Neurocomputing 593 (2024) 127810

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

pNNCLR: Stochastic pseudo neighborhoods for contrastive learning based unsupervised representation learning problems

Momojit Biswas [a], Himanshu Buckchash [b,*], Dilip K. Prasad [b]

[a] Jadavpur University, Kolkata, India
[b] Department of Computer Science, UiT The Arctic University of Norway, Tromso, Norway

ARTICLE INFO

Communicated by X. Gao

Keywords:
Self supervised learning
Deep learning
Image classification
Contrastive learning
Pseudo nearest neighbors

ABSTRACT

Nearest neighbor (NN) sampling provides more semantic variations than predefined transformations for self-supervised learning (SSL) based image recognition problems. However, its performance is restricted by the quality of the support set, which holds positive samples for the contrastive loss. In this work, we show that the quality of the support set plays a crucial role in any nearest neighbor based method for SSL. We then provide a refined baseline (pNNCLR) to the nearest neighbor based SSL approach (NNCLR). To this end, we introduce pseudo nearest neighbors (pNN) to control the quality of the support set, wherein, rather than sampling the nearest neighbors, we sample in the vicinity of hard nearest neighbors by varying the magnitude of the resultant vector and employing a stochastic sampling strategy to improve the performance. Additionally, to stabilize the effects of uncertainty in NN-based learning, we employ a smooth-weight-update approach for training the proposed network. Evaluation of the proposed method on multiple public image recognition and medical image recognition datasets shows that it performs up to 8 percent better than the baseline nearest neighbor method, and is comparable to other previously proposed SSL methods. The code is available at https://github.com/mb16biswas/pnnclr.

1. Introduction

Deep learning is rapidly revolutionizing almost every sector of our society. Off-the-shelf models are being used for feature/representation extraction, and standard models are being fine-tuned for their application to specific problems [1]. To train such models, efficient representation learning methods are required [2]. SSL or representation learning provides the backbone networks for many computer vision related tasks such as object detection, segmentation, image or video recognition, etc. [3]. Recent developments like NNCLR [4], SimSiam [5], Decoupled contrastive learning [2], CLIP [6], CAEs [7], are good examples of powerful feature extractors, and all employ some standard backbone network like ResNet [8] or EfficientNet [9]. Labeling the data is a costly operation, on the other hand, the main advantage of SSL models is their ability to learn better generic representations from the unlabeled data [10]. Foundational works like SimCLR [10] and SimSiam [5] have established that SSL models with slight finetuning (as low as 1% of the labeled data) can outperform their counterpart supervised models. Another advantage of SSL models is that they provide task-agnostic models which can easily be adapted using transfer learning to multiple kinds of downstream tasks (the tasks which are specialized cases of a larger generic task also known as the pretext task) [11].

Earlier models like the non-contrastive models (RotNet [11], Jigsaw [12]) used intelligently designed pretext tasks for providing the self-supervisory signal to the learning algorithm. These signals were based on some independent tasks like verifying the correct rotation [11], the correct sequence of frames [13], or the correct placement of tiles [12]. Recently, another branch of SSL methods, called contrastive learning (CL), has shown promising progress [10]. With better loss functions and image augmentations, these models have now exceeded the non-contrastive models. These contrastive learning based methods work by pushing closer the similar-looking (positive) samples and pushing apart non-similar class (negative) samples, without actually knowing the classes of these samples. However, recently, it was shown that the positives generated using augmentations are not very semantically diverse [4]. To overcome this, it was suggested to use the nearest neighbors of the anchors (the samples whose positive is to be found), since this leads to better representations by learning from non-trivial positive samples [4]. However, in our experiments and analysis with NNCLR [4], we found that the quality of the support set plays a crucial role in learning better representations. At the beginning of training, the

* Corresponding author.
E-mail address: himanshu.buckchash@uit.no (H. Buckchash).

https://doi.org/10.1016/j.neucom.2024.127810
Received 13 August 2023; Received in revised form 29 February 2024; Accepted 5 May 2024
Available online 11 May 2024

probability of finding a good nearest neighbor is low, and this can affect the overall learning in the SSL model. Based on these observations, this work presents stochastic pseudo nearest neighbors and the learning framework — pNNCLR.

The main objective of contrastive SSL methods is to employ a strategy or function, f, to arrange the latent space in such a way that similar class samples appear closer (attraction property) than distinct class samples (repulsion property) in the latent space. Nearest neighbor based methods try to amplify the diversity in the attraction process. The source of this diversity is the process of sampling the positives in the form of nearest neighbors and not augmented views. However, this amplification of diversity introduces a trade-off between the attraction and repulsion properties. Because, if the quality of the support set is low or the chosen nearest neighbors are incorrect (Section 3.3), f may negatively impact the main objective by reducing the intended attraction and repulsion properties. To improve this trade-off, independent of the support set quality, we hypothesize modifying f such that the diversity in the attraction process is amplified favorably, i.e., the positive samples retain the semantic variations and, at the same time, are not unfavorably distinct from the anchor point. We accomplish this by reducing the magnitude of the resultant vector in the direction of the nearest neighbor by a factor. Although, this controls the diversification in the attraction process, however, it reduces the semantic quality. To avoid this, a stochastic prior is imposed during the sampling of positives. This allows the expansion of uncertainty to increase the semantic information. As a consequence of these adaptations, the positives become more semantically diverse and related to the anchor point, thereby, helping in learning better representations by the model. We also found that by employing a smooth-weight-updation approach, the effects of uncertainty, introduced by nearest neighbor based learning, can be stabilized to a significant extent.

The SimCLR [10] method has served as the main motivation for different self supervised learning methods such as NNCLR [4], DCLR [2], SimSiam [5], MoCo [14], and also to the proposed pNNCLR method. All these methods share the same Siamese network architecture. However, each new method has added an improvement to the SimCLR methods. NNCLR [4] added the idea of using nearest neighbors in order to increase the semantic variations to achieve better learning. In the same lines, the proposed pNNCLR method overcomes the challenges of NNCLR [4] method by adding the idea of pseudo nearest neighbors and stochastic sampling in nearest neighbor based contrastive learning methods.

We tested the proposed ideas and found that they significantly improve the performance of the proposed method over our baseline, NNCLR [4], on image recognition tasks. Following this, we also tested it for the medical datasets and found a favorable performance. Following are the main contributions of this work:

- We studied the suitability of nearest neighbor based semantic information enrichment in contrastive learning. This led us to the observation that a pseudo nearest neighbor approach may work better than a hard nearest neighbor approach. Following this, we introduced stochastic pseudo nearest neighbors (pNN) to control the quality of the positive samples in contrastive learning based SSL methods. To the best of our knowledge, we are the first to introduce a pseudo nearest neighbor based self-supervised learning approach in the family of contrastive learning based self-supervised learning methods. It significantly improved the performance of the proposed pNNCLR method.
- We theoretically proved that the nearest neighbor based approaches are highly sensitive in the beginning phase of training and may therefore reach a sub-optimal performance. Using this motivation, we developed the pseudo nearest neighbor sampling (pNN). We further improved our pNN approach by incorporating the idea of stochasticity in the pseudo nearest neighbors.
- We showed that a smooth-weight-updation approach in nearest neighbor based contrastive learning methods is highly useful in controlling the uncertainty in sampling. Using this, we proposed our contrastive learning based method called pNNCLR, which significantly improves over our baseline NNCLR [4] method.
- We performed comprehensive experiments and ablation studies to empirically verify the superiority of our contribution over medical as well as non-medical datasets.

The remainder of the paper is organized as follows. Section 2 presents the related literature on representation learning methods. Section 3 presents the details of our baseline and the proposed method. Section 4 presents the implementation details, evaluation, and comparison of the results on different datasets. After this, the conclusion and appendix are presented.

2. Related work

Representation learning methods have underpinned some of the recent highly successful pretrained networks and amazing feats of AI — GPT 3 [15], CLIP [6], SAM [16], ChatGPT [17], SEER [18]. These representation learning methods are mainly trained in a self-supervised manner. A couple of years earlier, self-supervised approaches were dominated by non-contrastive methods or by pretext task based methods (Context prediction [19], Jigsaw [12], RotNet [11], Low-rank embedding [20], Multigraph weight learning [21]). However, the trend is shifting as the current best self-supervised methods are all based on some form of contrastive learning approach (SBCL [22], DINO [23]). Our baseline, NNCLR [4], is one such contrastive learning based method that employs nearest neighbor sampling. In this literature review, we cover the developments in SSL from the perspective of both — non-contrastive and contrastive — self-supervised methods. Table 1 provides a consolidated view of the literature.

2.1. Non-contrastive SSL methods

In the context of self-supervised learning, a *pretext task* refers to a puzzle or sub-task that is solved by the SSL model. The objective is to learn the underlying structure of the data by deriving a supervision signal from the sub-task in an unsupervised manner, i.e., without relying on any labeled data [19]. Doersch et al. explored spatial context as a supervision signal for training visual representations [19]. Their approach involved dividing a region in image into 9 patches, then sampling pairs from these patches and training the model to predict the relative position of a patch given another patch from the pair. They achieved unsupervised object discovery and improved performance on object detection tasks. Zhang et al. used cross channel color space prediction as a pretext task [24,26]. They used a CNN to predict *ab* color space from L channel of the CIE Lab^* color space. It was found that colorization could be a useful option for learning representations for vision tasks. Pathak et al. also used context information for their inpainting pretext task by forcing their context encoding CNN to predict the missing region in an input image [25]. Although predicting the entire region is an under constrained task, however, their approach produced strong image representations. Another line of work (Shuffle learn [13], Sequence sorting [33], Sustained order verification [34], Odd one out [35]), used frame order prediction as the pretext task. These models learned meaningful image representations for vision tasks. However, it was not as strong as the representations learned by other pretext tasks, like context prediction or spatial rearrangement. Noroozi et al. proposed an even more challenging pretext task of sorting all nine pieces of a Jigsaw puzzle [12]. They also suggested several shortcut prevention approaches as they emphasized — "A good self-supervised task is neither simple nor ambiguous". Gidaris et al. proposed a simple rotation prediction as the pretext task for CNNs [11]. The objective was to predict the angle of rotation from 0, 90, 180,

Table 1
A consolidated literature review is presented for both — contrastive and non-contrastive SSL methods. Note that most of the backbones are CNN based.

Author	Year	Method	Contrastive	Backbone	Approach/Pretext-task
Doersch et al. [19]	2015	SpatialContext	✗	VGG	Predicting spatial context of a patch in relation to another patch in a spatially consistent array of nine patches.
Zhang et al. [24]	2016	CCEncoder	✗	In-house, VGG	Cross channel prediction using auto-encoder network.
Pathak et al. [25]	2016	ContextEncoder	✗	AlexNet	Inpainting of missing patch using context auto-encoders with channelwise fully-connected layers.
Misra et al. [13]	2016	ShuffleAndLearn	✗	SiameseAlexNet	Ordering of frames with sequence binary verification.
Noroozi et al. [12]	2016	ContextFreeNet-work	✗	SiameseAlexNet	Rearrangement of shuffled Jigsaw puzzle like sub-images.
Zhang et al. [26]	2017	SplitBrainAu-toEncoder	✗	Channelwise AlexNet	Correct prediction of rearranged incomplete input channels.
Gidaris et al. [11]	2018	RotNet	✗	AlexNet	Correct prediction of rotated images.
Oord et al. [27]	2018	CPC	✓	ResNet-v2–101	Using contrastive predictive coding in PixelCNN auto-regressive recurrent neural networks for prediction of output embedding vectors.
Caron et al. [28]	2020	SwAV	✓	ResNet-50	Instead of pairwise contrastive loss, an online clustering of multiple views of same image is performed to learn the features and cluster assignments.
Chen et al. [29]	2020	iGPT-XL	✗	GPT-2 BERT	Auto-regressive prediction of pixels using transformers.
Chen et al. [10]	2020	SimCLR	✓	ResNet-50	A simple approach based on contrastive loss, non-linearity layer, augmentations and large batch sizes.
He et al. [14]	2020	MoCo	✓	ResNet-50	An online dictionary approach for contrastive learning using memory bank and momentum contrast.
Grill et al. [30]	2020	BYOL	✓	ResNet-50	Does not use negative pairs for contrastive loss while using momentum contrast.
Chen et al. [5]	2021	SimSiam	✓	ResNet-50	Uses a simple Siamese network without negative pairs, large batch size, momentum contrast.
Caron et al. [23]	2021	DINO	✓	ViT-S/16	Contrastive learning on vision transformers using a codistillation approach.
Goyal et al. [18]	2021	SEER	✓	RegNet-Y	SwAV method is used to train large SSL model on very large dataset in the wild.
Dwibedi et al. [4]	2021	NNCLR	✓	ResNet-50	Uses a nearest neighbor approach to increase the semantic variation during learning.
Xie et al. [31]	2022	SimMIM	✗	ViT-B	Correct prediction of patch level masked images using transformers.
Yeh et al. [2]	2022	Decoupled CLR	✓	ResNet-50	Remove the positive term from the denominator of the InfoNCE loss to reduce the positive–negative-coupling.
Zhang et al. [32]	2023	ADCLR	✓	ViT-S/16	Transformer based approach for dense contrastive learning by balancing global and patch-level losses.
Hou et al. [22]	2023	SBCL	✓	ResNet-50	A hierarchical online clustering like SwAV to balance the emphasis between head class and long-tailed class.

270 degrees. Rotation turned out to be a simple yet powerful SSL strategy since it does not leave any easily detectable low-level visual shortcut for trivial feature learning. Chen et al. adapted a GPT-2 scale transformer model from Masked Language Modeling (MLM) to Masked Image Modeling (MIM) on pixels of down-scaled images [29]. Objective of the pretext task was to auto-regressively predict the masked pixels in the transformer output in a BERT-like sense. Following the same line, Zhou et al. introduced an online visual tokenizer for MIM [36]. They showed that better semantics could be learned by simultaneously training the tokenizer with the MIM transformer through knowledge distillation. Xie et al. simplified the previous transformer based masked prediction pretext task methods by dropping blockwise masking and tokenization [31]. Their model achieved competitive results with just linear probing.

2.2. Contrastive SSL methods

Although, pretext based methods achieved good representations, however, Misra et al. showed that they all followed a covariant style of modeling [37]. Misra et al. advocated the superiority of an invariant style of modeling over a covariant. Their work sits between non-contrastive and contrastive SSL methods. The main contribution is the noise contrastive estimation formulation which involves the generation of positive and negative pairs using the Jigsaw objective [12]. Contrastive SSL methods differ in their approach by including the pretext

task in the model architecture itself in the form of augmentations. Additionally, the model objective changes from equivariance (where the model tries to adjust itself according to the variation in input) to invariance (where the model ignores the changes in the input in order to become agnostic to those transformations). To be specific, the SSL supervision signal is derived by enforcing the equivalence of multiple views of the same input image [10]. One such initial work by Oord et al. proposed Contrastive Predictive Coding (CPC) for unsupervised representation learning [27]. They applied noise contrastive loss (NCL) over future predictions in latent space of auto-regressive models for speech, text, and images. An important aspect of their NCL formulation was inclusion of negative samples. Later it was picked up and improved by Chen et al. [10]. They proposed a simple contrastive learning approach in which they emphasized on compositions in data augmentation, role of non-linear transformations in top layers, and larger batch size. He et al. proposed the idea of dictionary look-up by maintaining a dictionary of encoded keys as negatives for contrastive learning [14]. This allowed keeping a larger set than the batch size as negatives. Similar to He et al. Girll et al. proposed online and target network based contrastive learning where the target network avoids direct gradient flow and takes updates from the online network [30]. They also showed that the setup does not require negative samples for training the network. To reduce the overall computation in contrastive SSL methods, Caron et al. proposed to avoid pairwise comparisons by employing online clustering of the representations and by enforcing consistency in cluster assignments of representations corresponding to different views of the same sample [28]. Chen et al. refined the contrastive SSL aspects of previous works [5]. They trained a Siamese network with contrastive loss but without negative sample pairs and without large batch sizes. They avoided trivial solutions and attained competitive performance by avoiding gradient propagation in one of the branches of the Siamese network. Caron et al. proposed a self-supervised knowledge distillation approach called DINO [23]. They showed that vision transformers learn better semantic segmentation and k-NN features than CNNs. Goyal et al. validated the contemporary contrastive SSL approaches in the wild by training with one billion random images [18]. Yeh et al. proposed decoupling the positive samples from the denominator of the InfoNCE contrastive loss to remove the negative–positive-coupling effect in contrastive SSL methods [2]. Zhang et al. introduced query patches for contrasting in addition to global contrasting [32]. Nearest neighbor based methods like [4,22] emphasized increasing semantic variation by sampling the nearest neighbors in the latent space. The proposed work is similar in spirit to the nearest neighbor sampling based works.

3. Method

In this section, we first formalize the representation learning problem. After this, the NNCLR approach is described, and finally, we present the details of the proposed method.

3.1. Problem

Representation learning aims to learn a model which can map its inputs to corresponding vectors in such a way that for any two closely related inputs, their vectorial representations are also close, and far away for any two unrelated or distinctly related inputs. For a given dataset $\mathcal{X}$ having images $x_i | i \in [1, N]$, we wish to learn a homomorphism, model $\mathcal{M}_\theta$ parameterized by θ, such that for any three inputs x_i, x_j and x_k, $\mathcal{M}_\theta$ gives three corresponding vectors $\boldsymbol{u}_i$, $\boldsymbol{u}_j$, and $\boldsymbol{u}_k$ respectively, such that the following condition holds:

$$d_x(x_i, x_j) \star d_x(x_i, x_k) \Leftrightarrow d_v(\boldsymbol{u}_i, \boldsymbol{u}_j) \star d_v(\boldsymbol{u}_i, \boldsymbol{u}_k), \tag{1}$$

where, d_x and d_v represent the distance function in image and vector spaces respectively, and $\star$ is any relational operator like $\ll$.

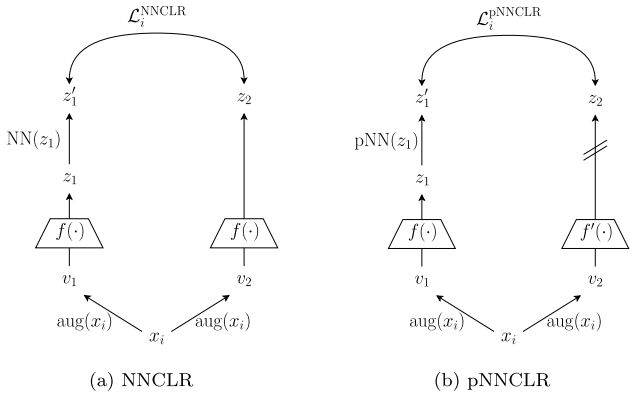


Fig. 1. Model diagrams of NNCLR and pNNCLR methods. The cross sign in pNNCLR denotes stop-gradient operation. $f(\cdot)$ is the backbone encoder network. $\text{aug}(x_i)$ is a random transformation function that generates a new view for x_i. $\text{pNN}(\cdot)$ is the proposed pseudo nearest neighbor sampling function.

3.2. NNCLR

Contrastive learning methods like SimCLR [10] or BYOL [30] train by generating two augmented views v_1, v_2 for the same input x_i. During the loss calculation, embeddings corresponding to v_1, v_2 are treated as positives, whereas embeddings corresponding to all other $x_j | j \neq i$ are treated as negatives to v_1, v_2. A variant of InforNCE loss [27] like

$$\mathcal{L}_i^{SimCLR} = -\log \frac{\exp(z_i \cdot z_i^+ / \tau)}{\sum_{k=1}^{n} \exp(z_i \cdot z_k^+ / \tau)}, \tag{2}$$

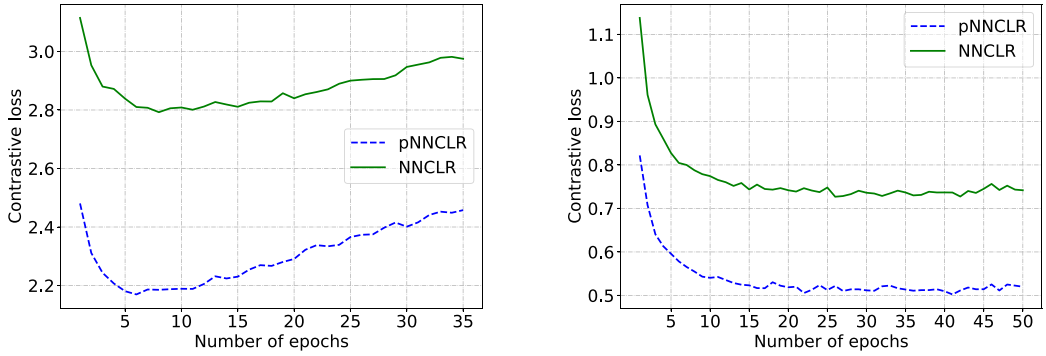
is used, where z_i is the embedding or the vector corresponding to the view v_1, z_i^+ is the positive pair of z_i. The set, $z_k^+ | k \in [1, n]$, denotes all embeddings in the mini-batch (with size n), including the positive z_i^+ and negatives $z_k^- | k \neq i$. τ denotes the softmax temperature. The operation, $\boldsymbol{u} \cdot \boldsymbol{v}$ in Eq. (2), represents a similarity function, generally a dot product of the normalized vectors $\boldsymbol{u}$, $\boldsymbol{v}$ or their cosine similarity. NNCLR improves this approach by replacing z_i with its nearest neighbor, $\text{NN}(z_i)$, as shown in Fig. 1(a). The nearest neighbor is found from a support set Q, which is maintained by inserting the current batch items and removing the oldest batch items from it in a first-in-first-out manner for every training iteration. Using $\text{NN}(z_i)$, NNCLR loss function for $x_i \in \text{batch}\{x_k | 1 \leq k \leq n\}$ becomes:

$$\mathcal{L}_i^{NNCLR} = -\log \frac{\exp(\text{NN}(z_i) \cdot z_i^+ / \tau)}{\sum_{k=1}^{n} \exp(\text{NN}(z_i) \cdot z_k^+ / \tau)}. \tag{3}$$

3.3. pNNCLR

Although the intuition of semantic variability behind NNCLR has shown promising results compared to other recent developments in contrastive learning methods [4], however, NNCLR achieves this at a cost, since the probability of finding a hard nearest neighbor, from the support set Q, belonging to the same class is quite low ($\sim 50\%$) at the beginning of training (details in Appendix A). I.e., if $\text{class}(\cdot)$ denotes the class membership function, the approximate (since it depends on the support set size) maximal probability, $P[\text{class}(\text{NN}(z_i)) = \text{class}(z_i)]$, that the nearest neighbor belongs to the same class as z_i or x_i is around 50% (details in Appendix A), which means there is a 50% chance that the nearest neighbor is from a different class. This reduces the inter-class variation of the representations, leading to a decline in performance. This was also seen in the NNCLR method's loss plots (Fig. 2). NNCLR incurs a higher loss in the beginning of training.

If we carefully investigate the intuition behind NNCLR, we will find that it is trying to increase the semantic variability between the two views v_1, v_2. However, by doing so, it is also dispersing the intra-class representations to have a larger mean deviation. These



(a) Tiny-imagenet (b) STL-10

Fig. 2. Loss plots of NNCLR and pNNCLR methods on Tiny-imagenet and STL-10 datasets. Note, NNCLR incurs a higher loss right from the beginning of training.

two ideas are inversely related. To overcome this trade-off, this work proposes to use soft or pseudo nearest neighbor function, pNN(·), in place of NN(·), to perform better irrespective of the probability $P[\text{class}(\text{NN}(z_i)) = \text{class}(z_i)]$. The proposed method is called, pNNCLR, pseudo/probabilistic nearest neighbor CLR (Fig. 1(b)). Function pNN(·), works by sampling a point z_i'' in the direction of the vector $\overrightarrow{z_i\ \text{NN}(z_i)}$ such that the resultant vector $\overrightarrow{z_i\ z_i''}$ has a shorter magnitude than $\overrightarrow{z_i\ \text{NN}(z_i)}$. This shortness is controlled by a scalar hyperparameter $\alpha \in (0, 1)$ as:

$$z_i'' \leftarrow z_i + (1 - \alpha)(\text{pNN}(z_i) - z_i). \tag{4}$$

While the probability $P[\text{class}(\text{NN}(z_i)) = \text{class}(z_i)]$ improves by using z_i'' over z_i, some semantic variability is lost. To dilute this effect, we found that we can stochastically resample in the vicinity of z_i''. This is done by using a Gaussian prior with mean z_i'' and standard deviation which is a fraction, β, of $\|\overrightarrow{z_i\ z_i''}\|$, where $\|\cdot\|$ denotes magnitude of a vector. This is shown in Eq. (5).

$$z_i' \sim \mathcal{N}(z_i'', \beta\|\overrightarrow{z_i\ z_i''}\|), \tag{5}$$

where, $\mathcal{N}$ stands for a normal distribution, and $\beta \in (0, 1)$ is a scalar constant. Due to higher uncertainty in NN based approach, we slow down the weight updation process of the encoder network $f'(\cdot)$, shown in Fig. 1(b), by stopping the gradient flow in non pNN(·) branch (also called the target branch [30]). Providing a smoother updation of weights by using following:

$$\theta_{f'} \leftarrow \lambda\theta_{f'} + (1 - \lambda)\theta_f, \tag{6}$$

where, θ stands for network parameters, f is the online network, f' is the target network, $\lambda \in (0, 1)$ is a constant that controls the effect of f over f'. By replacing the nearest neighbor function in Eq. (3), we obtain the loss for pNNCLR, as:

$$\mathcal{L}_i^{pNNCLR} = -\log \frac{\exp(\text{pNN}(z_i) \cdot z_i^+/\tau)}{\sum_{k=1}^{n} \exp(\text{pNN}(z_i) \cdot z_k^+/\tau)}, \tag{7}$$

using this, the loss for the entire mini-batch, b, of size N_b becomes:

$$\mathcal{L}_b^{pNNCLR} = \frac{1}{N_b} \sum_{i=1}^{N_b} \mathcal{L}_i^{pNNCLR}. \tag{8}$$

The total loss, $\mathcal{L}^{pNNCLR}$, is a symmetrized loss obtained by swaping the views v_1, v_2 in Eq. (8), as:

$$\mathcal{L}^{pNNCLR} = \mathcal{L}_b^{pNNCLR}(v_1, v_2) + \mathcal{L}_b^{pNNCLR}(v_2, v_1). \tag{9}$$

4. Experiments

This section first describes the experimental arrangement. Next, the implementation and dataset related details are presented. After this, the results of the proposed pNNCLR approach are compared with the recent methods for SSL for the linear evaluation task. Towards the end, some ablations, discussion on results, and future directions are presented.

4.1. Implementation details

We have used the batch size of 64, and 10000 as the size of the support set. The embedding size was kept at 2048. The optimizer was Adam, and the learning rate was set to 0.001. The images for every dataset were resized to 96 × 96. ResNet-50 [8] was used as the base encoder network. The final prediction layer of ResNet was removed, and a Global average pooling was used for flattening, followed by two dense layers having 2048 nodes, and a batch norm was present between these two dense layers as shown in Fig. 3, encoder network. During testing, the encoder was frozen after the fine-tuning, and an additional dense layer having the softmax activation was used for classification (linear probing), as shown in Fig. 3, classification network. Five non-medical datasets (STL-10, Cifar-10,100, Tiny-imagenet, Pascal-VOC) and three medical datasets (Blood-MNIST, PCAM, Path-MNIST) were used for evaluation and comparison purposes. Details of these datasets and their splitting strategy for training and testing purposes is provided in the next section. Top-1 accuracy was the metric used for all our experiments unless stated otherwise.

4.2. Datasets

STL-10. It is a standard dataset derived from Imagenet [38] for developing self-supervised learning algorithms. It has 100000 unlabeled and 13000 labeled images from 10 classes (like bird, cat, truck) [39]. All models reported here, were trained for 100 epochs on this dataset.

Cifar-10 and Cifar-100. Each of these datasets consists of 60000 images [40]. Cifar-10 consists of 10 classes, whereas Cifar-100 consists of 100 classes. General class labels are bird, dog, ship, horse, truck, etc.

Tiny-imagenet. This dataset contains 120000 images from 200 classes [41].

Pascal-VOC. This dataset contains 20 classes like vehicles, airplanes, animals, etc. It contains approximately 3000 images.

Blood-MNIST and Path-MNIST. Both of these datasets belong to a large collection of biomedical images [42]. Blood-MNIST has approximately 17000 images belonging to 8 classes from blood cell microscopy. Path-MNIST has 9 classes having approximately 100000 images of Colon pathology.

PCAM or PatchCamelyon. It is a binary image classification dataset [43] having approximately 327000 images extracted from histopathologic scans of lymph node sections to indicate the presence of metastatic tissue.

Features learned from STL-10 were used to apply transfer learning to other datasets. Approximately ∼ 2% of the images were used for applying transfer learning using a linear layer on the pretrained encoder model.

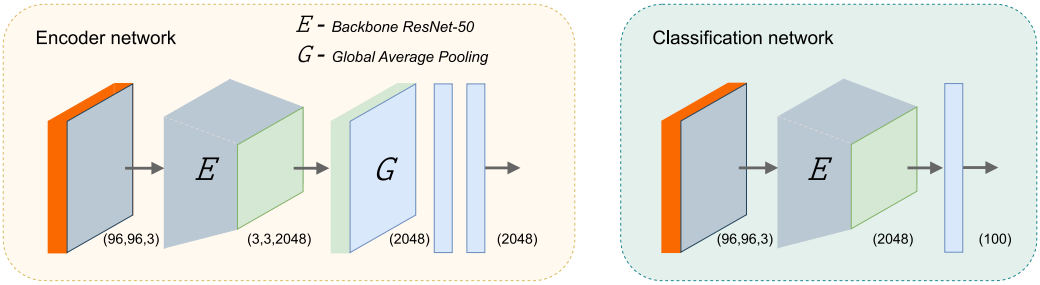


Fig. 3. Proposed pNNCLR architecture details considering Cifar-100 as the downstream task. Left, architecture for contrastive SSL training. Right, linear probing adaptation on Cifar-100 dataset.

Table 2
Results are shown for Top-1 accuracy (↑) on the non-medical datasets on the image recognition task. For each dataset, **best** method is marked in bold, second best is underlined.

Method	STL-10	Cifar-10	Cifar-100	Tiny-imagenet	Pascal-VOC	Mean (Top-1 acc.)
Baseline (NNCLR [4])	0.7548	0.9441	0.7763	0.3929	–	0.7170
MoCo v2 [14]	0.8355	0.9411	0.7804	0.4651	0.4900	0.7555
BYOL [30]	0.8044	0.9456	0.7882	0.4577	–	0.7489
SimCLR [10]	0.7974	0.9428	0.7812	0.4660	–	0.7468
SimSiam [5]	0.8067	0.9418	0.7811	0.3284	–	0.7145
DINO [23]	0.8200	0.9459	0.7809	0.3161	–	0.7157
Decoupled CLR [2]	0.8343	0.9498	0.7812	0.4245	–	0.7474
Proposed method (pNNCLR)	**0.8413**	**0.9582**	**0.7885**	**0.4856**	**0.5066**	**0.7684**

4.3. Results

To evaluate the performance of the proposed method, it is compared with very competitive recent benchmarked self-supervised learning works. The results are presented in Tables 2 and 3 for five non-medical and three medical type of datasets, respectively. Our baseline work is the NNCLR method [4], which was published in ICCV 2021. NNCLR extends SimCLR [10] by introducing the idea of sampling the nearest neighbor instead of the other view, in the latent space. MoCo [14] and BYOL [30] are momentum contrast based approaches for contrastive learning, and appeared in CVPR 2020 and NeurIPS 2020 respectively. SimCLR [10] is the baseline approach of NNCLR, and was published in PMLR 2020. SimSiam [5] further extended SimCLR by showing that even without using the negative samples, good performance could be achieved in SSL. It was published in CVPR 2021. DINO [23] is a vision transformer based method and was published in ICCV 2021. Decoupled CLR [2] removed the positive-loss terms from the denominator of the InfoNCE loss [27] and proposed a simple CL method which was published in ECCV 2022.

We have used Top-1, Top-3, Top-5, F1-score and Recall as the metrics for performance evaluation of the proposed method in comparison to other works on eight different datasets as explained in the previous paragraph. Furthermore, accuracy plots and T-SNE plots are also shown to provide a more diverse understanding of the performance of the proposed method. Now, in the following paragraphs, we provide an analysis of the results.

Table 2 presents the results for non-medical datasets using Top-1 accuracy metric. It can be seen that the proposed method, pNNCLR, achieved the highest Top-1 accuracy for each of the datasets among all methods. It surpassed the baseline, NNCLR, by a maximum ∼ 8% on STL-10 and Tiny-imagenet datasets, and by ∼ 4% on average over all other datasets. The second best performance was attained by MoCo [14], which is ∼ 1% less than pNNCLR, on average. Whereas, a more recently proposed method, called DCLR [2], also performed comparably with the second best method MoCo [14], achieving a superior performance on Cifar-10 and Cifar-100 datasets.

Table 3 presents the results for medical datasets using Top-1 accuracy metric. For the Blood-MNIST dataset, MoCo [14] attained the best results with an accuracy of 89.06% while the proposed pNNCLR method lagged behind by ∼ 2%; however, the proposed pNNCLR method performed ∼ 7% better than the baseline NNCLR [4] and ∼ 4% better than DCLR [2], a more recent variant of SimCLR [10]. On the PCAM dataset, again the best performance was attained by MoCo [14] while pNNCLR lagged behind by only ∼ 1%, however, the proposed pNNCLR method did attain the second best performance. On the Path-MNIST dataset, pNNCLR attained the best results while the second best performance was attained Decoupled CLR [2] method. On average, pNNCLR lagged behind by the best performance by less than 1% while surpassing the baseline NNCLR by ∼ 4%. Overall, no method had a clear winning. Both MoCo [14] and proposed pNNCLR performed comparatively, while pNNCLR completely surpassed the performance of baseline NNCLR [4] on all three medical datasets.

Table 3
Results are shown for Top-1 accuracy on the medical datasets on the image recognition task. For each dataset, **best** method is marked in bold, second best is underlined.

Method	Blood-MNIST	PCAM	Path-MNIST	Mean (Top-1 acc.)
Baseline (NNCLR [4])	0.7969	0.8849	0.8292	0.8370
MoCo v2 [14]	**0.8906**	**0.9180**	0.8562	**0.8882**
BYOL [30]	0.8516	0.8868	0.8365	0.8583
SimCLR [10]	0.8594	0.8951	0.8552	0.8699
SimSiam [5]	0.8594	0.8397	0.7917	0.8302
DINO [23]	0.7500	0.7824	0.7698	0.7674
Decoupled CLR [2]	0.8281	0.8884	0.8615	0.8593
Proposed method (pNNCLR)	0.8672	0.9025	**0.8708**	0.8801

Table 4 presents the results for non-medical and medical datasets combined, using Top-5 accuracy metric. It can be seen that the proposed pNNCLR method achieves best Top-5 accuracy on all datasets among all methods. For STL-10 dataset, pNNCLR's performance is comparable to the MoCo [14], DINO [23], and DCLR [2] methods. However, pNNCLR achieves notably better performance on Cifar-10 and Cifar-100 datasets than its close competitors MoCo [14] and DCLR [2]. On Tiny-imagenet dataset, proposed pNNCLR method achieved ∼ 6% superior performance than the baseline NNCLR [4] method. On Blood-MNIST dataset, DCLR [2], MoCo [14] and the proposed pNNCLR methods achieved 100% Top-5 accuracy, whereas, on Path-MNIST dataset, SimCLR [10] and proposed pNNCLR achieved 100% Top-5 accuracy. The PCAM dataset was not included in Top-5 performance comparison because it is a binary dataset and Top-5 metric cannot be evaluated for it.

Table 4
Results are shown for Top-5 accuracy on all datasets on the image recognition task. For each dataset, **best** method is marked in bold. PCAM dataset is not included since it is a binary classification dataset.

Method	STL-10	Cifar-10	Cifar-100	Tiny-imagenet	Blood-MNIST	Path-MNIST
Baseline (NNCLR) [4]	0.9905	0.9890	0.9405	0.6559	0.9921	0.9979
MoCo v2 [14]	0.9917	0.9887	0.9439	0.7000	1.0000	0.9979
BYOL [30]	0.9912	0.9892	0.9503	0.6955	0.9843	0.9979
SimCLR [10]	0.9911	0.9889	0.9445	0.7006	0.9921	1.0000
SimSiam [5]	0.9913	0.9888	0.9445	0.6164	0.9743	0.9947
DINO [23]	0.9915	0.9893	0.9443	0.6089	0.9843	0.9927
Decoupled CLR [2]	0.9917	0.9897	0.9445	0.6752	1.0000	0.9989
Proposed method (pNNCLR)	**0.9918**	**0.9908**	**0.9506**	**0.7126**	**1.0000**	**1.0000**

Table 5
Results are shown for Top-3 accuracy on all datasets on the image recognition task. For each dataset, **best** method is marked in bold. Since Cifar-100 and Tiny-imagenet datasets have 100 or more classes, therefore, Top-10 accuracy is reported for them.

Method	STL-10	Cifar-10	Cifar-100	Tiny-imagenet	Blood-MNIST	Path-MNIST
Baseline (NNCLR) [4]	0.9527	0.9700	0.9826	0.7573	0.9765	0.9822
MoCo v2 [14]	0.9702	0.9689	0.9854	0.7918	0.9531	0.9854
BYOL [30]	0.9635	0.9706	0.9906	0.7883	0.9531	0.9760
SimCLR [10]	0.9620	0.9695	0.9859	0.7922	0.9609	**0.9875**
SimSiam [5]	0.9640	0.9691	0.9858	0.7264	0.9531	0.9843
DINO [23]	0.9669	0.9707	0.9857	0.7205	0.9296	0.9583
Decoupled CLR [2]	0.9699	0.9722	0.9859	0.7724	**0.9765**	0.9854
Proposed method (pNNCLR)	**0.9715**	**0.9755**	**0.9908**	**0.8016**	0.9687	0.9833

Table 6
Results are shown for F1-score on all datasets on the image recognition task. For each dataset, **best** method is marked in bold.

Method	STL-10	Cifar-10	Cifar-100	Tiny-imagenet	Blood-MNIST	PCAM	Path-MNIST
Baseline (NNCLR) [4]	0.7553	0.9400	0.7919	0.3857	0.7932	0.9115	0.8275
MoCo v2 [14]	0.8356	0.9370	0.7962	0.4677	**0.8786**	**0.9501**	0.8517
BYOL [30]	0.8046	0.9414	0.8045	0.4593	0.8558	0.9137	0.8366
SimCLR [10]	0.7977	0.9387	0.7971	0.4688	0.8492	0.9234	0.8454
SimSiam [5]	0.8069	0.9377	0.7970	0.3124	0.8367	0.8588	0.8014
DINO [23]	0.8202	0.9417	0.7967	0.2984	0.7196	0.7920	0.7543
Decoupled CLR [2]	0.8344	0.9455	0.7971	0.4216	0.8180	0.9156	0.8598
Proposed method (pNNCLR)	**0.8413**	**0.9537**	**0.8048**	**0.4910**	0.8561	0.9321	**0.8572**

Table 7
Results are shown for Recall on all datasets on the image recognition task. For each dataset, **best** method is marked in bold.

Method	STL-10	Cifar-10	Cifar-100	Tiny-imagenet	Blood-MNIST	PCAM	Path-MNIST
Baseline (NNCLR) [4]	0.7673	0.9625	0.8206	0.3604	0.7109	0.8918	0.7802
MoCo v2 [14]	0.8957	0.9573	0.8294	0.4864	**0.8281**	0.9070	0.8187
BYOL [30]	0.8462	0.9651	0.8461	0.4735	0.7734	0.8845	0.8135
SimCLR [10]	0.8351	0.9602	0.8311	0.4879	0.8125	0.8998	0.8166
SimSiam [5]	0.8498	0.9585	0.8309	0.2478	0.7991	0.8228	0.7385
DINO [23]	0.8710	0.9656	0.8304	0.2264	0.6250	0.7914	0.6708
Decoupled CLR [2]	0.8938	0.9723	0.8311	0.4155	0.7890	0.8734	**0.8312**
Proposed method (pNNCLR)	**0.9049**	**0.9868**	**0.8467**	**0.5221**	0.8125	**0.9280**	0.8270

Table 5 presents the results for non-medical and medical datasets combined, using Top-3 accuracy metric. Proposed pNNCLR method achieved best accuracy on STL-10 dataset, surpassing the baseline NNCLR [4] by ~ 2%. The proposed pNNCLR method achieved the best performance on Cifar-10, Cifar-100 and Tiny-imagenet datasets as well; exceeding the baseline NNCLR [4] by ~ 5% on Tiny-imagenet dataset. On the Blood-MNIST dataset, DCLR [2] method performed best surpassing the proposed pNNCLR method by ~ 1%, however, the pNNCLR method achieved the second best Top-3 accuracy score. SimCLR [10] achieved best performance on the Path-MNIST dataset, however, its score was comparable with MoCo [14] and DCLR [2] methods. Overall, the proposed pNNCLR method achieved better performance than baseline NNCLR [4] on all datasets. However, pNNCLR performed better for non-medical datasets than medical datasets. The PCAM dataset was not included in Top-3 performance comparison because it is a binary dataset and Top-3 metric cannot be evaluated for it.

Table 6 presents the results for non-medical and medical datasets combined, using F1-score. F1-score is a common metric for evaluation model's performance. The proposed method achieved ~ 10% better performance than the baseline NNCLR [4] on STL-10 and Tiny-imagenet datasets. The proposed method achieved best performance on STL-10, Cifar-10, Cifar-100 and Tiny-imagenet datasets, while MoCo [14] and DCLR [2] showing comparative performance. On Blood-MNIST and

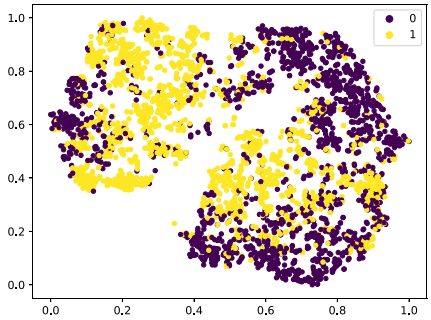
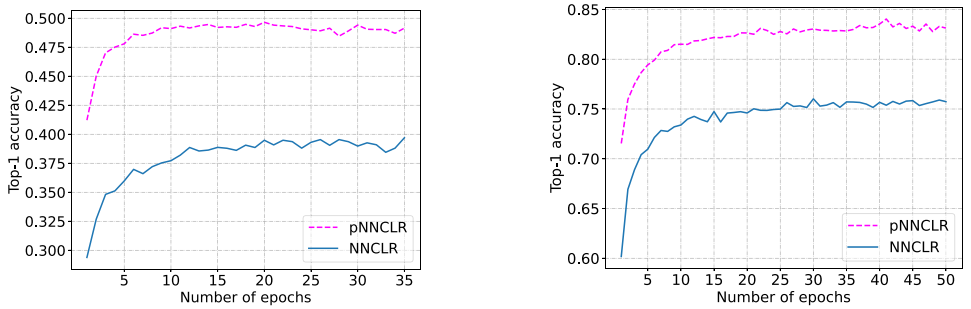
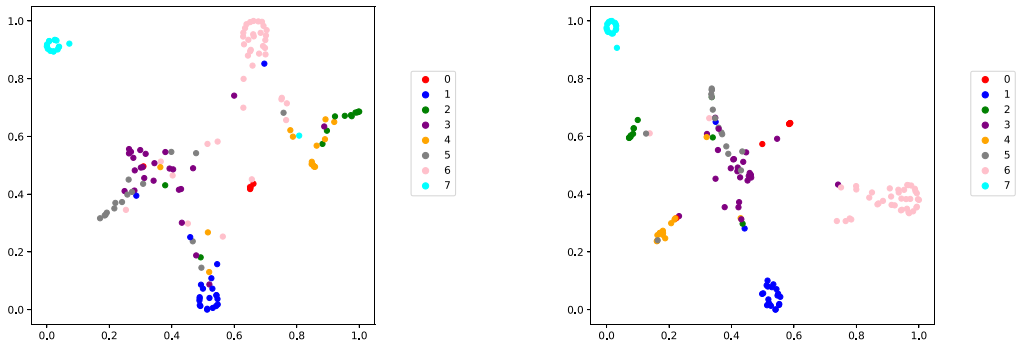


Fig. 4. T-sne plot of the representations learned by the proposed pNNCLR method on the PatchCamelyon (PCAM) medical dataset.



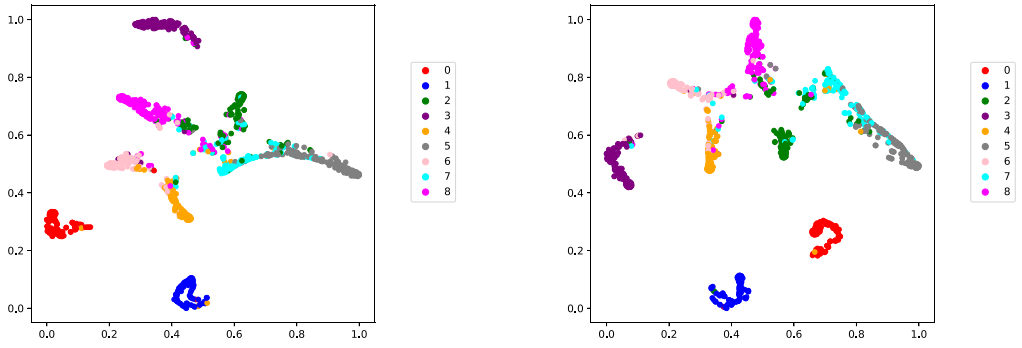
(a) Tiny-imagenet (b) STL-10

Fig. 5. Top-1 accuracy plots of the baseline NNCLR and proposed pNNCLR methods on Tiny-imagenet and STL-10 datasets.



(a) NNCLR (b) pNNCLR

Fig. 6. T-sne plot of the representations learned by the baseline NNCLR and proposed pNNCLR methods on the Blood-MNIST medical dataset.



(a) NNCLR (b) pNNCLR

Fig. 7. T-sne plot of the representations learned by the baseline NNCLR and proposed pNNCLR methods on the Path-MNIST medical dataset.

PCAM datasets, MoCo [14] achieved best performance. However, on Path-MNIST medical dataset, the proposed pNNCLR method achieved best performance exceeding the baseline NNCLR [4] by ~ 3%.

Table 7 presents the results for non-medical and medical datasets combined, using Recall score. It can be seen that the recall score of the proposed pNNCLR method is quite high in comparison to the baseline NNCLR [4] and it is the best among all methods on all four non-medical datasets. On the Blood-MNIST dataset, MoCo [14] performs best by attaining a Recall value of 0.8281, while, SimCLR [10] and the proposed pNNCLR method achieved a comparable value of 0.8125. The proposed pNNCLR method performed best for the PCAM dataset attaining a Recall value of 0.9280. However, DCLR [2] method performed best on the Path-MNIST dataset attaining a Recall score of 0.8312, followed by a Recall score of 0.8270 attained by the proposed pNNCLR method. Overall, there was no clear winning method for the medical datasets, however, the proposed pNNCLR method performed quite competitively to other best performing methods like MoCo [14] and DCLR [2].

Next, we present the accuracy and T-SNE embedding plots to further analyze the performance of methods in comparison to the proposed pNNCLR method. Fig. 4 presents a low dimensional view of the representations learned by the proposed pNNCLR method on the PCAM dataset for the binary classification of the presence of metastatic tissues in lymph node scans. Although, pNNCLR achieved an accuracy of 90.25%, it can be noted that the two classes are separated from each other, however, there is some overlap of samples, which suggests that these could have created the misclassification for the proposed pNNCLR method. Fig. 5 shows the Top-1 accuracy plots of the proposed pNNCLR method vs. the baseline NNCLR [4], on the Tiny-imagenet and STL-10 datasets. It can be noted that the proposed pNNCLR method attains comparatively better performance than the baseline NNCLR [4] method, right from the beginning phase of training, on both of the

datasets. A possible cause for this behavior can be found in the basic arguments on relaxing the hard-nearest-neighbors due to the presence of higher uncertainty for nearest-neighbor based algorithms while they begin to train using the self-supervised loss. The same is also explained in the Section 3.3. While, for both NNCLR [4] and the proposed pNNCLR methods, the performance saturates asymptotically, pNNCLR attains a much better Top-1 performance than NNCLR [4]. Fig. 6 presents a low-dimensional view of the representations learned by the baseline NNCLR and the proposed pNNCLR methods for the Blood-MNIST datasets. This helps in having an approximate idea on how separable the representations learned by the methods are. We can see in Fig. 6(b), the representations learned by the proposed pNNCLR method are a bit more resolved and separately located than the representations of baseline NNCLR [4] method. Especially, for the class 1, 2, 4, the proposed pNNCLR method resolves better than the baseline NNCLR [4] method. Fig. 7 presents the T-SNE plots for the baseline NNCLR [4] and the proposed pNNCLR methods on the Path-MNIST medical dataset. It can be noted that the proposed pNNCLR methods does a better job on spreading the class representations towards outside than the baseline [4] method. Also, class 2 is better resolved by the pNNCLR method than NNCLR [4]. Also, the distance between the class 4 and 7 is higher in Fig. 7(b) than in Fig. 7(a).

Overall, these quantitative and qualitative results indicate that the proposed pNNCLR method performs notably better than the baseline NNCLR [4] method on all non-medical and medical datasets. The features learned by the proposed pNNCLR method were better resolved than the baseline NNCLR [4] method. An interesting result was that the proposed pNNCLR method performed much better than the other compared SSL methods on the non-medical datasets. While, for the medical datasets, there was no clearly winning method and the proposed pNNCLR method performed quite comparatively to other SSL methods on the medical datasets. MoCo [14] and DCLR [2] methods gave a strong competition to the proposed pNNCLR method. These methods performed comparably better than the rest of the SSL methods on both non-medical and medical datasets.

4.4. Ablations

This section presents the behavior or the change in behavior of the baseline NNCLR [4] and the proposed pNNCLR methods when the key training hyperparameters were varied. The ablation study was performed with the STL-10 and Tiny-imagenet datasets. In the first ablation, we examined the effect of each proposed modification to the baseline NNCLR [4] method. The results corresponding to these variations is reported in the Table 8 corresponding to the baseline NNCLR [4] and the proposed pNNCLR methods. Before discussing the results, we want to define the expansions of key modifications — *swu* denotes smooth weight updation, *pNN* denotes pseudo nearest neighbor sampling, and *noise* denotes the operation of stochastic sampling by adding noise. Referencing to the Table 8, modifying baseline NNCLR [4] with *swu*, improved the accuracy by ∼ 7% on STL-10 and by ∼ 8% on Tiny-imagenet datasets. Modifying baseline NNCLR [4] with (*swu+pNN*), improved the accuracy further by ∼ 1% to 83.86% in STL-10 and 48.42% in Tiny-imagenet datasets. Modifying baseline NNCLR [4] with (*swu+pNN+noise*) further improved the accuracy to 84.13% on the STL-10 dataset and to 48.56% on Tiny-imagenet dataset. These results show that *swu* significantly stabilizes the uncertainty in the learning process, while the pseudo nearest neighbor sampling strategy provides better performance than the hard nearest neighbor sampling.

Table 9 presents variation in the hyperparameters α — the pseudo sampling control hyperparameter, and β — the noise control hyperparameter, on the STL-10 and Tiny-imagenet datasets. A value of 0.10 provided a better result over $\beta = 0.05$ by improving the Top-1 accuracy by ∼ 1%. Similarly, $\alpha = 0.25$ provided the best results for STL-10 dataset, however, for Tiny-imagenet dataset $\alpha = 0.15$ gave the best results. Note that α hyperparameter was ablated in the absence of β to monitor its sensitivity independently. A value of $\alpha = 0.25$ was chosen for all experiments.

Tables 10–12, provide results of the ablation study on the embedding size, support-set size and batch size. Comparing the baseline NNCLR [4] method with the proposed pNNCLR method on STL-10 and Tiny-imagenet datasets. Table 10 shows that the performance of the proposed method increases as the embedding vector size increases from 512 to 4096, on the STL-10 dataset. On the other hand, the baseline NNCLR [4] method's performance maxes out at embedding size 2048. Similarly, on the Tiny-imagenet dataset, the performance of the baseline NNCLR [4] method maxes out at embedding size 2048, however, the performance of the proposed pNNCLR method keeps improving as

Table 8
Effect of modifications in the baseline NNCLR method with smooth-weight-update denoted as (swu), pseudo neighborhood as (pNN), and addition of noise in sampling. Top-1 accuracy is reported for each experiment on the STL-10 and Tiny-imagenet datasets.

Method	STL-10	Tiny-imagenet
Baseline (NNCLR [4])	0.7548	0.3929
pNNCLR (swu)	0.8257	0.4715
pNNCLR (swu + pNN)	0.8386	0.4842
pNNCLR (swu + pNN + noise)	0.8413	0.4856

Table 9
Variation in the hyperparameters (β and α) of the proposed pNN(·) (pseudo nearest neighbor) sampling approach, is reported on the STL-10 and Tiny-imagenet datasets.

	Top-1 accuracy (↑)	
β	STL-10	Tiny-imagenet
0.05	0.8376	0.4820
0.10	0.8413	0.4856
α	STL-10	Tiny-imagenet
0.05	0.8286	0.4734
0.10	0.8295	0.4736
0.15	0.8321	0.4892
0.25	0.8386	0.4787

Table 10
Effect of using different embedding sizes on the baseline and the proposed method. Top-1 accuracy is reported on the STL-10 and Tiny-imagenet datasets.

Embedding size	STL-10		Tiny-imagenet	
	NNCLR [4]	Proposed method	NNCLR	Proposed method
512	0.6061	0.7986	0.3345	0.4367
1024	0.6580	0.8143	0.3634	0.4598
2048	0.7548	0.8413	0.3929	0.4856
4096	0.7537	0.8429	0.3918	0.4886

Table 11
Effect of using different support-set or queue sizes. Top-1 accuracy is reported for the baseline and the proposed method on STL-10 and Tiny-imagenet datasets.

Queue size	STL-10		Tiny-imagenet	
	NNCLR [4]	Proposed method	NNCLR	Proposed method
5000	0.7191	0.8329	0.3668	0.4434
10000	0.7548	0.8413	0.3929	0.4856
15000	0.7540	0.8407	0.3920	0.4812
20000	0.7555	0.8411	0.3927	0.4815

Table 12
Effect of using different batch sizes. Top-1 accuracy is reported for the baseline and the proposed method on STL-10 and Tiny-imagenet datasets.

Batch size	STL-10		Tiny-imagenet	
	NNCLR [4]	Proposed method	NNCLR	Proposed method
16	0.7051	0.7916	0.3402	0.4271
32	0.7141	0.8178	0.3534	0.4437
64	0.7548	0.8413	0.3929	0.4856

the embedding size grows. Motivated by these observations and to keep a common ground for the baseline and the proposed methods, we used 2048 as the embedding size in all our experiments.

Table 11 shows the effect of varying the queue or the support-set sizes on the proposed NNCLR [4] and the proposed pNNCLR methods. On the STL-10 dataset, the performance of the proposed method maxes out at a queue size of 10000. On the other hand, performance of the baseline method maxes out at 20000; however, it is comparable to its performance at 10000, and it drops at 15000. On the Tiny-imagenet dataset, the baseline NNCLR [4] and the proposed pNNCLR methods achieved the best performance at queue size 10000. Following these observations, we took 10000 as the size of the support set in all our experiments.

Table 12 presents the effect of varying the batch sizes for the baseline NNCLR [4] and the proposed pNNCLR methods on STL-10 and Tiny-imagenet datasets. On STL-10 dataset, it can be seen that the baseline NNCLR [4] method improves it performance linearly as the batch size is doubled. The same pattern can be observed for the proposed pNNCLR method which improves its performance from Top-1 accuracy of 79.16% at batch size 16 to 84.13% at batch size 64. On Tiny-imagenet dataset, a similar pattern is observed. Both, baseline NNCLR [4] and the proposed pNNCLR methods improved their Top-1 accuracy linearly as the batch size was doubled from 16 to 32 to 64. Overall, the ablation on batch size, shows that the proposed NNCLR [4] and the baseline pNNCLR methods achieved their best performance at a batch size of 64. Following this, 64 was used as the batch size in all our experiments. In the next section, we try to extend our understanding about the baseline NNCLR [4] and the proposed pNNCLR methods by providing more qualitative analysis.

4.5. Trade-offs between model accuracy and complexity

From the results, it is clear that the proposed pNNCLR method improves the accuracy upon the baselines NNCLR [4] and SimCLR [10]. However, it does add a small overhead to the computational cost. In this section, we analyze the complexity vs. accuracy of the proposed and the baseline methods. As the baseline and the proposed method share some common components, we can assume a common complexity for them. Such as, the encoder/model complexity C_{model} denotes the operations in the forward pass, including the execution of encoders along with linear layers and view generation. $C_{NN(z)}$, which is a common operation in NNCLR and pNNCLR, denotes the complexity of the operation of finding the nearest neighbor of z. $C_{aggregation}$, denotes the complexity of the aggregation operations of aggregating the gradients in the NNCLR or SimCLR method, or the weight updates in the pNNCLR in the non-gradient target branch. Note both — aggregation in NNCLR and weight updates in pNNCLR — bear the same time complexity, therefore, they are both denoted with $C_{aggregation}$. The time complexity of backpropagation through the encoder/model, $C_{backprop}$, is also same for NNCLR and pNNCLR. Using these, the time complexity for the NNCLR method for one iteration comes out to be:

$$C_{NNCLR} = C_{model} + C_{NN(z)} + 2 \times C_{backprop} + C_{aggregation}. \tag{10}$$

$$C_{pNNCLR} = C_{model} + O(1) + C_{NN(z)} + C_{backprop} + C_{aggregation}. \tag{11}$$

Eq. (11) provides the time complexity of the pNNCLR method for one iteration. From Eq. (10) and (11), we deduce that pNNCLR's pNN(·) operation (using Eq. (4) and (5)) incurs a very small $O(1)$ cost over NNCLR's NN(·) operation. However, it lowers the complexity by a factor of $C_{backprop}$ where $C_{backprop} \gg O(1)$. Hence, the overall complexity of the proposed pNNCLR method is lower than that of the baseline NNCLR method. In our experiments, we found that one epoch of SimCLR method takes around 65.875 s, one epoch of NNCLR method takes around 67.714 s, while one epoch of the proposed pNNCLR method takes around 84.857 s time. A possible reason for the longer running time of the pNNCLR method is that the weight update in pNNCLR method uses an unparallelized or non-vectorized implementation.

Thus, we observe that the proposed pNNCLR method delivers improved performance compared to the baseline NNCLR method while maintaining a similar or even slightly reduced complexity.

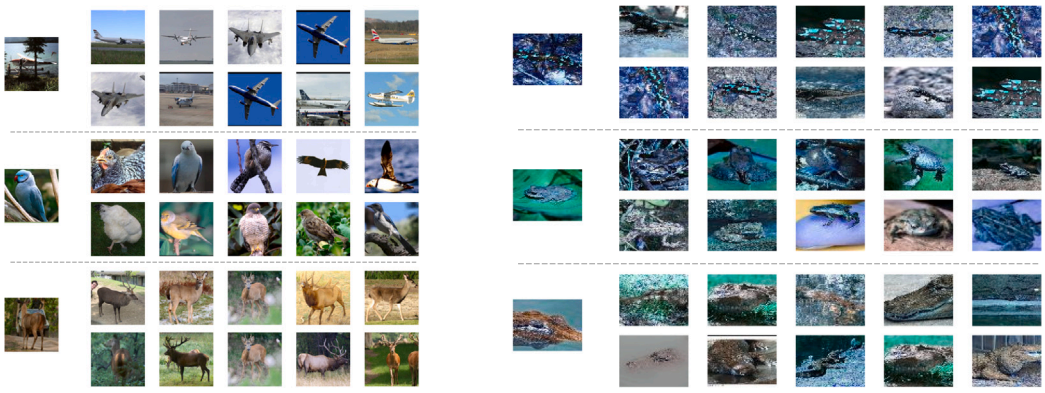
4.6. On the inconsistency of performance among baselines

In the literature, NNCLR [4] has been shown to perform better than SimCLR [10], however, in our experiments (Tables 2, 4 and 6), for some datasets, SimCLR has shown better performance than NNCLR. To investigate this issue, we repeated our experiments multiple times by training both SimCLR and NNCLR baselines for over 100 epochs, the results are reported in Table 13. A reduction in the difference between the mean accuracy of both methods was observed. On the STL-10 dataset, SimCLR performed better than NNCLR by $\sim$ 0.5% whereas NNCLR performed $\sim$ 1% better than SimCLR on the Tiny-imagenet dataset. These results indicated that NNCLR performs better than SimCLR when the classification problem has larger number of classes (such as $\sim$ 200 in Tiny-imagenet dataset).

4.7. Image retrieval task

We were interested in seeing the generic understanding capabilities of our self-supervised trained models. For this purpose, we designed an information retrieval task where we do not use the finetuning, rather we used the SSL trained models before any finetuning. The image retrieval task was composed of two image sets: the query set and the retrieval set. A query set was created by randomly choosing 5 images from the test set of a dataset. The retrieval set was created by randomly choosing 100 images from the train set of the corresponding dataset. For each query image, we asked the model to chose five images (which the model thinks are the most nearest ones to the query image) from the corresponding retrieval set, and then rank them in order of their closeness in the latent space w.r.t. the query image. The model runs a k-nearest neighbor algorithm for finding the first five nearest neighbors of the query image among all the images in the retrieval set using a fixed distance measure. The distance measure could be like Mean Squared Error (MSE) or KL-divergence.

Fig. 8 shows the results of image retrieval task for three query images on STL-10 and Tiny-imagenet datasets for baseline NNCLR [4] and the proposed pNNCLR methods. On STL-10 dataset, first query is about a small airplane. The retrieved results show that both baseline NNCLR [4] and proposed pNNCLR methods have a good understanding of objects, however, baseline NNCLR [4] fails to retrieve the exact same kind of plane in the top five retrievals. On the other hand, the proposed pNNCLR method retrieves the same model of airplane in its fifth retrieval. For the second query on a small bird, both methods fail to locate same kind of bird in their first retrieval. However, in comparison to the baseline NNCLR [4] method, the proposed pNNCLR method retrieves more such images that look and have similar size as the query image. For the third query on deer, both method perform equally well. On the Tiny-imagenet dataset, the first query is about some kind of lizard. It can be seen that both method retrieve good matching images, however, the retrievals done by the proposed pNNCLR method look more similar to the query image than the baseline NNCLR [4] method. The second query image is about a frog. Both methods do a good job by retrieving frog like objects in all five of their retrievals. However, the retrievals made by the proposed pNNCLR method look slightly better than the baseline NNCLR [4] method. The last query image is about some alligator. Again, it could be noticed that the images retrieved by the proposed pNNCLR method have more similar view point and shape as the query image than the baseline NNCLR [4] method. Overall, both methods did a good job on image retrieval task, however, the proposed pNNCLR method showed a slightly better performance.



(a) STL-10 (b) Tiny-imagenet

Fig. 8. The two sub-figures show the performance of the baseline NNCLR and proposed pNNCLR methods on image retrieval task for STL-10 and Tiny-imagenet datasets. There are three query images in the leftmost column for each dataset. For each query image there are two rows of retrieved images. The top row corresponds to baseline NNCLR and the bottom row corresponds to proposed pNNCLR method. The retrieved images in each row are ordered (from left to right) in their increasing retrieval rank, i.e., the model thinks that the one on the left is more closer to query image than the one on the right.

Table 13
Proposed pNNCLR is compared with the baseline NNCLR method using various metrics such as Top-1, F1-score, Recall and Top-5 accuracy for the STL-10 and Tiny-imagenet datasets in the image recognition task. Each experiment was repeated three times for each method, and their mean and standard deviation are reported.

Method	Experiment#	STL-10				Tiny-imagenet			
		Top-1	F1-score	Recall	Top-5	Top-1	F1-score	Recall	Top-5
SimCLR	1	0.7972	0.8063	0.8221	0.9954	0.4252	0.4527	0.4616	0.7619
	2	0.7881	0.7972	0.7909	0.9971	0.3924	0.4257	0.4238	0.7483
	3	0.7885	0.7971	0.7899	0.9917	0.4116	0.4395	0.4404	0.7539
	Mean	0.7913	0.8002	0.8010	0.9947	0.4097	0.4393	0.4419	0.7547
	Std Dev	0.0051	0.0052	0.0183	0.0027	0.164	0.0135	0.0189	0.0068
NNCLR	1	0.7896	0.7783	0.7891	0.9907	0.4354	0.4184	0.4212	0.7572
	2	0.7844	0.7713	0.7814	0.9929	0.4197	0.4066	0.3929	0.7366
	3	0.7846	0.7721	0.7829	0.9904	0.4036	0.3920	0.3914	0.7366
	Mean	0.7862	0.7739	0.7845	0.9913	0.4196	0.4057	0.4018	0.7435
	Std Dev	0.0029	0.0038	0.0040	0.0013	0.0158	0.0132	0.0167	0.0119
pNNCLR	1	0.8404	0.8397	0.9029	0.9919	0.4827	0.4835	0.5112	0.7971
	2	0.8412	0.8412	0.9192	0.9929	0.4735	0.4771	0.4900	0.7810
	3	0.8435	0.8443	0.9086	0.9927	0.4810	0.4807	0.4956	0.7966
	Mean	0.8417	0.8417	0.9102	0.9925	0.4791	0.4804	0.4989	0.7916
	Std Dev	0.0015	0.0023	0.0082	0.0005	0.0048	0.0031	0.0109	0.0091

Table 14
Baseline NNCLR [4] and the proposed method are compared for randomly selected five query images on a nearest neighbor based image retrieval task on the STL-10 and Tiny-imagenet datasets. The nearest neighbors are found using two different metrics corresponding to the same query image, i.e., MSE and KL divergence.

STL-10				
Query image	Mean squared error (MSE)		KL divergence	
	NNCLR [4]	Proposed method	NNCLR	Proposed method
1	0.0620	0.0460	429.0289	352.0296
2	0.0663	0.0455	518.1351	367.1804
3	0.0629	0.0385	430.6262	333.5937
4	0.0594	0.0413	469.5992	306.8343
5	0.0485	0.0374	367.4638	330.1322
Tiny-imagenet				
Query image	Mean squared error (MSE)		KL divergence	
	NNCLR	Proposed method	NNCLR	Proposed method
1	0.0954	0.0452	850.0895	287.7035
2	0.0874	0.0488	693.1914	355.6485
3	0.1003	0.0452	778.6684	281.6701
4	0.0902	0.0506	778.2897	287.0299
5	0.1232	0.0479	1034.672	309.4056

We also did some quantitative experiments on the image retrieval task as shown by Table 14. Here, we report the MSE and KL-divergence scores corresponding to randomly chosen five query images (from the test set) and their corresponding immediate nearest neighbors (from the retrieval set). The results are reported for the baseline NNCLR [4] and the proposed pNNCLR methods on STL-10 and Tiny-imagenet datasets. On STL-10 dataset, for all query images (form 1 to 5), the proposed pNNCLR method gives better MSE performance than the

baseline NNCLR [4] method by attaining a mean score of ∼ 0.02. Furthermore, the proposed pNNCLR method gives better KL-divergence score than the baseline NNCLR [4] method by a mean margin of ∼ 100. On the Tiny-imagenet dataset, a similar observation can be made, where the proposed pNNCLR method attains better MSE performance than the baseline NNCLR [4] method by an average margin of ∼ 0.05, and a better KL-divergence performance by a mean margin of ∼ 500. These results show that the proposed pNNCLR method significantly outperformed the baseline pNNCLR [4] method on the image retrieval tasks.

4.8. Future directions

In our experiments, we noticed that the size of the support set affects the performance of the SSL method. This has also been observed by other researchers. Conversely, it entails that we need to explore the role of nearest neighbors or pseudo nearest neighbors in other types of SSL methods. We hypothesize that the performance of other SSL methods may benefit from increased diversification of the semantic information. We can also move in the direction of improving the quality of nearest neighbors or the support set as it directly affects the learning in the SSL model. We can also experiment with how effective the features learned by different SSL methods are on the medical image segmentation task. We can also explore the effect of model agnostic variance regularization functions in NNCLR or pNNCLR [44].

5. Conclusion

In this paper, we studied the problem of contrastive self-supervised learning. We covered the development of the field from non-contrastive methods to contrastive methods. It was found that the nearest neighbor sampling based methods were good at increasing semantic variations during unsupervised SSL learning. However, the shortcomings of these methods were also discussed. Further, we proposed pNNCLR, a pseudo nearest neighbor based contrastive learning method, to overcome the weakness of the widely used NNCLR method. We showed how the choice of nearest neighbors in the support set can affect the quality of the learned representations. To avoid this, pNNCLR introduced the use of pseudo nearest neighbors (pNN) with stochastic sampling. Further, a smooth weight updation strategy was also used to stabilize the uncertainty in the learning process. The proposed modifications and multiple recent SSL methods were evaluated on different medical and non-medical standard datasets. Various ablations were performed to fine-tune the hyperparameters. The experiments show that the proposed sampling strategy performs significantly better than the baseline NNCLR approach while competing favorably against the other recent SSL methods.

CRediT authorship contribution statement

Momojit Biswas: Formal analysis, Software, Validation, Visualization, Writing – review & editing. **Himanshu Buckchash:** Conceptualization, Methodology, Writing – original draft, Writing – review & editing. **Dilip K. Prasad:** Conceptualization, Funding acquisition, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data and code used during this study is publicly available.

Acknowledgment

Funding: This work was supported by the Research Council of Norway Project (nanoAI, Project ID: 325741), H2020 Project (OrganVision, Project ID: 964800), HORIZON-ERC-POC Project (Spermotile, Project ID: 101123485), and VirtualStain (UiT, Cristin Project ID: 2061348).

Appendix A. Same class nearest neighbor probability calculation

Suppose, the dataset $\mathcal{D}$ on which we are training our SSL network contains classes $c_1, c_2, \ldots, c_{N_c}$, where N_c is the number of classes. Each class c_i has N_e number of items, i.e., we assume that we have a balanced dataset. When choosing a nearest neighbor $\mathrm{NN}(x_i)$ or $\mathrm{NN}(z_i)$ for a view of the input x_i, from randomly formed support set Q with cardinality N_q, the approximate maximal probability, $P[\psi]$ or $P[\mathrm{class}(\mathrm{NN}(z_i)) = \mathrm{class}(z_i)]$, can be calculated as follows. Here, z_i is the corresponding embedding of one of the views of x_i. We assume that every individual item $x \in \mathcal{D}$ has an equal probability of being randomly selected for forming the support set Q. Then, $P[\psi]$ can be defined as:

$$\begin{aligned} &P[\psi] = P[\mathrm{A} \cap \mathrm{B}] = P[\mathrm{A}|\mathrm{B}] \cdot P[\mathrm{B}] \\ \text{where, } &\mathrm{A} : \mathrm{class}(\mathrm{NN}(z_i)) = \mathrm{class}(z_i), \text{ if } P[\mathrm{B}] = 1 \\ &\mathrm{B} : \left|\{q_i | q_i \in Q \text{ and } \mathrm{class}(q_i) = \mathrm{class}(z_i)\}\right| \geq 1, \end{aligned} \tag{A.1}$$

where $|\cdot|$ denotes the set cardinality. In other words, $P[\psi]$ is the probability of the nearest neighbor function $\mathrm{NN}(\cdot)$ choosing the correct class. If $\binom{\cdot}{\cdot}$ denotes the binomial coefficient, the probability of the support set Q getting at least one item from the correct class, $P[\mathrm{B}]$, when items in Q are randomly picked from $\mathcal{D}$ with an equal probability, becomes:

$$P[\mathrm{B}] = 1 - P[\overline{\mathrm{B}}]. \tag{A.2}$$

$$P[\overline{\mathrm{B}}] = \frac{\binom{(N_c-1)N_e}{N_q}}{\binom{N_c N_e}{N_q}}. \tag{A.3}$$

using (A.3) in (A.2), we get

$$\begin{aligned} P[\mathrm{B}] &= 1 - \frac{\binom{(N_c-1)N_e}{N_q}}{\binom{N_c N_e}{N_q}} \\ &= 1 - \frac{(N_c N_e - N_e)!(N_c N_e - N_q)!}{(N_c N_e - N_e - N_q)!(N_c N_e)!} \\ &= 1 - \frac{N_c N_e - N_q}{N_c N_e} \cdot \frac{N_c N_e - 1 - N_q}{N_c N_e - 1} \cdots \frac{N_c N_e - N_e + 1 - N_q}{N_c N_e - N_e + 1}. \end{aligned} \tag{A.4}$$

$$\frac{N_c N_e - N_q}{N_c N_e} \cdot \frac{N_c N_e - 1 - N_q}{N_c N_e - 1} \cdots \frac{N_c N_e - N_e + 1 - N_q}{N_c N_e - N_e + 1} < \left(\frac{N_c N_e - N_q}{N_c N_e}\right)^{N_e}. \tag{A.5}$$

$$\frac{N_c N_e - N_q}{N_c N_e} \cdot \frac{N_c N_e - 1 - N_q}{N_c N_e - 1} \cdots \frac{N_c N_e - N_e + 1 - N_q}{N_c N_e - N_e + 1} > \left(\frac{N_c N_e - N_e + 1 - N_q}{N_c N_e - N_e + 1}\right)^{N_e}. \tag{A.6}$$

Using Eq. (A.4), Eq. (A.5) and (A.6), we get a lower and upper bound on $P[\mathrm{B}]$ as:

$$1 - \left(\frac{N_c N_e - N_q}{N_c N_e}\right)^{N_e} < P[\mathrm{B}] < 1 - \left(\frac{N_c N_e - N_e + 1 - N_q}{N_c N_e - N_e + 1}\right)^{N_e}. \tag{A.7}$$

Now, let us consider two scenarios, both with a fixed value of $N_q = 10000$. First, when we have a very large dataset where $N_c = 1000$ and $N_e = 1000$, we get $P[\mathrm{B}] \simeq 0.9999$, using Eq. (A.7). Second, when

we have a relatively smaller dataset where $N_c = 100$ and $N_e = 100$, or $N_c = 10$ and $N_e = 1000$, we get $P[\mathrm{B}] \simeq 1$, using Eq. (A.7). Hence, the probability $P[\mathrm{B}]$ stays $\simeq 1$ for both smaller as well as larger size datasets. This changes Eq. (A.1) as:

$$P[\psi] = P[\mathrm{class}(\mathrm{NN}(z_i)) = \mathrm{class}(z_i)] = P[\mathrm{A}], \tag{A.8}$$

which implies that the maximal probability, $P[\psi]$, is proportionate to the probability of choosing the correct class, which in turn depends on the quality of the representations learned by the model $\mathcal{M}_\theta$. As we know that $\mathcal{M}_\theta$ is randomly initialized and stays quite random for the starting epochs, the probability of choosing the correct class becomes $P[\psi] \simeq 0.5$, which is as good as a random selection.

References

[1] N. Adaloglou, F. Michels, H. Kalisch, M. Kollmann, Exploring the limits of deep image clustering using pretrained models, 2023, arXiv preprint arXiv:2303.17896.

[2] C.-H. Yeh, C.-Y. Hong, Y.-C. Hsu, T.-L. Liu, Y. Chen, Y. LeCun, Decoupled contrastive learning, in: European Conference on Computer Vision, Springer, 2022, pp. 668–684.

[3] Y. Zheng, C. Li, X. Zhou, H. Chen, H. Xu, Y. Li, H. Zhang, X. Li, H. Sun, X. Huang, et al., Application of transfer learning and ensemble learning in image-level classification for breast histopathology, Intell. Med. 3 (02) (2023) 115–128.

[4] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, A. Zisserman, With a little help from my friends: Nearest-neighbor contrastive learning of visual representations, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 9588–9597.

[5] X. Chen, K. He, Exploring simple siamese representation learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 15750–15758.

[6] A. Radford, J.W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al., Learning transferable visual models from natural language supervision, in: International Conference on Machine Learning, PMLR, 2021, pp. 8748–8763.

[7] X. Chen, M. Ding, X. Wang, Y. Xin, S. Mo, Y. Wang, S. Han, P. Luo, G. Zeng, J. Wang, Context autoencoder for self-supervised representation learning, 2022, arXiv preprint arXiv:2202.03026.

[8] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[9] B. Koonce, B. Koonce, EfficientNet, in: Convolutional Neural Networks with Swift for Tensorflow: Image Recognition and Dataset Categorization, Springer, 2021, pp. 109–123.

[10] T. Chen, S. Kornblith, M. Norouzi, G. Hinton, A simple framework for contrastive learning of visual representations, in: International Conference on Machine Learning, PMLR, 2020, pp. 1597–1607.

[11] S. Gidaris, P. Singh, N. Komodakis, Unsupervised representation learning by predicting image rotations, 2018, arXiv preprint arXiv:1803.07728.

[12] M. Noroozi, P. Favaro, Unsupervised learning of visual representations by solving jigsaw puzzles, in: European Conference on Computer Vision, Springer, 2016, pp. 69–84.

[13] I. Misra, C.L. Zitnick, M. Hebert, Shuffle and learn: unsupervised learning using temporal order verification, in: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, the Netherlands, October 11–14, 2016, Proceedings, Part I. Vol. 14, Springer, 2016, pp. 527–544.

[14] K. He, H. Fan, Y. Wu, S. Xie, R. Girshick, Momentum contrast for unsupervised visual representation learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 9729–9738.

[15] T. Brown, B. Mann, N. Ryder, M. Subbiah, J.D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, in: Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 1877–1901.

[16] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A.C. Berg, W.-Y. Lo, et al., Segment anything, 2023, arXiv preprint arXiv:2304.02643.

[17] Y. Liu, T. Han, S. Ma, J. Zhang, Y. Yang, J. Tian, H. He, A. Li, M. He, Z. Liu, et al., Summary of chatgpt/gpt-4 research and perspective towards the future of large language models, 2023, arXiv preprint arXiv:2304.01852.

[18] P. Goyal, M. Caron, B. Lefaudeux, M. Xu, P. Wang, V. Pai, M. Singh, V. Liptchinsky, I. Misra, A. Joulin, et al., Self-supervised pretraining of visual features in the wild, 2021, arXiv preprint arXiv:2103.01988.

[19] C. Doersch, A. Gupta, A.A. Efros, Unsupervised visual representation learning by context prediction, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1422–1430.

[20] C. Tang, X. Liu, X. Zhu, J. Xiong, M. Li, J. Xia, X. Wang, L. Wang, Feature selective projection with low-rank embedding and dual Laplacian regularization, IEEE Trans. Knowl. Data Eng. 32 (9) (2019) 1747–1760.

[21] C. Tang, X. Zheng, W. Zhang, X. Liu, X. Zhu, E. Zhu, Unsupervised feature selection via multiple graph fusion and feature weight learning, Sci. China Inf. Sci. 66 (5) (2023) 1–17.

[22] C. Hou, J. Zhang, H. Wang, T. Zhou, Subclass-balancing contrastive learning for long-tailed recognition, 2023, arXiv preprint arXiv:2306.15925.

[23] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, A. Joulin, Emerging properties in self-supervised vision transformers, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 9650–9660.

[24] R. Zhang, P. Isola, A.A. Efros, Colorful image colorization, in: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, the Netherlands, October 11-14, 2016, Proceedings, Part III. Vol. 14, Springer, 2016, pp. 649–666.

[25] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, A.A. Efros, Context encoders: Feature learning by inpainting, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2536–2544.

[26] R. Zhang, P. Isola, A.A. Efros, Split-brain autoencoders: Unsupervised learning by cross-channel prediction, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1058–1067.

[27] A.v.d. Oord, Y. Li, O. Vinyals, Representation learning with contrastive predictive coding, 2018, arXiv preprint arXiv:1807.03748.

[28] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, A. Joulin, Unsupervised learning of visual features by contrasting cluster assignments, in: Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 9912–9924.

[29] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, I. Sutskever, Generative pretraining from pixels, in: International Conference on Machine Learning, PMLR, 2020, pp. 1691–1703.

[30] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, et al., Bootstrap your own latent-a new approach to self-supervised learning, in: Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 21271–21284.

[31] Z. Xie, Z. Zhang, Y. Cao, Y. Lin, J. Bao, Z. Yao, Q. Dai, H. Hu, Simmim: A simple framework for masked image modeling, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 9653–9663.

[32] S. Zhang, F. Zhu, R. Zhao, J. Yan, Patch-level contrasting without patch correspondence for accurate and dense contrastive representation learning, 2023, arXiv preprint arXiv:2306.13337.

[33] H.-Y. Lee, J.-B. Huang, M. Singh, M.-H. Yang, Unsupervised representation learning by sorting sequences, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 667–676.

[34] H. Buckchash, B. Raman, Sustained self-supervised pretraining for temporal order verification, in: Pattern Recognition and Machine Intelligence: 8th International Conference, PReMI 2019, Tezpur, India, December 17-20, 2019, Proceedings, Part I, Springer, 2019, pp. 140–149.

[35] B. Fernando, H. Bilen, E. Gavves, S. Gould, Self-supervised video representation learning with odd-one-out networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 3636–3645.

[36] J. Zhou, C. Wei, H. Wang, W. Shen, C. Xie, A. Yuille, T. Kong, ibot: Image bert pre-training with online tokenizer, 2021, arXiv preprint arXiv:2111.07832.

[37] I. Misra, L.v.d. Maaten, Self-supervised learning of pretext-invariant representations, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 6707–6717.

[38] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2009, pp. 248–255.

[39] A. Coates, A. Ng, H. Lee, An analysis of single-layer networks in unsupervised feature learning, in: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, 2011, pp. 215–223.

[40] A. Krizhevsky, G. Hinton, et al., Learning Multiple Layers of Features from Tiny Images, Toronto, ON, Canada, 2009.

[41] M.A. Mnmoustafa, Tiny ImageNet, Kaggle, 2017, URL https://kaggle.com/competitions/tiny-imagenet.

[42] J. Yang, R. Shi, D. Wei, Z. Liu, L. Zhao, B. Ke, H. Pfister, B. Ni, MedMNIST v2-A large-scale lightweight benchmark for 2D and 3D biomedical image classification, Sci. Data 10 (1) (2023) 41.

[43] B.S. Veeling, J. Linmans, J. Winkens, T. Cohen, M. Welling, Rotation equivariant CNNs for digital pathology, in: Medical Image Computing and Computer Assisted Intervention–MICCAI 2018: 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part II. Vol. 11, Springer, 2018, pp. 210–218.

[44] A. Bardes, J. Ponce, Y. Lecun, VICReg: Variance-invariance-covariance regularization for self-supervised learning, in: ICLR 2022-International Conference on Learning Representations, 2022.

Momojit Biswas is a researcher at TCS-Research & Innovation Labs, India. He received the Bachelor of Technology degree from Jadavpur University, Kolkata, India. His research interests include the areas of machine learning, computer vision, and deep learning, with a specific focus on exploring the vast potential of Natural Language Processing (NLP).

Himanshu Buckchash is a postdoc at the Department of Computer Science, UiT The Arctic University of Norway. He received the Ph.D. degree in Computer Science and Engineering from Indian Institute of Technology Roorkee. His research interests include self supervised learning, deep learning and computer vision.

Dilip K. Prasad is an associate professor at the Department of Computer Science, UiT The Arctic University of Norway. He received the Ph.D. degree in Computer Science and Engineering from Nanyang Technological University, Singapore. His current research interests include image processing, machine learning and artificial intelligence. He has published 100+ internationally peer-reviewed research articles. He has been a reviewer for more than 60 journals and 30+ conferences including CVPR, ICCV, ECCV, NIPS, AAAI, BMVC, WACV, ACCV, etc.