Multi-scale Deep Nearest Neighbors

Abhijeet Chauhan*
School of Computer Science
Carleton University
Ottawa, Canada
abhijeetchauhan@cmail.carleton.ca

Omid Davoudi*
School of Computer Science
Carleton University
Ottawa, Canada
omiddavoudi@cmail.carleton.ca

Majid Komeili†
School of Computer Science
Carleton University
Ottawa, Canada
majid.komeili@carleton.ca

***Abstract*—We propose a differentiable loss function for learning an embedding space by minimizing the upper bound of the leave-one-out classification error rate of 1-nearest neighbor classification error in the latent space. To evaluate the resulting space, in addition to the classification performance, we examine the problem of finding subclasses. In many applications, it is desired to detect unknown subclasses that might exist within known classes. For example, discovering subtypes of a known disease may help develop customized treatments. Analogous to the hierarchical clustering, subclasses might exist on different scales. The proposed method provides a mechanism to target subclasses in different scales.**

I. Introduction

K-nearest neighbor is one of the most intuitive and interpretable classifiers in that the intuition behind the model is easy to understand; a decision can be understood by its neighbouring samples. However, in many applications such as image classification, k-NN does not work in the original space. The predictive power of the k-NN classifier decreases in high dimensional spaces due to the curse of dimensionality where all pairs of samples have almost the same distance. There has been a lot of efforts in improving the input space of k-NN. Earlier efforts focused on manually designing feature descriptors for extracting a low dimensional space. Examples of such descriptors are Histogram of Gradients (HOG) [1], Local Binary Patterns (LBP) [2] for images and term frequency-inverse document frequency (TF-IDF) [3] for text data.

Instead of manually extracting features from input data, one can learn a representation by projecting the input space to an embedding space. Along this line, large margin nearest neighbor (LMNN) [4] learns a linear transformation by maximizing the margin defined over three samples: an anchor, a sample that has the same label as the anchor (also called a positive sample), and a sample that has a label different than the anchor (also called a negative sample or imposter). LMNN is designed for k-NN but its main limitation is that it can only learn a linear mapping from the input space to an embedding space. More recently in the field of deep learning, deep neural networks have been used to learn a non-linear transformation into an Embedding space. To learn such a transformation of input to a latent space many methods are motivated by various loss functions. More notably, the triplet loss [5] aims to maximize the margin defined over a set of triplets where a triplet consists of an anchor, positive, and negative samples. In [6] the triplet loss is motivated to increase the distance between the anchor and hardest negative (i.e. the nearest negative) while decreasing the distance between the anchor and hardest positive (i.e. farthest positive). The triplet loss is a discontinuous function and shows poor convergence behaviour [6] and often additional steps for triplet selection [6] or constructing batches [7] are required to make a neural network converge with triplet loss. Another issue is that it assumes that classes have a single mode. It does not allow disjoint sets within classes. In many applications, it is essential to learn an embedding space that preserves the subclass or clusters within the classes originally fed to the classifier. Identifying homogeneous subgroups is important because it may enable the researchers to focus on each subgroup and find more efficient and effective treatments specific to that subgroup [8].

In this paper, we propose a loss function to learn an embedding space designed for k-NN that aims to satisfy the above conditions. It learns to distinguish the known classes in a supervised fashion while implicitly improving the clustering behavior within the known classes by minimizing the upper bound of the leave-one-out classification error rate of 1-nearest neighbor classifier in the latent space. We will refer to the proposed loss as multi-scale deep nearest neighbors (MsDNN). There have been several previous works at the intersection of k-NN and neural networks. Deep k-nearest neighbours [9] and neural nearest neighbours (N^3) [10] do not provide a loss function and hence are not comparable with MsDNN. Compared with the soft nearest neighbor (SNN) [11], [12], MsDNN loss is formulated in a large margin framework where sample margins are updated only if the sample is misclassified. MsDNN loss is fully differentiable and can be used with common neural network structures to train end-to-end.

We evaluate the resulting space from two angles. From the classification view, during testing, we run the k-nn classifier and report the classification accuracy. But classification accuracy does not guarantee a good embedding space. Therefore, we run k-means clustering in the embedding space. We hypothesize that if a class consists of several disjoint sets (aka subclasses or clusters), a good embedding space should keep them separate. So, a simple clustering method such as k-means would identify them. Analogous to the hierarchical clustering,

This work is in part supported by the Natural Sciences and Engineering Research Council.

* Equal contributions. † Corresponding author.

MsDNN can be used to target subclasses at different scales.

II. Related work

There has been a number of previous attempts at the intersection of k-NN and neural networks. In the following, we briefly review them and highlight their difference with the proposed method. Papernot and McDaniel introduced Deep k-nearest neighbors (DkNN) [9] which applies an ensemble of k-NN classifiers to the activations at intermediate layers of a neural network to measure uncertainty and improve the robustness against adversarial attacks. DkNN does not offer a loss function. It can be seen as a post-hoc approach for improving the robustness of a neural network classifier against adversarial attacks. The neural network itself needs to be trained separately using common loss functions such as cross-entropy.

MsDNN is different than DkNN in the following ways: a) Unlike MsDNN, DkNN is a post-hoc mechanism that should be applied on top of an already-trained neural network. In contrast, MsDNN is a loss function for training a network. b) The post-hoc mechanism of DkNN requires calibration data in addition to the training data. MsDNN only requires training data. c) The output of DkNN is prediction, confidence, and credibility, whereas the output of MsDNN is an embedding space. d) From the results reported in [9], DkNN gives almost no improvement on classification accuracy compared to the original network.

Plötz and Roth [10] introduced neural nearest neighbors block (N^3 block) for non-local processing which can be used for exploiting self-similarity, for example, in image restoration. However, N^3 does not offer a loss function. Indeed, it relies on a loss function such as cross-entropy for training neural networks augmented with (N^3 block) blocks.

MsDNN is different than N^3 in the following ways: a) In N^3, there is no notion of margin—neither expected margin nor otherwise. b) N^3 does not discuss how the class labels could be used to determine the label of a query sample. For a query sample, the output of the N^3 method is a set of k expected nearest neighbors, and this process has nothing to do with class labels. c) In N^3, the set mentioned above is only incorporated inside a special block called N3Block, which is meant to be interleaved somewhere between layers of a local processing network. In contrast, MsDNN is a loss function that allows learning a "space" where classes can have disjoint subsets. Therefore, N3Block and MsDNN serve very different purposes. d) The experiments in [10] are all about the N3Block, and all comparison methods are about the very specific areas of image denoising and correspondence estimation/classification. In contrast, our experiments are on the general area of deep metric learning, and we compare our method with the related general-purpose methods.

Salakhutdinov and Hinton introduced soft nearest neighbor (SNN) for training neural networks [11]. Recently [12] demonstrated several use cases of the SNN loss. MsDNN is related to the SNN loss in that both compute a notion of the probability of being the nearest sample. However, MsDNN aggregates the features of the neighbors while SNN aggregates the labels of the neighbors. This allows us to explicitly define sample margin and relate it to the expected LOO 1-NN classification error; –i.e. a sample is misclassified if its margin is negative. MsDNN loss maximizes the margin of a sample only if the sample is misclassified. In contrast, SNN does not have such a selective update mechanism.

Relation to metric learning methods: Metric learning methods have been used in many tasks such as face verification [13], [14], image-based search [15], visual tracking [16], [17] and person re-identification [18], [19] where the goal is to learn a metric such that samples from the same class are more similar while samples from opposite classes are less similar. They treat each training class as a monolithic entity and usually lose most of the distinctions of the sub-divisions inside these classes. Most of the recent metric learning methods are based on deep learning, the two most common of which are Siamese networks and triplet loss based networks [20]. Similar to MsDNN, the triplet loss [6] is based on a large margin framework applied to a set of triplets: anchor p_i, positive ${p_i}^+$ and negative ${p_i}^-$. However, an important difference is that the triplet loss is based on the hardest positive and hardest negative samples. The choice of the hardest (farthest) positive instead of the nearest positive (nearest hit) effectively forces the subclasses to collapse into a single mode. Moreover, the triplet loss is a discontinuous function in that a small change in the backbone network's parameters may cause a jump in the loss value due to a change in the hardest positive and/or hardest negative samples of some anchors. This makes it hard to train a neural network with triplet loss [6]. There has been a lot of works on devising strategies for triplet mining [21] or batch construction [7] to alleviate this issue. In contrast, MsDNN is seeking a different direction by using a differentiable formulation that eliminates the need for such workarounds.

III. Proposed Method

Consider an embedding space defined by a mapping function $\mathbf{f}$ parameterized with θ. The margin of an input sample $\mathbf{x}^{(i)}$ in the embedding space can be defined as follows.

$$\mathbf{r}^{(n)}(\theta) = d\left(\mathbf{f}_\theta(\mathbf{x}^{(n)}), \mathbf{f}_\theta(\mathrm{NM}(\mathbf{x}^{(n)}))\right) - d\left(\mathbf{f}_\theta(\mathbf{x}^{(n)}), \mathbf{f}_\theta(\mathrm{NH}(\mathbf{x}^{(n)}))\right) \quad (1)$$

where $\mathrm{NM}(\mathbf{x}^{(i)})$ is the nearest neighbor of $\mathbf{x}^{(i)}$ with a different class label (nearest miss) and $\mathrm{NH}(\mathbf{x}^{(i)})$ is the nearest neighbor of $\mathbf{x}^{(i)}$ with the same class label as $\mathbf{x}^{(i)}$ (nearest hit). $d(\cdot)$ is a distance function. For this paper, we use Euclidean distance. Intuitively, a positive margin for a sample means that the sample will be correctly classified using 1-NN classifier when we leave that sample out. Ideally, the parameters θ should be learned such that $\mathbf{r}^{(n)}(\theta)$ be positive for all samples, i.e. zero LOO 1-NN classification error rate. The margin defined in (1) is quite discontinuous and hence hard to optimize. To alleviate this, following [10], we define the nearest sample

as the expectation over all possible candidates for being the nearest sample and define the expected margin for the n-th sample as follows.

$$\begin{aligned}\bar{\mathbf{r}}^{(n)}(\theta) &= \left\|\mathbf{f}_\theta(\mathbf{x}^{(n)}) - \mathbb{E}_{i\sim\mathcal{M}_n}\mathbf{f}_\theta(\mathbf{x}^{(i)})\right\|_2^2 \\ &\quad - \left\|\mathbf{f}_\theta(\mathbf{x}^{(n)}) - \mathbb{E}_{i\sim\mathcal{H}_n}\mathbf{f}_\theta(\mathbf{x}^{(i)})\right\|_2^2 \\ &= \left\|\mathbf{f}_\theta(\mathbf{x}^{(n)}) - \sum_{i\in\mathcal{M}_n} P(\mathbf{x}^{(i)} = \mathrm{NM}(\mathbf{x}^{(n)})|\theta)\ \mathbf{f}_\theta(\mathbf{x}^{(i)})\right\|_2^2 \\ &\quad - \left\|\mathbf{f}_\theta(\mathbf{x}^{(n)}) - \sum_{i\in\mathcal{H}_n} P(\mathbf{x}^{(i)} = \mathrm{NH}(\mathbf{x}^{(n)})|\theta)\ \mathbf{f}_\theta(\mathbf{x}^{(i)})\right\|_2^2 \quad (2)\end{aligned}$$

$\mathcal{M}_n$ and $\mathcal{H}_n$ denote the set of all possible candidates for $\mathrm{NM}(\mathbf{x}^{(n)})$ and $\mathrm{NH}(\mathbf{x}^{(n)})$ respectively and are defined as $\mathcal{M}_n = \left\{j \in \{1,\dots,M\} \mid y^{(j)} \neq y^{(n)}\right\}$ and $\mathcal{H}_n = \left\{j \in \{1,\dots,M\} \mid y^{(j)} = y^{(n)}, j \neq n\right\}$. $\mathbb{E}_{j\sim\mathcal{M}_n}$ denotes the expectation computed with respect to $\mathcal{M}_n$. $P(\mathbf{x}^{(i)} = \mathrm{NM}(\mathbf{x}^{(n)})|\theta)$ and $P(\mathbf{x}^{(i)} = \mathrm{NH}(\mathbf{x}^{(n)})|\theta)$ are the probabilities of a sample $\mathbf{x}^{(i)}$ being the nearest miss or hit of $\mathbf{x}^{(n)}$, respectively. They can be estimated via the standard kernel density estimation.

$$\begin{aligned}&P(\mathbf{x}^{(i)} = \mathrm{NM}(\mathbf{x}^{(n)})|\theta) \\ &\quad = \frac{k\left(d\left(\mathbf{f}_\theta(\mathbf{x}^{(n)}), \mathbf{f}_\theta(\mathbf{x}^{(i)})\right)\right)}{\sum_{j\in\mathcal{M}_n} k\left(d\left(\mathbf{f}_\theta(\mathbf{x}^{(n)}), \mathbf{f}_\theta(\mathbf{x}^{(j)})\right)\right)}, \quad \forall i \in \mathcal{M}_n \quad (3)\end{aligned}$$

$$\begin{aligned}&P(\mathbf{x}^{(i)} = \mathrm{NH}(\mathbf{x}^{(n)})|\theta) \\ &\quad = \frac{k\left(d\left(\mathbf{f}_\theta(\mathbf{x}^{(n)}), \mathbf{f}_\theta(\mathbf{x}^{(i)})\right)\right)}{\sum_{j\in\mathcal{H}_n} k\left(d\left(\mathbf{f}_\theta(\mathbf{x}^{(n)}), \mathbf{f}_\theta(\mathbf{x}^{(j)})\right)\right)}, \quad \forall i \in \mathcal{H}_n \quad (4)\end{aligned}$$

$k(\cdot)$ is a kernel function. We use the exponent kernel $k(d) = exp(-d/\sigma)$, where σ is a hyper-parameter that determines the resolution at which the data is locally analyzed. Now that the margins are defined, the problem of learning the mapping $\mathbf{f}_\theta$ can be solved within a large margin framework. The two most common margin formulations are based on Hinge loss and logistic loss. We use logistic loss, though experimentally we noticed similar results with Hinge loss as well. Using a logistic loss formulation leads to the following loss function.

$$\mathcal{L}_{\text{Margin}}(\theta) = \sum_{n=1}^{M} \log\left(1 + \exp\left(-\bar{\mathbf{r}}^{(n)}(\theta)\right)\right) \quad (5)$$

M is the number of samples in a batch. By adding the logistic loss, optimization process focuses mostly on increasing the negative margins (i.e. correcting the erroneous samples) rather than further increasing the margin of the samples already have a positive margin. Moreover, since the logistic loss function is an upper bound of the misclassification loss function, up to a difference of a constant factor, our algorithm can be regarded as minimizing the upper bound of the LOO classification error in the embedding space.

Motivated by the success of the autoencoders (AE) in deep embedding clustering [22], [23], we propose to augment the loss function in (5) with an additional term based on the reconstruction loss of an autoencoder. The autoencoder term acts as a regularization and helps the embedding space encode more relevant information. Using an autoencoder defined with $\mathbf{f}_\theta$ and $\mathbf{g}_\phi$ as encoder and decoder respectively, leads to the following loss function.

$$\begin{aligned}\mathcal{L} = \mathcal{L}_{\text{Margin}} + \lambda\mathcal{L}_{\text{AE}} &= \sum_{n=1}^{M} \log\left(1 + \exp\left(-\bar{\mathbf{r}}^{(n)}(\theta)\right)\right) \\ &\quad + \lambda\ell_{\text{rec}}\left(\mathbf{g}_\phi(\mathbf{f}_\theta(\mathbf{x}^{(n)})), \mathbf{x}^{(n)}\right) \quad (6)\end{aligned}$$

where λ is a hyper-parameter that trades the supervised term $\mathcal{L}_{\text{Margin}}$ for the autoencoder loss $\mathcal{L}_{\text{AE}}$. We use mean square error as the reconstruction loss ℓ_{rec} of the autoencoder. Pseudocode of the proposed methods is shown in Algorithm 1.

Input: $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{M}$, σ, λ
Output: $\{\mathbf{f}_\theta, \mathbf{g}_\phi\}$

repeat
  $\mathbf{f}_\theta^{prev.} = \mathbf{f}_\theta$;
  $\mathbf{g}_\phi^{prev.} = \mathbf{g}_\phi$;
  for $n \leftarrow 1$ **to** M **do**
    Compute $\mathbf{p}_{NM}^{(n)}$ using $\mathbf{f}_\theta^{prev.}$ and $\mathbf{g}_\phi^{prev.}$ using (3);
    Compute $\mathbf{p}_{NH}^{(n)}$ using $\mathbf{f}_\theta^{prev.}$ and $\mathbf{g}_\phi^{prev.}$ using (4);
    Compute $\bar{\mathbf{r}}^{(n)}(\theta)$ using (2);
  end
  Compute $\mathcal{L}$ as in (6) and update θ and ϕ using gradient descent.;
until *one epoch is done*;

Algorithm 1: Pseudo code of the proposed method for one epoch.

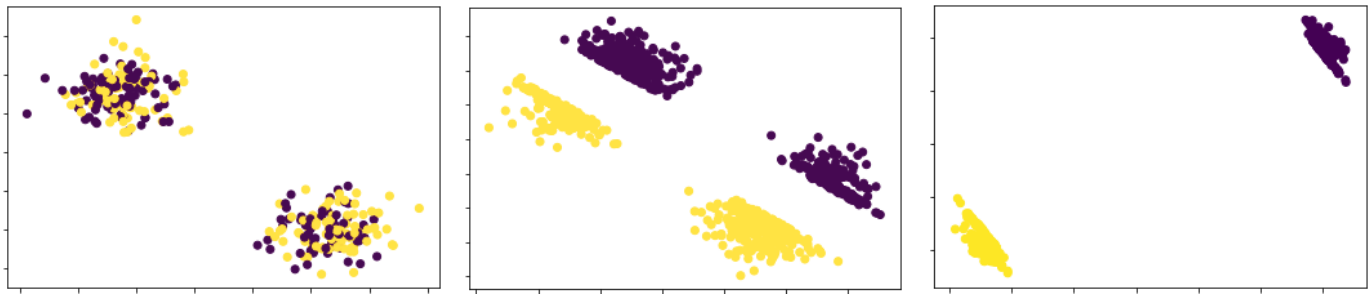
In the extreme case when $\sigma \to 0$, MsDNN will reduce to a triplet selection method where anchors simply pick their nearest positive and nearest negative samples (i.e. using eq. (1) instead of the probabilistic version of margin as in eq. (2). We have implemented such a baseline and in spite of our efforts, it did not converged in most cases. This might be expected since a similar issue has been previously reported in the literature for the popular triplet loss [6]. While the mainstream approach to address this issue is based on carefully designing a strategy for triplet selection [6] or constructing batches [7], we suggest that this issue may be alleviated by replacing the positive and negative samples with their expected values. To validate this, we modified the triplet loss accordingly and run experiments on MNIST. The resulting loss achieved about 99% classification accuracy for a wide range of σ. This is comparable to the performance of the triplet loss with hard negative mining [6] but trains on average about 30% faster and don't need any hard negative mining too.

IV. Experimental Results

In this section, we first demonstrate the ability of the Ms-DNN loss in learning similarities among samples at different scales.

A. MsDNN for learning at different scales

We first present a simulation study on a synthesized dataset. This dataset consisted of 500 points in a 2-dimensional space



(a) (b) (c)

Fig. 1: Effect of minimize the MsDNN loss by applying gradient descent on the input space. (a) 2-dimensional synthetic data with two classes shown in yellow and black. Each class initially has two subclasses. (b) The resulting input space when σ is small. (c) The resulting input space when σ is large.

where the points belong to two classes shown with different colors. The points in each subclass are sampled from a Gaussian distribution as shown in Figure 1 (a). We apply gradient descent to the dimensions of the input space. Through gradient descent, the points are moved in the input space such that the MsDNN loss be minimized. The parameter σ controls the resolution at which samples are locally processed such that their margin is maximized. Figure 1 (b) shows the results when σ is small (here it is set to 1). Likewise, Figure 1 (c) shows the results space when σ is large (here it is set to 25). We want to highlight that there is no backbone network involved in this experiment and the input itself is updated with gradient descent. Intuitively, samples in Figure 1 (a) move in the direction of the gradient and depending on the value of σ they will end up in either Figure 1 (b) or (c). In both cases, the two classes become separated. When σ is small, MsDNN preserves the subclasses. When σ is large, subclasses merge. The proposed method not only learns to separate different classes but also enables us to look for subclasses at different scales.

To further illustrate the effect of σ we designed another experiment using the MNIST[1] dataset. MNIST originally has 10 classes. We define a binary classification problem on it: small numbers versus large numbers. Digits 0 to 4 belong to one class and digits 5 to 9 belong to the other class. This way, we simulate a binary classification problem where each class has five unknown subclasses. We trained a simple convolutional network on this dataset using MsDNN loss. The backbone network is similar to LeNet architecture. See Section IV-D for details. To demonstrate the effectiveness of MsDNN in learning embedding spaces that preserve or merge subclasses (depending on σ), we visualize the resulting embedding space in two ways: 1) using t-SNE to reduce the dimension to two. 2) computing pair-wise symmetric Kullback–Leibler divergence between subclasses. Figure 2 (a) shows the t-SNE plots [24] when σ is small. One may observe that the two classes are well separated and subclasses are also quite visible. For comparison, the ground-truth for the subclasses are shown in Figure 2 (b) using ten different colors. Figure 2 (d) shows the t-SNE plots when σ is large. It can be seen that the two classes are well separated but no distinct subclass exists. The ground-truth for the subclasses are shown in Figure 2 (e) which indicates that the subclasses are blended. We used t-SNE with default parameters as implemented in scikit-learn, a public ML library. Ideally, we would like to see that the distribution of samples from different subclasses overlaps when σ is larges; and likewise, when σ is small, subclasses come from distinct distributions. To verify this, the pair-wise symmetric Kullback–Leibler divergence between subclasses are shown in Figure 2. Figure 2 (c) shows the results when σ is small, and Figure 2 (f) shows the results when σ is large. For example, consider the yellow quarter at the top left corner of Figure 2 (f). This indicates that the symmetric KL divergence between the distribution of the subclasses are small which is in line with the t-SNE plot in Figure 2 (d). Note that the pair-wise symmetric Kullback–Leibler divergence is computed in the embedding space not the 2D space of t-SNE. The dimension of the embedding space is 128.

We further investigate the performance of MsDNN for different values of σ using Normalized Mutual Information (NMI). In addition to the MNIST, we perform experiments on USPS[2], 20NewsGroup[3], and Fashion-MNIST[4]. All datasets except the 20NewsGroup originally come with 10 classes and we simulate a binary classification problem for each dataset by assigning the first five classes to one class and the rest to another class. Each of the resulting datasets (except 20NewGroup) has two classes where each class contains five unknown subclasses. The 20NewsGroup dataset originally has twenty classes categorized into six categories. We pick the six categories as six classes where each class has several unknown subclasses. We train a classifier using MsDNN loss and run k-means in the resulting embedding space and compare the resulting clusters with the ground-truth using NMI. As shown in Figure 3, as σ becomes larger, k-means' ability to detect the unknown subclasses in the embedding space diminishes, suggesting that the subclasses are being merged as the MsDNN

[1]http://yann.lecun.com/exdb/mnist/

[2]https://www.openml.org/d/41070

[3]http://qwone.com/ jason/20Newsgroups/

[4]https://github.com/zalandoresearch/fashion-mnist

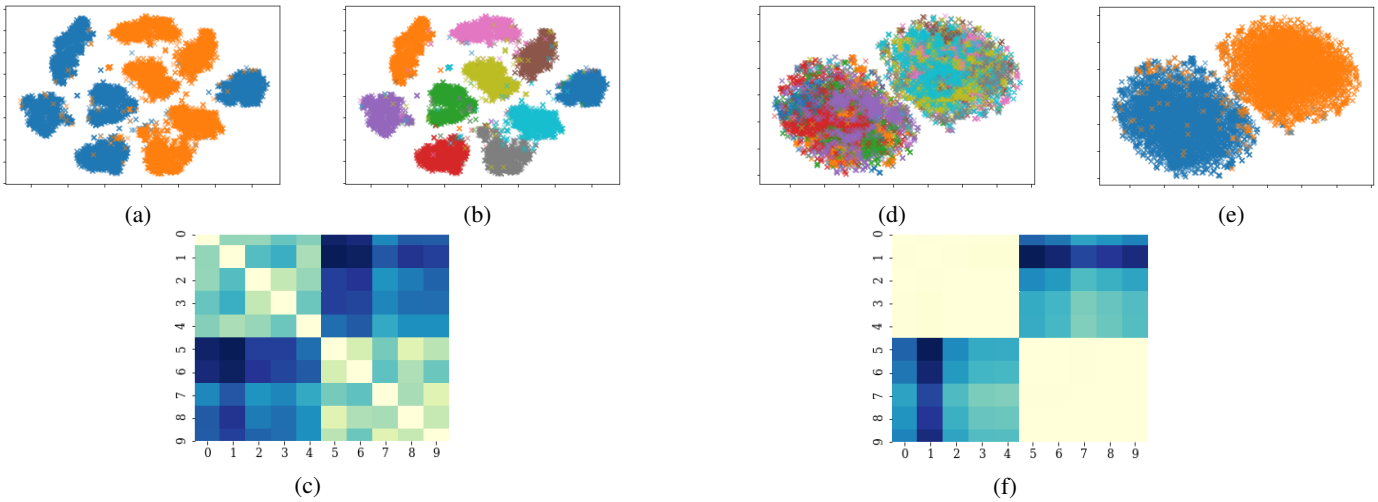


Fig. 2: Visualization of the resulting embedding space for MNIST dataset when $sigma$ is small (a-c) versus when sigma is large (d-f). (a) and (d) show t-SNE plots where samples are colored according to the ground-truth binary labels. (b) and (e) is the same as (a) and (d) respectively, but samples are colored according to the ten ground-truth subclasses. (c) and (d) show the pairwise symmetric KL-divergence between the ten subclasses. Lighter colors denote lower KL-divergence. The dimension of the embedding space is 128 and the value of σ for (a-c) and (d-f) are 0.03 and 1 respectively.

processes data with a coarser resolution. Here, the parameter k in k-means is set to the number of clusters/subclasses i.e. twenty for the 20NewsGroup and ten for the rest of the datasets.

B. Evaluating the embedding space

In general, a good embedding space should learn the similarity structures among samples. From a supervised learning perspective, classes should be well separated from each other. Moreover, beyond the class labels, the unknown subclasses should also be preserved. Here subclasses are the clusters within the classes. As a result, we evaluate our models from two different angles: Clustering performance and classification performance.

Clustering performance: We consider the k-means' ability in detecting subclasses as a measure of how well the subclasses are separated in an embedding space. We compare the proposed methods with the metric learning methods including triplet loss [6], contrastive loss [25], SNN loss [11] and lifted structure loss [7]. Table I shows the NMI and clustering accuracy of k-means where the parameter k in k-means is set to the number of subclasses i.e. twenty for the 20NewsGroup and ten for the rest of the datasets. It can be seen that, on average, the proposed methods outperform all comparison methods in both NMI and clustering accuracy. Also, the addition of the autoencoder term to the loss function has improved the results compared with the original MsDNN. We used the same backbone networks for all methods. For DkNN, we trained the networks with cross-entropy loss and applied the DkNN mechanism on top. For N3 [10], we augmented the same backbone network with the N3 block and trained with cross-entropy loss. Note that in both cases, by embedding space, we mean the layer before the final layer and it has the same dimension as other metric learning methods (more implementation details in Section IV-D).

Comparison with deep clustering methods: MsDNN learns an embedding space to distinguish different classes while allowing disjoint subclasses. Therefore, a shallow clustering method such as k-means can be used in the resulting embedding space to discover subclasses. Since we measure the clustering performance as a proxy to the true similarities between samples, the reader might be interested to know how clustering methods and specially deep clustering methods perform. Deep clustering methods learn an embedding space such that samples form tight clusters. We compare with Deep Embedding Clustering (DEC) [23] and Deep Clustering Networks (DCN) [22]. There is another category of clustering methods known as semi-supervised clustering that use some auxiliary information, often in the form of *cluster* labels of a small portion of samples, to facilitate the clustering process. Since such fine-grained labels are not available in our scenario, we cannot compare our method with them. However, a subcategory of the semi-supervised clustering methods is based on forming a set of pairwise constraints in the form of must-link and cannot-link pairs [26]–[31]. In a classification setting, we know that samples from opposite classes should not be in the same cluster. So, we can assume a cannot-link between them. Along this line, we compare with Semi-supervised Deep Embedded Clustering (SDEC) [32]. Note that most of the existing deep semi-supervised clustering methods [29]–[31], [33], [34] requires both must-link and cannot-link information to function, and therefore we cannot compare with them. Table II shows the NMI and clustering accuracy for the comparison clustering methods. It can be seen that, on average, the proposed methods outperform them in both NMI and clustering accuracy. Unlike metric learning methods, clustering

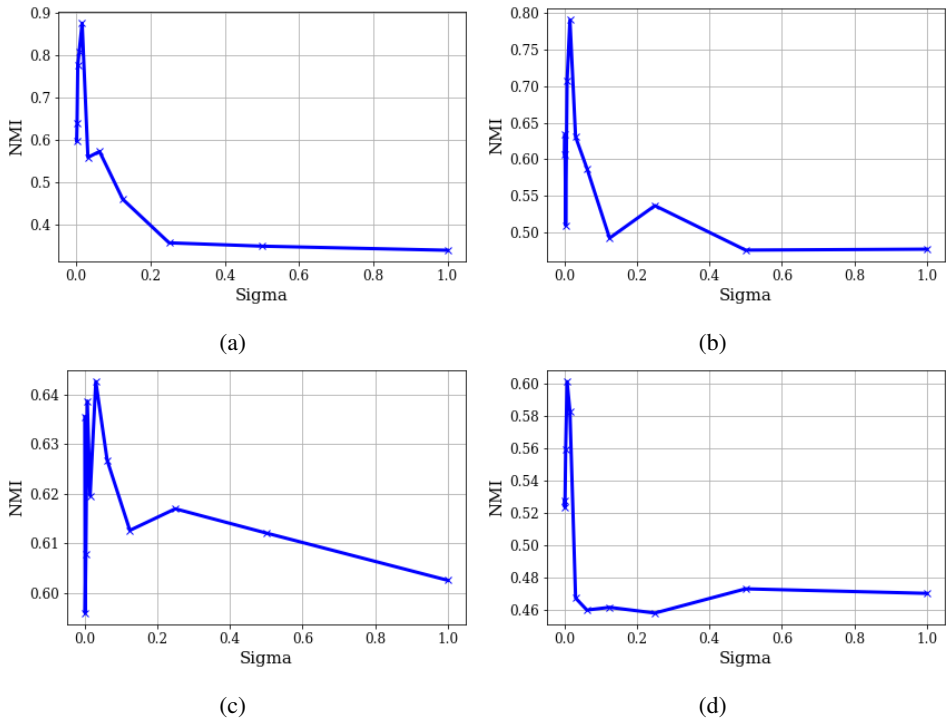


Fig. 3: NMI for different values of σ. (a) MNIST. (b) USPS. (c) 20NewsGroup. (d) F-MNIST.

methods do not use the coarse class labels, but instead they are equipped with mechanisms specially designed for the harder task of detecting clusters in data. For scientific rigor, we compare with them. The proposed methods, despite being simple, outperform the deep embedded clustering methods.

Classification performance: We use a k-NN classifier to measure how well different classes are separated in the embedded space. Different values for k are tested from among 1, 3, 5, and 7 and the results on the test set for the best k are reported in Table III. Note that this is a six-way classification for the 20NewsGroup and binary classification for the rest of the datasets. It can be seen that the classification performance of the proposed methods, are comparable or better than the comparison methods. The two methods with closer classification accuracy to the proposed methods are N^3Net and Triplet loss. While the proposed methods still have 1-2% better classification accuracy, the actual merit of the proposed methods is in their superior clustering accuracy which is, on average, approximately twice better than N^3Net and Triplet loss (34.1% and 31.77% for N^3Net and Triplet loss versus 64.06% and 67.7% for MsDNN and MsDNN-AE). This is analogous to the two scenarios illustrated in Figure 2 where in both scenarios the coarse classes are well separated while subclasses are separated (on the left) and mixed (on the right). Considering both clustering and classification accuracy, we conclude that the proposed methods have learned an embedded space that preserves the similarities between samples better than the comparison methods.

Probabilistic margin versus deterministic margin: The choice of the expected margin $\bar{\mathbf{r}}^{(n)}(\theta)$ as in (2) over $\mathbf{r}^{(n)}(\theta)$ as in (1) is crucial. To show the effectiveness of the expected margin $\bar{\mathbf{r}}^{(n)}(\theta)$, we tried to train the networks using the deterministic margin $\mathbf{r}^{(n)}(\theta)$, however in most cases the networks with deterministic margin did not converge.

Fair comparison in Experiments We used the same backbone network in all experiments for both classification and clustering tasks, except for DEC and DCN, where we report the results from other papers. However, our backbone network indeed has fewer parameters than DCN/DEC. All datasets come with their standard train/test splits. For each dataset, we do an internal 5-fold cross-validation on the standard train set to select the best hyperparameters based on the *classification* accuracy. We then train on the entire standard train set using the selected hyperparameters and calculate classification accuracy on the test set. We want to highlight that for evaluating clustering performance, there is no such a notion of train/test splits. None of the comparison clustering methods have been considered a separate test set. This is a common practice in the clustering community [23]. Nevertheless, we want to highlight that MsDNN is *not* a clustering method, and we do not tune for clustering performance. We do model selection based on classification performance via cross-validation.

C. Computational complexity

For a batch of M samples, MsDNN loss has M margin terms and for each $2(M-1)$ distances should be computed. Therefore, following [38] the overall complexity of the MsDNN is $O(n^2)$ where n is the number of data points. This is better than the time complexity of the unmodified triplet loss which is $O(n^3)$ [38]. In practice, we found that the training

TABLE I: For comparison with supervised methods. Evaluation of the proposed method on the train set after using cross validation for parameter selection.

	MNIST		USPS		20NewsGroup		F-MNIST	
Methods	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC
SNN	**89.05**	93.97	46.58	55.50	59.38	**32.95**	59.89	49.86
Triplet loss	38.01	27.60	38.38	38.54	57.45	30.59	38.28	30.38
Contrasive loss	35.57	28.06	49.11	51.55	61.84	32.61	42.39	27.64
Lifted Structure loss	31.67	25.07	20.92	28.56	57.40	31.87	39.10	33.21
N^3Net	46.44	38.85	37.89	36.29	61.35	31.92	45.04	29.37
DkNN	85.59	81.44	57.33	49.84	59.15	29.38	51.26	43.59
MsDNN	87.49	94.12	79.03	78.61	61.96	32.27	60.13	51.27
MsDNN-AE	88.09	**94.40**	**80.13**	**81.12**	**64.34**	32.34	**62.60**	**63.04**

TABLE II: For comparison with clustering methods. Evaluation of the proposed method on the train set after using cross validation for parameter selection.

	MNIST		USPS		20NewsGroup		F-MNIST	
Methods	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC
K-means	49.07	51.30	63.83	67.65	35.70	26.47	51.19	47.38
DEC	80	84	75.29	74.08	$45.36^{\dagger}$	$\mathbf{50.11}^{\dagger}$	$54^{\ddagger}$	$51^{\ddagger}$
DCN	81	83	68.3	68.8	$48^{\ddagger}$	$44^{\ddagger}$	$55.8^{\S}$	$50.1^{\S}$
SDEC	74.58	71.07	43.23	50.04	7.76	9.92	50.80	51.58
MsDNN	87.49	94.12	79.03	78.61	61.96	32.27	60.13	51.27
MsDNN-AE	**88.09**	**94.40**	**80.13**	**81.12**	**64.34**	32.34	**62.60**	**63.04**

The results denoted by (†), (§), (‡), (††) are reported from [32], [35], [36], and [37] respectively.

TABLE III: Evaluation of the classification accuracy of the proposed method. Here, 20NG stands for the 20NewsGroup dataset. The last column, AVG, is the average classification accuracy over all datasets.

	MNIST	USPS	20NG	F-MNIST	AVG
SNN	99.19	85.95	90.56	94.64	92.59
Triplet	99.24	95.15	87.47	94.80	94.17
Lifted Structure	99.30	70.21	69.59	94.75	83.46
Contrastive	99.04	92.09	88.72	94.03	93.47
N^3Net	98.94	96.73	90.59	94.62	95.22
DkNN	96.83	85.2	89.72	89.72	90.37
MsDNN	99.23	97.06	90.76	94.27	95.33
MsDNN-AE	99.08	96.91	91.79	93.77	**95.39**

time of our method is similar to that of the triplet loss and SNN loss. For example, the average time per epoch on the MNIST dataset is 23, 33, and 21 seconds respectively for MsDNN loss, triplet loss, and SNN loss using a P100 GPU and four CPU cores. That is because the computational time for computing the gradient of the loss is negligible compared with the time needed to update the parameters of the backbone network. MsDNN-AE took 38 seconds per epoch. We run experiments for up to 100 epochs except for the USPS dataset where we run for 500 epochs.

D. Implementation details

For MNIST, and F-MNIST we used a neural network with six convolutional layers followed by one fully connected layer. For the USPS dataset, we used a similar architecture, but with fewer filters. For the 20Newsgroup dataset, we used a feed-forward network with 11 layers. The data was vectorized using TF-IDF [3] with a maximum word count of 75000. For MsDNN-AE, the decoder for MNIST, F-MNIST, and USPS consisted of one fully connected layer followed by six deconvolution layers. For 20NewsGroup, the decoder is simply a mirror of the encoder's layers. The embedding layer was of size 128 for MNIST, and F-MNIST and 32 for other datasets. The hyper-parameters σ and λ are searched among $\{2^{-10}, 2^{-9}, \cdots 2^{10}\}$, and $\{0.1, 0.5, 1.0, 10\}$ respectively. For the sake of fair comparison, for the temperature parameter of the SNN loss, we searched the same range as in σ in MsDNN. The parameter k is searched among $\{1, 3, 5\}$ for N^3Net. The parameters are selected based on 5-fold cross-validation. The train/test splits come originally with each dataset. We have implemented MsDNN in TensorFlow. Our code is included as supplementary material and will be made publicly available upon publication.

V. Conclusions and Future Work

In this paper, we offered MsDNN, a differentiable loss that minimizes the upper bound of the leave-one-out classification error rate of 1-nearest neighbor classifier in the latent space. MsDNN is formulated in a large margin framework where the sample margins are defined based on the expected nearest hit (same label) and expected nearest miss (opposite label) samples. MsDNN aims to maximize the sample margins so that all samples have a positive margin. The output of the MsDNN is an embedding space. In our experiments, k-NN and k-means were used for evaluating the utility of the resulting space. We empirically demonstrated that the embedding space learned by MsDNN can preserve the relationship between

samples to discover subclasses while separating classes that were originally given during training.

A direction for further research could be to use MsDNN to find subclasses in applications such as diagnosing autism spectrum disorder (ASD) or depression, where a large variability has been observed among the patients [39]. It has been suggested that such variability can be associated with the existence of some unknown homogeneous subgroups [8], [39], [40]. Identifying homogeneous subgroups is important because it may enable researchers to focus on each subgroup and find more efficient and effective treatments specific to that subgroup [8].

REFERENCES

[1] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, vol. I, pp. 886–893, 2005.

[2] M. Pietikäinen, A. Hadid, G. Zhao, and T. Ahonen, *Computer vision using local binary patterns*. Springer Science & Business Media, 2011, vol. 40.

[3] J. Ramos *et al.*, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, vol. 242. New Jersey, USA, 2003, pp. 133–142.

[4] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, pp. 207–244, 2009.

[5] ——, "Distance metric learning for large margin nearest neighbor classification." *ournal of Machine Learning Research 10, no. 25*, 2009.

[6] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[7] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese, "Deep metric learning via lifted structured feature embedding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4004–4012.

[8] J. Ellegood, E. Anagnostou, B. Babineau, J. Crawley, L. Lin, M. Genestine, E. Dicicco-Bloom, J. Lai, J. Foster, O. Penagarikano *et al.*, "Clustering autism: using neuroanatomical differences in 26 mouse models to gain insight into the heterogeneity," *Molecular psychiatry*, vol. 20, no. 1, pp. 118–125, 2015.

[9] N. Papernot and P. McDaniel, "Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning," *arXiv preprint arXiv:1803.04765*, 2018.

[10] T. Plötz and S. Roth, "Neural nearest neighbors networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 1087–1098.

[11] R. Salakhutdinov and G. Hinton, "Learning a nonlinear embedding by preserving class neighbourhood structure," in *Artificial Intelligence and Statistics*, 2007, pp. 412–419.

[12] N. Frosst, N. Papernot, and G. Hinton, "Analyzing and improving representations with the soft nearest neighbor loss," *arXiv preprint arXiv:1902.01889*, 2019.

[13] J. Hu, J. Lu, and Y.-P. Tan, "Discriminative deep metric learning for face verification in the wild," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1875–1882.

[14] Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification," in *Advances in neural information processing systems*, 2014, pp. 1988–1996.

[15] P. Wu, S. C. Hoi, H. Xia, P. Zhao, D. Wang, and C. Miao, "Online multimodal deep similarity learning with application to image retrieval," in *Proceedings of the 21st ACM international conference on Multimedia*, 2013, pp. 153–162.

[16] J. Hu, J. Lu, and Y.-P. Tan, "Deep metric learning for visual tracking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 11, pp. 2056–2068, 2015.

[17] H. Li, Y. Li, and F. Porikli, "Deeptrack: Learning discriminative feature representations online for robust visual tracking," *IEEE Transactions on Image Processing*, vol. 25, no. 4, pp. 1834–1848, 2015.

[18] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Deep metric learning for person re-identification," in *2014 22nd International Conference on Pattern Recognition*. IEEE, 2014, pp. 34–39.

[19] S. Liao, Y. Hu, X. Zhu, and S. Z. Li, "Person re-identification by local maximal occurrence representation and metric learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2197–2206.

[20] J. Lu, J. Hu, and J. Zhou, "Deep metric learning for visual understanding: An overview of recent advances," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 76–84, 2017.

[21] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu, "Learning fine-grained image similarity with deep ranking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1386–1393.

[22] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards k-means-friendly spaces: Simultaneous deep learning and clustering," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3861–3870.

[23] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *International conference on machine learning*, 2016, pp. 478–487.

[24] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[25] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, vol. 2. Lille, 2015.

[26] S. Basu, A. Banerjee, and R. Mooney, "Semi-supervised clustering by seeding," in *In Proceedings of 19th International Conference on Machine Learning (ICML-2002*. Citeseer, 2002.

[27] B. Kulis, S. Basu, I. Dhillon, and R. Mooney, "Semi-supervised graph clustering: a kernel approach," *Machine learning*, vol. 74, no. 1, pp. 1–22, 2009.

[28] W. Tang, H. Xiong, S. Zhong, and J. Wu, "Enhancing semi-supervised clustering: a feature projection perspective," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2007, pp. 707–716.

[29] G. Chen, "Deep transductive semi-supervised maximum margin clustering," *arXiv preprint arXiv:1501.06237*, 2015.

[30] D. Ienco and R. G. Pensa, "Deep triplet-driven semi-supervised embedding clustering," in *International Conference on Discovery Science*. Springer, 2019, pp. 220–234.

[31] L. A. Vilhagra, E. R. Fernandes, and B. M. Nogueira, "Textcsn: a semi-supervised approach for text clustering using pairwise constraints and convolutional siamese network," in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, 2020, pp. 1135–1142.

[32] Y. Ren, K. Hu, X. Dai, L. Pan, S. C. Hoi, and Z. Xu, "Semi-supervised deep embedded clustering," *Neurocomputing*, vol. 325, pp. 121–130, 2019.

[33] M. Śmieja, Ł. Struski, and M. A. Figueiredo, "A classification-based approach to semi-supervised clustering with pairwise constraints," *Neural Networks*, 2020.

[34] A. Shukla, G. S. Cheema, and S. Anand, "Semi-supervised clustering with neural networks," *arXiv preprint arXiv:1806.01547*, 2018.

[35] Q. Zhu and Z. Wang, "An image clustering auto-encoder based on predefined evenly-distributed class centroids and mmd distance," *Neural Processing Letters*, pp. 1–16, 2020.

[36] S. E. Chazan, S. Gannot, and J. Goldberger, "Deep clustering based on a mixture of autoencoders," in *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, 2019, pp. 1–6.

[37] X. Li, H. Yin, K. Zhou, and X. Zhou, "Semi-supervised clustering with deep metric learning and graph embedding," *World Wide Web*, pp. 1–18, 2019.

[38] T.-T. Do, T. Tran, I. Reid, V. Kumar, T. Hoang, and G. Carneiro, "A theoretically sound upper bound on the triplet loss for improving the efficiency of deep distance metric learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 404–10 413.

[39] R. K. Lenroot and P. K. Yeung, "Heterogeneity within autism spectrum disorders: what have we learned from neuroimaging studies?" *Frontiers in human neuroscience*, vol. 7, p. 733, 2013.

[40] H. Witt, S. C. Mack, M. Ryzhova, S. Bender, M. Sill, R. Isserlin, A. Benner, T. Hielscher, T. Milde, M. Remke *et al.*, "Delineation of two clinically and molecularly distinct subgroups of posterior fossa ependymoma," *Cancer cell*, vol. 20, no. 2, pp. 143–157, 2011.