Deep kNN for Medical Image Classification

Jiaxin Zhuang[1], Jiabin Cai[1], Ruixuan Wang[1*], Jianguo Zhang[2*], Wei-Shi Zheng[1,3,4]

[1] School of Data and Computer Science, Sun Yat-sen University, China
[2] Department of Computer Science and Engineering, Southern Unive rsity of Science and Technology, China
[3] Key Laboratory of Machine Intelligence and Advanced Computing, MOE, Guangzhou, China
[4] Pazhou Lab, Guangzhou, China

Abstract. Human-level diagnostic performance from intelligent systems often depends on large set of training data. However, the amount of available data for model training may be limited for part of diseases, which would cause the widely adopted deep learning models not generalizing well. One alternative simple approach to small class prediction is the traditional k-nearest neighbor (kNN). However, due to the non-parametric characteristics of kNN, it is difficult to combine the kNN classification into the learning of feature extractor. This paper proposes an end-to-end learning strategy to unify the kNN classification and the feature extraction procedure. The basic idea is to enforce that each training sample and its K nearest neighbors belong to the same class during learning the feature extractor. Experiments on multiple small-class and class-imbalanced medical image datasets showed that the proposed deep kNN outperforms both kNN and other strong classifiers.

Keywords: Small class · Deep kNN · Intelligent diagnosis

1 Introduction

With recent advance particularly in deep learning, intelligent diagnosis has shown human-level performance for various diseases [1, 2]. However, in many intelligent diagnosis tasks, the amount of available data for model training is often limited for some or all diseases. Small training data often leads to the over-fitting issue, i.e., the trained model does not generalize well to new data. While the issue can be often alleviated by fine-tuning a model which was originally trained in another task [3], such transfer learning may not work well if the image domain in the current task is far from that of the original task. Recently developed meta-learning techniques for few-shot learning problems seem to provide a plausible solution to the small-class classification tasks [4]. However, these techniques often presume the access to a large number of additional small classes for model training, which is impractical in the tasks of intelligent diagnosis.

*corresponding authors.

Another simple but often effective approach is the k-nearest neighbor (kNN) classification [5, 11], where feature extraction and/or dimensionality reduction are often performed to obtain a concise feature vector to represent the original image [8, 12, 13]. However, because the feature extraction procedure is often predetermined and independent of the classification task of interest, the extracted features may not be discriminative enough for the classification task. To alleviate this issue, metric learning can be applied to transform the extracted features into a new feature space where the transformed features become more discriminative, such as neighborhood component analysis (NCA) [14] and large margin nearest neighbor methods [15]. The metric learning methods depend on the originally extracted features which may already omit certain features helpful for the classification task. To learn to extract discriminative features directly from original image data, recently a triplet loss function was proposed to train a convolutional neural network (CNN) through which an image will be transformed to a low-dimensional but discriminative feature vector [16, 17]. However, the process of CNN training is independent of the latter kNN classification step. It would be desirable to unify feature extraction and kNN classification into a single step as done in current CNN classifiers. However, searching for K nearest neighbours for each training data is a non-differentiable operation, and therefore it is not easy to combine the neighbor search into the training of a feature extractor.

This paper proposes an end-to-end learning strategy to unify the kNN classification and the feature extraction process, particularly for classification of small classes. The basic idea is to enforce that each training image and its K nearest neighbors belong to the same class during learning feature extractor. By unifying the feature extraction and the kNN classification procedure, a better feature extractor can be learned specifically for the kNN classifier and the task of interest. Comprehensive evaluations on multiple medical image datasets showed that the proposed approach, called deep kNN, outperforms various kNNs and even CNN classifiers particularly for small class prediction. Compared to the traditional triplet loss, the proposed novel loss function can help train the feature extractor much faster, as confirmed in experiments. What's more, the proposed deep kNN is independent of network architectures, and therefore can be directly combined with any existing convolutional or fully connected neural network architectures.

2 Deep kNN

A traditional kNN classifier does not include feature extraction, i.e., the feature extraction is done separately from the k-nearest neighbor search. Since the feature extraction does not consider any task-specific information, extracted features could be not discriminative enough for the kNN classification. It would be ideal to learn to extract features specific to the classification task of interest. However, due to the non-parametric characteristics of the nearest neighbor search process, so far it is not clear how to combine feature learning and the k-nearest neighbor search into a unified process for the kNN classifier.

2.1 Problem formulation

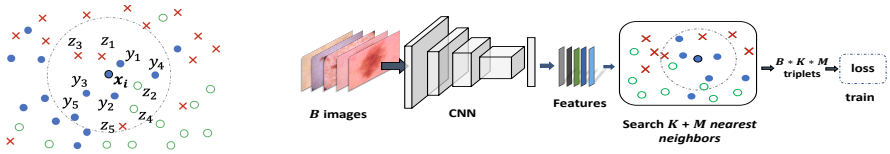

Fig. 1. Demonstration of neighbor search (left) and deep kNN training (right). Left: for each image $\mathbf{x}_i$, K nearest neighbors $\{\mathbf{y}_k^{t,i}\}$ of the same class and M nearest neighbors $\{\mathbf{z}_m^{t,i}\}$ from all other classes are searched to generate $K \cdot M$ triplets. Here K=5, M=5, $\mathbf{y}_k^{t,i}$ and $\mathbf{z}_m^{t,i}$ are simplified to $\mathbf{y}_k$ and $\mathbf{z}_m$. Right: during training, K within-class and M cross-class nearest neighbors are searched within the mini-batch for each image.

We propose a deep learning strategy to naturally unify the two steps of kNN classifier, where the feature extractor is represented by a convolutional or fully connected neural network whose output of the last layer is a feature vector representing the input image. The intuitive idea behind the strategy is to train a feature extractor based on which any training image and its k-nearest neighbors are forced to belong to the same class. In this way, the k-nearest neighbor search procedure is naturally incorporated into the process of training the feature extractor. If such a feature extractor can be trained, we would expect any test image and its K nearest neighbors in the training set is probably from the same class. The challenge is how to formulate this idea for feature extractor training.

We propose a novel triplet loss to solve this challenge. At the beginning stage of training a feature extractor, since the parameters of the feature extractor are initially either randomly set or from a pre-trained model based on a public dataset (e.g., ImageNet), it is not surprising that the distributions of different classes of images would be interleaved in the feature space. That means, among the several nearest neighbors of any specific training image, one or more neighbors may not be from the same class of the training image. To make any image and its nearest neighbors belong to the same class, the feature extractor needs to be updated such that the distance between the image and any of its nearest neighbors is closer than the distance between the image and any image of other classes in the feature space. Formally, denote the feature extractor after the $(t-1)^{th}$ training iteration by $\mathbf{f}(\cdot;\boldsymbol{\theta}_{t-1})$, where $\boldsymbol{\theta}$ represents the parameters of the feature extractor to be learned. Also denote the i-th training image by $\mathbf{x}_i$, its K nearest neighbors of the same class after the $(t-1)^{th}$ training iteration by $\{\mathbf{y}_k^{t,i}, k = 1, \ldots, K\}$, and its M nearest neighbors from all the other classes after the $(t-1)^{th}$ training iteration by $\{\mathbf{z}_m^{t,i}, m = 1, \ldots, M\}$ (Figure 1, Left). Then, after the t^{th} training iteration, for each training image $\mathbf{x}_i$, we expect its K nearest neighbors of the same class based on the updated feature extractor $\mathbf{f}(\cdot;\boldsymbol{\theta}_t)$ are closer to the training image $\mathbf{x}_i$ compared to the distance between $\mathbf{x}_i$

and any image of the other classes (particularly the M nearest neighbors from the other classes), i.e., the following inequality is expected to be satisfied,

$$\|\mathbf{f}(\mathbf{x}_i;\boldsymbol{\theta}_t) - \mathbf{f}(\mathbf{y}_k^{t,i};\boldsymbol{\theta}_t)\| + \alpha < \|\mathbf{f}(\mathbf{x}_i;\boldsymbol{\theta}_t) - \mathbf{f}(\mathbf{z}_m^{t,i};\boldsymbol{\theta}_t)\|, \ \forall\, k, m \tag{1}$$

where $\|\cdot\|$ represents the L_p norm ($p = 2$ in experiments), and α is a positive constant further enforcing the inequality constraint. Based on this inequality constraint, the loss function for the feature extractor can be defined as

$$l(\mathbf{x}_i, \mathbf{y}_k^{t,i}, \mathbf{z}_m^{t,i}; \boldsymbol{\theta}_t) = [\|\mathbf{f}(\mathbf{x}_i;\boldsymbol{\theta}_t) - \mathbf{f}(\mathbf{y}_k^{t,i};\boldsymbol{\theta}_t)\| + \alpha - \|\mathbf{f}(\mathbf{x}_i;\boldsymbol{\theta}_t) - \mathbf{f}(\mathbf{z}_m^{t,i};\boldsymbol{\theta}_t)\|]_+ \tag{2}$$

where $[d]_+ = \max(0, d)$, such that the loss is 0 when the inequality constraint is satisfied, and becomes larger when the constraint is further from being satisfied. Considering all N images in the training set, the loss function becomes

$$L(\boldsymbol{\theta}_t) = \frac{1}{NKM} \sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{m=1}^{M} l(\mathbf{x}_i, \mathbf{y}_k^{t,i}, \mathbf{z}_m^{t,i}; \boldsymbol{\theta}_t)\,. \tag{3}$$

2.2 Neighbor search during training

Since feature extractor is updated over training iterations, part of (or the whole) K nearest neighbors for one specific training image based on the feature extractor at previous iteration could become no longer the K nearest neighbors at current iteration. Therefore, after updating the feature extractor by minimizing $L(\boldsymbol{\theta}_t)$ at the t^{th} iteration, for each training image, its K nearest neighbors of the same class and M nearest neighbors from the other classes will be searched (based on Euclidean distance here) and updated again before updating the feature extractor in next iteration. In this way, the k-nearest neighbor classification performance on the training dataset will be naturally evaluated and gradually improved during feature extractor training. Therefore, the proposed training strategy unifies the feature extractor and the kNN classification, thus called *deep kNN*. Note during the training process, the non-parametric k-nearest neighbor search plays the role of providing training data at each iteration, and the search process is not involved in the derivative of extractor parameters. Therefore, the difficulty in differentiating the non-parametric process is naturally circumvented.

In analogy to the stochastic gradient descent (SGD) widely adopted for training deep learning models, a similar SGD method can be used here to update the feature extractor. In this case, the feature extractor will be updated once per mini-batch of training images. However, this would cause the update (search) of K nearest neighbors for each image in the next mini-batch set much more computationally expensive, because all training images need to be fed into the feature extractor after each mini-batch training to find the K nearest neighbors for each image in the new mini-batch set. To alleviate this issue, similar to the sampling strategy for minimizing the triplet loss in related work [18, 16], the $(K + M)$ nearest neighbors for each image may be searched just within the mini-batch (Figure 1, Right). To guarantee that enough number of within-

and cross-class nearest neighbors exist in each mini-batch, stratified sampling was adopted for mini-batch generation (see Experimental setup). Experiments showed such within-batch nearest neighbor search did not downgrade deep kNN performance compared to searching for nearest neighbors from the whole dataset.

2.3 Comparison with traditional triplet loss

While the proposed triplet loss has a similar form compared to the traditional triplet loss [16, 17], there exists a few significant differences between them. Traditionally, triplets are formed often by the *furthest* within-class pairs and the nearest cross-class pairs, while the proposed triplet is formed by the *nearest* within-class pairs and nearest cross-class pairs. The objective of traditional triplet loss is to enforce all pair-wise distances within a class are smaller than the distances between any data of the class and all data of other classes. In comparison, the proposed method only requires any data and its K nearest neighbors of the same class are closer than the distance between the data and other classes of data. That means, traditional triplet loss is used to train a feature extractor based on which the distribution of each class of data is clustered more compactly, while the proposed triplet loss is used to help separate distributions of different classes apart from each other such that the K nearest neighbors of any data belong to the same class as that of the data, without requiring the distribution of each class to be compactly clustered (Supplementary Figure S1(a)(b)). Such a difference also indicates that the feature extractor based on the proposed triple loss can be trained more easily (i.e., faster convergence) than that based on traditional triplet loss. Last but important, the proposed triplet loss can be used to train a deep kNN by combining the two steps of kNN classification, while the traditional triplet loss cannot. As shown below, deep kNN performs much better than traditional two-step kNN classifiers.

3 Experiment

Experimental setup: Experiments were extensively carried out on two skin image datasets, SD-198 and Skin-7, and one Pneumonia X-ray dataset *, each including small classes and/or different level of class imbalance (Table 1). For each dataset, images were resized into 256×256 and randomly cropped to 224×224 pixels for training, followed by a random horizontal flip. The similar operation was performed for testing, except that only one cropped image was generated from the center region of each test image. During training a deep kNN, the batch size B varied a bit for different datasets due to the varying scales of datasets, set to 96 on Skin-7 and pneumonia dataset, and 90 on SD-198 dataset. To guarantee K nearest neighbors of the same class for each image in a batch particularly on the SD-198 dataset, the stratified sampling was adopted, i.e., forming batches by randomly sampling 9 classes and then randomly sampling 10 images for each

*https://www.kaggle.com/c/rsna-pneumonia-detection-challenge.

Table 1. Dataset statistics. More ticks denote more imbalanced.

Dataset	#Class	#Train	#Test	ImageSize	SmallestClass	LargestClass	Imbalance
Pneumonia	3	21345	5339	1024×1024	6012	11821	$\surd$
SD-198 [19]	198	5,206	1377	[260, 4752]	9	60	$\surd\surd$
Skin-7 [20]	7	8005	2005	600×450	115	6705	$\surd\surd\surd$

of the 9 class. During training, the SGD optimizer was used, with the initial learning rate 0.01. Learning rate decayed by 0.5 for every 20 epochs on Skin-7 dataset and pneumonia dataset, and every 30 epochs on SD-198 dataset.

During testing, all test and training images were fed into the trained feature extractor only once to get the corresponding feature vectors. For each test image, its k-nearest neighbors within the training dataset were found in the feature space. The class label of the test image was then determined by the majority class in those neighbors. For simplicity, K denotes the number of nearest neighbors used in training while K_p denotes the number of nearest neighbors used in testing. Overall accuracy (Acc), mean class recall over all classes (MCR) and the recall on the smallest class (RS) were used for measurement.

Effectiveness of deep kNN: To demonstrate effectiveness of the proposed deep kNN, our method is compared to several baseline methods. Two baseline kNN classifiers, *kNN (VGG19)* and *kNN (ResNet)*, were respectively built on the feature extractor part of *VGG19* [21] and *ResNet50* [22] pretrained on ImageNet. The other baseline kNN *Triplet (ResNet50)* was built on the feature extractor trained with the traditional triplet loss [17], using the pretrained ResNet50 on ImageNet as the backbone. The same ResNet50 backbone was trained with the proposed deep kNN learning strategy. Note that similar amount of effort was put into tuning each baseline method. Table 2 showed that the proposed deep kNN (rows 4, 8) outperforms all the three baseline kNN classifiers (rows 1-3, 5-7). It is reasonable that both the baseline *kNN (VGG19)* and *kNN (ResNet)* were outperformed by the *Triplet (ResNet)* and the proposed deep kNN, because the latter were trained using the specific data of the task of interest. However, it is surprising that the deep kNN also outperforms the *Triplet (ResNet)*, especially considering that the distribution of the *training* data based on the traditional triplet loss are more compactly clustered than that based on the proposed triplet loss (Supplementary Figure S1(a)(b)). Detailed inspection shows that the distribution of the test dataset based on the deep kNN is more compactly clustered than that based on the traditional triplet loss (Supplementary Figure S1(c)(d)), suggesting the feature extractor trained by the proposed triplet loss is more generalizable to unknown data than that by the traditional triplet loss.

We also compared deep kNN with CNN classifier of the same ResNet50 backbone. Standard deviations over multiple runs are within the range [0.5%, 1.2%] for accuracy, mean class recall (MCR), and the recall on the smallest class in all experiments. The overall performance (Acc, MCR) of deep kNN (Table 2, rows 4, 8) is close to that of CNN classifier (Table 2, 2nd last row, using class-weighted

Table 2. Comparison between deep kNN and various baselines on multiple datasets. K=5 and M=5. Similar findings were obtained with other values for K and M. For each model, multiple runs were performed with similar performance observed.

Datasets	Skin-7			Pneumonia			SD-198		
Methods	Acc	MCR	RS	Acc	MCR	RS	Acc	MCR	RS
kNN (VGG19, $K_p = 1$) [21]	71.4	42.0	37.9	52.7	52.4	43.1	24.5	24.4	20.8
kNN (ResNet50, $K_p = 1$) [22]	75.6	49.9	55.2	52.4	51.2	38.9	27.6	28.4	29.2
Triplet(ResNet50,$K_p = 1$) [16]	82.9	64.2	63.2	67.8	66.3	65.2	60.0	60.1	47.3
deep kNN (ours, $K_p = 1$)	**88.2**	**77.4**	**77.0**	**71.0**	**69.3**	**67.3**	**64.5**	**64.1**	**48.3**
kNN (VGG19, $K_p = 5$) [21]	70.7	30.8	10.3	55.4	55.7	45.0	15.1	13.3	4.2
kNN (ResNet50, $K_p = 5$) [22]	74.4	37.5	34.5	55.5	54.7	39.7	19.1	17.5	12.5
Triplet(ResNet50,$K_p = 5$) [16]	84.3	68.3	67.3	70.0	68.7	66.3	60.2	60.0	47.2
deep kNN (ours, $K_p = 5$)	**89.1**	**78.9**	**77.3**	**71.1**	**69.4**	**69.0**	**65.1**	**64.3**	**48.3**
Weighted-CE(ResNet50) [23]	88.0	80.2	76.6	71.1	69.1	70.1	61.9	62.4	47.7
deep kNN* (ours, $K_p = 5$)	**90.3**	**81.0**	**80.4**	**71.6**	**71.1**	**70.9**	**66.4**	**66.4**	**51.5**

cross entropy to handle data imbalance) on Skin-7 and Pneumonia. Importantly, on the small-class dataset SD-198 and the small class (column RS) of Skin-7 (no small class on Pneumonia), deep kNN clearly outperforms the CNN classifier, confirming that deep kNN works better particularly for small class prediction. CNN classifier could become overfitting on small classes, while deep kNN could largely alleviate this issue by increasing the number of training data (triplets) on small classes. Also interestingly, after fine-tuning the trained CNN classifiers (with the last fully connected layer removed) with the proposed triplet loss, the resulting new deep kNN (Table 2, last row) further improved the performance.

Deep kNN with MLP: In some scenarios, only feature vectors were originally available in dataset. In this case, the deep kNN learning strategy still works, not based on a CNN structure but on a multilayer perceptron (MLP) structure. To demonstrate effectiveness of deep kNN under this condition, features vector of each image in Skin-7 was extracted from output of the last convolutional layer of a pre-trained ResNet50, and then with the vectors as original data, three-layer MLPs (with batch normalization and ReLU activation) were trained respectively based the proposed triplet loss (*MLP+deep-kNN* in Table 3), the traditional triplet loss (*MLP+triplets*), and the cross-entropy loss (*MLP+CE*). The neighborhood component analysis (NCA) and large margin nearest neighbor (LMNN) methods were also evaluated with the feature vectors as input (note NCA and LMNN were not used as baselines in Table 2 because two-dimensional image data cannot be used as input to the two methods). Table 3 shows that all the linear (*NCA*, *LMNN*) and non-linear (*MLP+triplets*, *MLP+CE*, *MLP+deep-kNN*) transformations of the initial feature vectors would help improve the performance of kNN classification compared to the basic kNN (*kNN-basic*). Among them, deep kNN outperforms all others, again confirming the effectiveness of the proposed learning strategy for kNN classification.

Table 3. Performance comparison when MLP was used as backbone.

Methods	kNN-basic	NCA	LMNN	MLP+triplets	MLP+CE	MLP+deep-kNN
Acc	64.53	75.06	81.85	77.13	78.85	**85.09**
MCR	34.49	36.39	61.78	62.22	63.11	**66.08**
RS	10.34	27.59	62.07	63.43	64.21	**67.30**

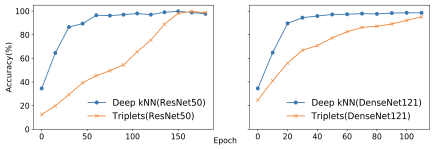

Fig. 2. Training process of feature extractor with the proposed (blue) and traditional triplet loss (orange). Left: ResNet50 on SD-198; Right: DenseNet121 on Skin-7.

Speed of convergence: We compared the effects of the proposed triplet loss with the traditional triplet loss on the convergence of the optimization using different backbone structures and datasets. It was observed that the proposed triplet loss takes much fewer epochs than the traditional traditional loss to reach the same level of training accuracy (Figure 2), faster in convergence regardless of model structures and datasets. Note that the deep kNN learning strategy only requires that each sample and its K nearest neighbours of the same class are close enough, while the traditional triplet loss requires all samples of the same class should be closer than the cross-class samples. The stronger constraints in the traditional triple loss is probably the key cause to the slower convergence.
Flexibility with Architecture. Deep kNN is independent of choice of model architectures. To show this, we test deep kNN with four widely used CNNs. For each backbone, the network was trained with cross-entropy loss on the dataset of task of interest, and the output layer was then removed to get the task-specific feature extractor. Such feature extractor was fixed and then a traditional kNN was used to predict each test image, resulting in a strong baseline kNN (because such baseline kNN is based on a task-specific feature extractor). With each backbone network, the strong baseline kNN was compared to the corresponding deep kNN model. Table 4 shows deep kNNs with different backbones perform slightly differently, but all outperforming the corresponding strong baseline kNNs, demonstrating the deep kNN learning strategy is robust to structures of feature extractors. Additional evaluations shows the deep kNN is also robust to the hyper parameters K, M, and K_p (Supplementary Tables S3 and S4).

4 Conclusion

In this paper, we introduced a novel deep learning approach, called deep kNN, which for the first time unifies the feature extraction and the kNN classification

Table 4. Performance of the deep kNN with different CNN backbones on the SD-198 dataset. Similar findings were obtained on other datasets. K=5, M=5 and K_p=5.

Models	VGG19		ResNet50		Dense121		SE-ResNet50	
	kNN	deep kNN	kNN	deep kNN	kNN	deep kNN	kNN	deep kNN
Acc	59.16	**61.51**	60.17	**65.12**	60.54	**64.02**	61.85	**62.79**
MCR	56.33	**61.93**	57.35	**64.34**	60.12	**65.11**	59.32	**62.27**
RS	43.12	**45.43**	46.01	**48.31**	50.42	**52.21**	50.01	**53.12**

procedure. Experiments showed that keep kNN not only performs better than traditional two-step kNN classifiers, but also better than CNN classifiers particularly on small class prediction. Therefore, deep kNN provides a new approach to intelligent diagnosis of diseases with limited training data.

Acknowledgement. Acknowledgement: This work is supported in part by the National Key Research and Development Program (grant No. 2018YFC1315402), the Guangdong Key Research and Development Program (grant No. 2019B020228001), the National Natural Science Foundation of China (grant No. U1811461), and the Guangzhou Science and Technology Program (grant No. 201904010260).

References

1. Esteva, Andre, et al.: Dermatologist-level classification of skin cancer with deep neural networks. Nature **542**(7639), 115–118 (2017)
2. Litjens, Geert, et al.: A survey on deep learning in medical image analysis. Med Image Anal **42**, 60–88 (2017)
3. He, Kaiming, Ross Girshick, and Piotr Dollár.: Rethinking imagenet pre-training. In: CVPR. pp. 4918–4927(2019)
4. Li, Aoxue, et al.: Large-scale few-shot learning: Knowledge transfer with class hierarchy. In: CVPR. pp. 7212–7220 (2019)
5. Bingham, Ella, and Heikki Mannila.: Random projection in dimensionality reduction: applications to image and text data. In: KDD. pp. 245–250 (2001)
6. Taigman, Yaniv, et al.: Deepface: Closing the gap to human-level performance in face verification. In: CVPR. pp. 1701–1708 (2014)
7. Bertinetto, Luca, et al. Fully-convolutional siamese networks for object tracking. In: ECCV. pp. 850–865 (2016)
8. Turk, Matthew, and Alex Pentland.: Face recognition using eigenfaces. In: CVPR. pp. 586–587 (1991)
9. Dalal, Navneet, and Bill Triggs.: Histograms of oriented gradients for human detection. In: CVPR pp. 886–893 (2005)
10. Wei, Xiu-Shen, Jianxin Wu, and Quan Cui.: Deep learning for fine-grained image analysis: A survey. arXiv:1907.03069 (2019)
11. Zhang, Hao, et al.: SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In: CVPR. pp. 2126–2136 (2006)
12. Koniusz, Piotr, et al.: Higher-order occurrence pooling for bags-of-words: Visual concept detection. IEEE Trans. Pattern Anal. Mach. Intell. **39**(2), 313–326 (2016)

13. Yosinski, Jason, et al.: How transferable are features in deep neural networks?. In: NeurIPS. pp. 3320–3328 (2014)
14. Goldberger, Jacob, et al.: Neighbourhood components analysis. In NeurIPS. pp. 513–520 (2005)
15. Weinberger, Kilian Q., and Lawrence K. Saul.: Distance metric learning for large margin nearest neighbor classification. J Mach Learn Res **10**(2) (2009)
16. Hermans, Alexander, Lucas Beyer, and Bastian Leibe.: In defense of the triplet loss for person re-identification. arXiv:1703.07737 (2017).
17. Schroff, Florian, Dmitry Kalenichenko, and James Philbin.: Facenet: A unified embedding for face recognition and clustering. In: CVPR. pp. 815–823 (2015)
18. Gordo, Albert, et al.: End-to-end learning of deep visual representations for image retrieval. IJCV **124**(2), 237–254 (2017)
19. Sun, Xiaoxiao, et al.: A benchmark for automatic visual classification of clinical skin disease images. In: ECCV. pp. 206–222 (2016)
20. Codella, Noel CF, et al.: Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (isbi), hosted by the international skin imaging collaboration (isic). In: ISBI.pp.168–172 (2018)
21. Simonyan, Karen, and Andrew Zisserman.: Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556 (2014)
22. He, Kaiming, et al. Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
23. Zhuang, Jiaxin, et al.: Care: Class attention to regions of lesion for classification on imbalanced data. In: MIDL. pp. 588–597 (2019)

Supplementary Material

Papar ID 2908

Anonymous Submission

Distributions of extracted feature vectors: While the distribution of the training dataset based on the traditional triplet loss is more compactly clustered than that based on the deep kNN (Figure S1(a)(b)), the distribution of the test dataset based on the deep kNN is more compactly clustered than that based on the traditional triplet loss (Figure S1(c)(d)), suggesting the feature extractor trained by the proposed triplet loss is more generalizable to unknown data than that by the traditional triplet loss.

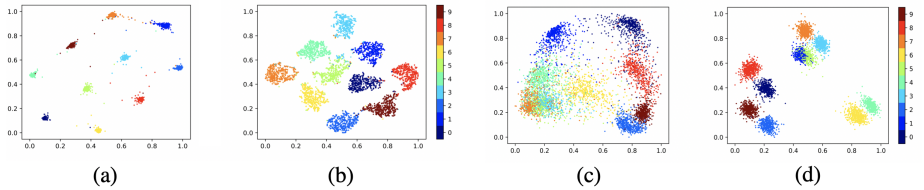

Fig. S1: Distributions of each class of data by applying t-SNE to feature extractor outputs. (a) with traditional triplet loss on MNIST dataset, (b) with deep kNN on MNIST, (c) with traditional triplet loss on STL10, (d) with deep kNN on STL10. Each dot corresponds to one image, with colors coding different classes.

Experiments on natural image datasets: In addition to the three medical datasets, the proposed method was also evaluated on three natural image dataset to demonstrate to its generalization on different image domains (Table S1). Results on natural image dataset are consistent with conclusions in medical image datasets (Table S2).

Effect of K and M. As the hyper parameters of deep kNN, K and M might affect the classification performance of the trained deep kNN. Here for simplicity, K_p was set the same as K. Table S3 shows that, by different combinations of K and M, the classification performance varied slightly, around 64% for overall accuracy (Acc). More importantly, all the deep kNNs outperform the baselines, together suggesting that the proposed deep kNN learning strategy is robust to hyper parameters K and M. In addition, a mid-level value of M (e.g., 5) seems slightly better than smaller or larger ones no matter what the K value is, and larger K (e.g., 7) would not improve classification performance, but with training time increased significantly. Therefore, K was set to 5 by default in other tests.

Effect of K_p. In essence, the number of nearest neighbours K_p used in the testing stage can be different from K used in the learning. To evaluate the effect

Table S1: Datasets for diverse natural image classification tasks, from general to fine-grained classifications.

Dataset	#Class	#Train	#Test	ImageSize	SmallestClass	LargestClass
STL10 [1]	10	5,000	8,000	96×96	500	500
Tiny-ImageNet [2]	200	100,000	10,000	64×64	500	500
CUB200-2011 [3]	200	5,994	5,794	[120, 500]	29	30

Table S2: Comparison between deep kNN and various baselines on natural datasets. K=5 and M=5. Similar findings were obtained with other values for K and M. Multiple runs were performed with similar performance observed.

Models	kNN (ResNet50, $K_p = 1$)	Triplet (ResNet50, $K_p = 1$)	deep kNN (ours, $K_p = 1$)	kNN (ResNet50, $K_p = 5$)	Triplet (ResNet50, $K_p = 5$)	deep kNN (ours, $K_p = 5$)	Weighted-CE (ResNet50)	deep kNN* (ours, $K_p = 5$)
STL10	92.39	69.31	**95.21**	93.32	70.52	**96.91**	94.06	**95.00**
CUB200	40.75	65.34	**74.95**	42.47	66.20	**75.32**	79.3	**81.10**
Tiny200	58.82	67.32	**74.32**	62.44	73.03	**75.32**	79.30	**80.52**

Table S3: Effect of K and M. Here $K_p = K$.

SD-198	K=3		K=5		K=7	
	Acc	MCA	Acc	MCA	Acc	MCA
M=1	63.7	64.5	62.4	61.6	63.4	62.6
M=5	**65.0**	**65.0**	**65.1**	**64.3**	64.2	**64.0**
M=9	64.8	63.9	63.4	62.2	**64.8**	63.6

Table S4: Effect of $K_p, M = 5$.

Dataset	SD-198		Skin-7		Pneumonia	
K_p	Acc	MCA	Acc	MCA	Acc	MCA
1	64.5	64.1	88.2	77.4	71.0	69.4
3	64.7	64.3	88.2	76.4	71.0	69.4
5	**65.1**	**64.3**	**89.1**	**78.9**	**71.1**	**69.4**
7	65.0	64.3	88.3	76.5	71.0	69.4
9	65.0	64.3	88.3	76.4	71.0	69.4

of K_p, deep kNN was trained by varying K_p, but with fixed K and M. Results in Table S4 showed that the performance of deep kNN varied little with varying K_p's on all the three datasets, all outperforming the baseline methods, suggesting that the deep kNN learning strategy is robust to the hyper parameter K_p as well.

References

1. Coates, A., Ng, A., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. In: the International Conference on Artificial Intelligence and Statistics (2011)
2. Le, Y., Yang, X.: Tiny imagenet visual recognition challenge. http://cs231n.stanford.edu (2015)
3. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: Caltech-UCSD Birds 200 (2011)