theoretical perspective, we have analyzed DACL in the binary classification setting, and extending this analysis to the multi-class setting might shed more light on developing a better Mixup-noise based contrastive learning method. Furthermore, since different kinds of Mixup-noise examined in this work are based only on *random* interpolation between *two* samples, extending the experiments by mixing between more than two samples or *learning* the optimal mixing policy through an auxiliary network is another promising avenue for future research.

References

Bachman, P., Hjelm, R. D., and Buchwalter, W. Learning representations by maximizing mutual information across views. In *NeurIPS*. 2019. 5, 6, 7, 8

Baevski, A., Zhou, H., Mohamed, A., and Auli, M. wav2vec 2.0: A framework for self-supervised learning of speech representations. In *NeurIPS*, 2020. 1

Bartlett, P. L. and Mendelson, S. Rademacher and gaussian complexities: Risk bounds and structural results. *JMLR*, 2002. 5, 21

Beckham, C., Honari, S., Verma, V., Lamb, A. M., Ghadiri, F., Hjelm, R. D., Bengio, Y., and Pal, C. In *NeurIPS*, 2019. 3, 8

Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., and Raffel, C. MixMatch: A Holistic Approach to Semi-Supervised Learning. In *NeurIPS*, 2019. 8

Berthelot, D., Raffel, C., Roy, A., and Goodfellow, I. Understanding and improving interpolation in autoencoders via an adversarial regularizer. In *ICLR*, 2019. 8

Cai, Q., Wang, Y., Pan, Y., Yao, T., and Mei, T. Joint contrastive learning with infinite possibilities. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 12638–12648. Curran Associates, Inc., 2020. 1

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. Unsupervised learning of visual features by contrasting cluster assignments, 2020. 8

Chen, M., Radford, A., Child, R., Wu, J., Jun, H., Luan, D., and Sutskever, I. Generative pretraining from pixels. In *ICML*, 2020a. 8

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. E. A simple framework for contrastive learning of visual representations. In *ICML*, 2020b. 1, 2, 3, 5, 6, 7, 8

Chopra, S., Hadsell, R., and LeCun, Y. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005. 1, 8

Clark, K., Luong, M.-T., Le, Q. V., and Manning, C. D. Electra: Pre-training text encoders as discriminators rather than generators. In *ICLR*, 2020. 1, 8

Dai, A. M. and Le, Q. V. Semi-supervised sequence learning. In *Advances in neural information processing systems*, pp. 3079–3087, 2015. 1, 8

de Sa, V. R. Learning classification with unlabeled data. In *Advances in Neural Information Processing Systems 6*. 1994. 8

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *ACL*. 8

Doersch, C., Gupta, A., and Efros, A. A. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015. 8

Donahue, J. and Simonyan, K. Large scale adversarial representation learning. In *NeurIPS*, 2019. 6, 7

Faramarzi, M., Amini, M., Badrinaaraayanan, A., Verma, V., and Chandar, S. Patchup: A regularization technique for convolutional neural networks. *arXiv preprint arXiv:2006.07794*, 2020. 8

Fernando, B., Bilen, H., Gavves, E., and Gould, S. Self-supervised video representation learning with odd-one-out networks. In *CVPR*, 2017. 8

Gidaris, S., Singh, P., and Komodakis, N. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018. 6, 7, 8

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., Piot, B., kavukcuoglu, k., Munos, R., and Valko, M. Bootstrap your own latent - a new approach to self-supervised learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 21271–21284. Curran Associates, Inc., 2020. 1, 7

Guo, H., Mao, Y., and Zhang, R. Augmenting data with mixup for sentence classification: An empirical study. *arXiv preprint arXiv:1905.08941*, 2019. 8

Hadsell, R., Chopra, S., and LeCun, Y. Dimensionality reduction by learning an invariant mapping. CVPR '06, pp. 1735–1742, USA, 2006. IEEE Computer Society. ISBN 0769525970. doi: 10.1109/CVPR.2006.100. URL https://doi.org/10.1109/CVPR.2006.100. 1, 2, 8

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *CVPR*, pp. 770–778, 2016. 6

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. B. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 1, 2, 5, 6, 7, 8

Hénaff, O. J., Srinivas, A., Fauw, J. D., Razavi, A., Doersch, C., Eslami, S. M. A., and van den Oord, A. Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*, 2019. 1, 5, 6, 7, 8

Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., and Bengio, Y. Learning deep representations by mutual information estimation and maximization. In *ICLR*, 2019. URL https://openreview.net/forum?id=Bklr3j0cKX. 8

Howard, J. and Ruder, S. Universal language model fine-tuning for text classification. In *ACL*, 2018. 1

Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., and Leskovec, J. Strategies for pre-training graph neural networks. In *ICLR*, 2020. 8

Jeong, J., Verma, V., Hyun, M., Kannala, J., and Kwak, N. Interpolation-based semi-supervised learning for object detection, 2020. 8

Kalantidis, Y., Sariyildiz, M. B., Pion, N., Weinzaepfel, P., and Larlus, D. Hard negative mixing for contrastive learning, 2020. 8, 9

Kawaguchi, K., Kaelbling, L. P., and Bengio, Y. Generalization in deep learning. *arXiv preprint arXiv:1710.05468*, 2017. 13

Kim, M., Tack, J., and Hwang, S. J. Adversarial self-supervised contrastive learning, 2020a. 9

Kim, S., Lee, G., Bae, S., and Yun, S.-Y. Mixco: Mix-up contrastive learning for visual representation, 2020b. 8, 9

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2014. URL http://arxiv.org/abs/1412.6980. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015. 7

Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., and Fidler, S. Skip-thought vectors. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28*, pp. 3294–3302. Curran Associates, Inc., 2015. URL http://papers.nips.cc/paper/5950-skip-thought-vectors.pdf. 8

Lam, M. W. Y., Wang, J., Su, D., and Yu, D. Mixup-breakdown: A consistency training method for improving generalization of speech separation models. In *ICASSP*, 2020. 8

Lamb, A., Verma, V., Kannala, J., and Bengio, Y. Interpolated adversarial training: Achieving robust neural networks without sacrificing too much accuracy. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, 2019. 8

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. Albert: A lite bert for self-supervised learning of language representations. In *ICLR*, 2020. 8

Lee, S., Lee, H., and Yoon, S. Adversarial vertex mixup: Toward better adversarially robust generalization. *CVPR*, 2020. 8

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*. 8

Loshchilov, I. and Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 6

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases

and their compositionality. In *Advances in Neural Information Processing Systems 26*. 2013. 8

Misra, I. and van der Maaten, L. Self-supervised learning of pretext-invariant representations. *CVPR*, 2020. 8

Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 1, 2, 8

Panfilov, E., Tiulpin, A., Klein, S., Nieminen, M. T., and Saarakkala, S. Improving robustness of deep learning based knee mri segmentation: Mixup and adversarial domain adaptation. In *ICCV Workshop*, 2019. 8

Pang, T., Xu, K., and Zhu, J. Mixup inference: Better exploiting mixup to defend adversarial attacks. In *ICLR*, 2020. 8

Pathak, D., Krähenbühl, P., Donahue, J., Darrell, T., and Efros, A. A. Context encoders: Feature learning by inpainting. In *CVPR*, pp. 2536–2544, 2016. 8

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018. 1, 8

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2019. 1, 8

Schneider, S., Baevski, A., Collobert, R., and Auli, M. wav2vec: Unsupervised pre-training for speech recognition. In *Interspeech*, 2019. 1

Shen, Z., Liu, Z., Liu, Z., Savvides, M., and Darrell, T. Rethinking image mixture for unsupervised visual representation learning, 2020. 8, 9

Simonovsky, M. and Komodakis, N. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *CVPR*, 2017. 7

Sohn, K. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*. 2016. 2

Sun, F.-Y., Hoffman, J., Verma, V., and Tang, J. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *ICLR*, 2020. 7, 8, 9

Tian, Y., Krishnan, D., and Isola, P. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019. 6, 7

Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., and Isola, P. What makes for good views for contrastive learning? In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6827–6839. Curran Associates, Inc., 2020. 1

Tokozume, Y., Ushiku, Y., and Harada, T. Between-class learning for image classification. In *CVPR*, 2017. 8

Tomashenko, N., Khokhlov, Y., and Estève, Y. Speaker adaptive training and mixup regularization for neural network acoustic models in automatic speech recognition. In *Interspeech*, pp. 2414–2418, 09 2018. 8

Trinh, T. H., Luong, M., and Le, Q. V. Selfie: Self-supervised pretraining for image embedding. *arXiv preprint arXiv:1906.02940*, 2019. 8

Veličković, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. Deep graph infomax. In *ICLR*, 2019. 8

Verma, V., Lamb, A., Beckham, C., Najafi, A., Mitliagkas, I., Lopez-Paz, D., and Bengio, Y. Manifold mixup: Better representations by interpolating hidden states. In *ICML*, 2019a. 2, 8

Verma, V., Lamb, A., Juho, K., Bengio, Y., and Lopez-Paz, D. Interpolation consistency training for semi-supervised learning. In *IJCAI*, 2019b. 8

Verma, V., Qu, M., Kawaguchi, K., Lamb, A., Bengio, Y., Kannala, J., and Tang, J. Graphmix: Improved training of gnns for semi-supervised learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(11):10024–10032, May 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/17203. 8

Wang, T. and Isola, P. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. 119:9929–9939, 13–18 Jul 2020. 1

Wang, Y., Wang, W., Liang, Y., Cai, Y., Liu, J., and Hooi, B. Nodeaug: Semi-supervised node classification with data augmentation. In *KDD*, 2020. 8

Weinberger, K. Q. and Saul, L. K. Distance metric learning for large margin nearest neighbor classification. *JMLR*, 2009. 2

Wu, Z., Xiong, Y., Yu, S. X., and Lin, D. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018. 7

Yanardag, P. and Vishwanathan, S. Deep graph kernels. In *KDD*, 2015. 7

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pp. 5753–5763, 2019. 8

Ye, M., Zhang, X., Yuen, P. C., and Chang, S.-F. Unsupervised embedding learning via invariant and spreading instance feature. In *CVPR*, 2019. 8

You, Y., Gitman, I., and Ginsburg, B. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017. 6

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., and Yoo, Y. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019. 8

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. *ICLR*, 2018. 2, 8

Zhang, R., Isola, P., and Efros, A. A. Colorful image colorization. In *ECCV*, 2016. 8

Zhang, R., Isola, P., and Efros, A. A. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *ICCV*, 2017. 8

Zhang, R., Yu, Y., and Zhang, C. Seqmix: Augmenting active sequence labeling via sequence mixup. In *EMNLP*, 2020. 8

Zhuang, C., Zhai, A. L., and Yamins, D. Local aggregation for unsupervised learning of visual embeddings. *ICCV*, 2019. 2

SUMMARY

Page 1

- Contrastive self-supervised learning has been very successful in the recent years. We saw many advancements like SimCLR, MOCO, etc in self-supervised learning that leverage contrastive learning. Although this approach has been very successful, most of these methods are domain-specific and rely heavily on data augmentation techniques.
- Data augmentation requires a bit of domain knowledge (although "bit" isn't enough). How? Given that most SSL formulate the problem by defining a pretext task, the pretext is domain specific hence requires a bit of domain knowledge. Plus, there are domains where data augmentation is not very straightforward or readily available e.g. tabular data where augmentation is more nuanced than other domains like computer vision.
- To overcome the above limitation, the authors proposed Domain Agnostic Contrastive Learning (DACL). The key to their approach is the use of Mixup noise to create similar and dissimilar examples by mixing the data samples differently either at input level or hidden state levels

Page 2, 3

- Contrastive Learning
  - CL involves two things. First, the selection of an anchor and corresponding positive and negative samples. Second, an objective that tries to bring the anchor and the positive samples closer in the embedding space than the anchor and the negative samples.
  - The above condition is fulfilled by choosing a similarity function such that sim(h, h+) > sim (h, h-). Although the similarity can be formulated in a number of ways, the most popular choice nowadays is the pairwise cosine similarity.
  - The objective function can also take many forms but the most popular nowadays is the InfoNCE loss used in SimCLR. Check equation 2) for more details

- Domain Agnostic Contrastive Learning with Mixup
  - Mixup is applied in two different ways. For data where we have a defined topology, the mixup is just data interpolation. For an anchor x, a positive sample is created in this fashion: x+ = (lambda * x) + (1-lambda)* x' Lambda is sampled from a uniform distribution U(alpha, 1.0) with alpha high alpha values. High alpha value makes sure that the anchor is more dominant in the mixed up sample. X' is just another sample from the input distribution.
  - For domains where the topology isn't fixed, the same interpolation is applied but on hidden representations of samples. For example if v is the hidden representation, then a positive sample is created as v+ = (lambda * v) + (1- lambda) * v'
  - The hypothesis for this particular method to work better compared to other types of noise is that a network will be forced to learn better if the noise captures the structure of the data manifold rather than being independent of it e.g. case of a Gaussian Noise. Mixup makes it hard for the network to figure out ways to bypass the noise forcing it to learn better features

- Forms of Mixup
  - Linear mixup, the one that we just saw above is an example of linear mixup
  - Geometric Mixup x+ = Weighted mean of (x^lambda , x'^(1-lambda)). Lambda is sampled from U (beta, 1.0) with high beta values
  - Binary Mixup x+ = (x O m) + (x' O (1-m)) m is sampled from a Bernoulli distribution and the mask is a binary mask

- Because we have three kinds of mixup, we can randomly choose one of them at each iteration and apply it for better training results

Page 5-7

- Experiments
  - For all experiments as common in Contrastive learning an encoder is used for pretraining. For linear evaluation, same protocol is followed as done in normal contrastive learning I.e. a linear classifier is trained on top of the frozen encoder and the test accuracy is used as a proxy for representational quality. The projection head is discarded during linear evaluation
  - For tabular and image data, hparam search is done for the alpha parameter from this set (0.5, 0.6, 0.7, 0.8, 0.9). The value of beta is set equal to alpha.
  - For binary mixup, rho parameter is search from th set (0.1, 0.3, 0.5)
  - For Gaussian noise, the mean is chosen from (0.05, 0.1, 0.3, 0.5) with a standard deviation of 1.0
  - For tabular data, 12 layer FFN is used with ReLU and BN, for image data ResNet 50(x4) along with 3 layer projection head is used., and for graph data GIN network is used for pretraining
  - For all experiments LARS is used for training. For some data, LARS is used during evaluation as well and for others Adam is used during evaluation

Criticism related to some of the experiments

- The authors used FMNIST and CIFAR-10 for tabular data, which IMO is a very bad choice. Tabular datasets are readily available and they could have picked up any dataset from Kaggle and should have compared the performance with and without Mixup.
- Compared to SimCLR, the performance of DACL is very poor. And when used with SimCLR, the performance boost is okayish, suggesting that Mixup is just complementary to the image data augmentations, not superior or comparable to other image based data augmentation used in Contrastive learning