Diffusion-Based Sampling for Deep Active Learning

Dan Kushnir
Bell Laboratories, NOKIA
Murray Hill, New Jersey 07094, USA
Email: dan.kushnir@nokia-bell-labs.coml

Luca Venturi
Meta
New York, NY 10003, USA
Email: luca.venturi.92@gmail.com

***Abstract*—The remarkable performance of deep neural networks depends on the availability of massive labeled training data. To alleviate the load of data annotation with labels, deep active learning aims to sample a minimal set of training points to be labelled which yields maximal model accuracy. We propose an efficient sampling criterion to sample data for annotation, which automatically shifts from an exploration type of sampling to a class-decision-boundary refinement. Our criterion relies on a process of diffusing the existing label information over a graph constructed from the hidden representation of the data. This graph representation captures the intrinsic geometry of the approximated labeling function. We analyze our sampling criterion and its exploration - refinement transition in light of the eigen-spectrum of the diffusion operator. Additionally, we provide a comprehensive sample complexity analysis that captures the two phases of exploration and refinement. The diffusion-based sampling criterion is shown to be advantageous over state-of-the-art criteria for deep active learning on synthetic and real benchmark data.**

I. Introduction

Deep learning has provided unprecedented performance in various semi-supervised learning tasks ranging from speech recognition to computer vision and natural language processing. Deep Convolutional Neural Networks (CNN), in particular, have demonstrated object recognition that exceeds human's performance. However, this success comes with the requirement for massive amounts of labelled data. While data collection at large scale has become easier, its annotation with labels has become a bottleneck for execution in many real-life problems.

Active learning provides a plethora of techniques that allow to sample a set of data points for labeling, which optimally minimizes the error probability under a fixed budget of a labeling effort (see [22] for review). A well known trade-off in active learning is between the exploration and refinement sampling strategies. Exploration aims at mapping the joint data and label distribution in order to identify decision boundaries, and typically yields higher model accuracy at the earlier stage of active learning over other baseline criteria. Refinement (also referred to as exploitation), on the other hand, samples labels at the proximity of a discovered class decision-boundary, improving its localization. Active learning exhibits improved performance when the balance between exploration and refinement is optimal.

Incorporating active learning into deep learning is yet a challenging task for several reasons. First, the network representation does not allow to construct a simple probabilistic sampling criterion that incorporates exploration and refinement. Thus, state-of-the-art focus on either exploration or on a refinement type criteria, which often leads to sub-optimal results. This state drives active deep learning methodology to focus mostly on simple refinement-type uncertainty sampling criteria (e.g. [8, 23]), which do not require using the classifier at the querying stage. Or, on using geometric criteria (e.g. [21]) that only explores the data distribution in its feature space.

We propose a graph Diffusion-based Deep Active Learning (DDAL) criterion. We utilize the graph diffusion process, learnt on a graph constructed from the penultimate layer, to derive our active sampling criterion. In trained networks, the penultimate representation is highly correlated with the labeling function, and hence provides an optimal representation to sample the labeling function. The main contributions of DDAL are:

1) **Exploration-refinement trade-off**: We derive a sampling criterion which exhibits an automatic switch from exploration to refinement-based sampling. Our criterion explores graph nodes that are unreachable via diffusion, and then refines the boundary where diffusion-derived labels are ambivalent. This natural switch avoids the need in active learning for an additional optimization machinery (e.g. [25]) to decide which criterion type to use. We analyze the exploration-refinement switch in light of the eigen-spectrum of the diffusion iterant. We provide a sketch of the application of our method in Fig 1.
2) **Sample complexity analysis**: We provide sample complexity analysis for the two phases of exploration and refinement. The analysis depicts complexity that is logarithmic in N, and linear in the decision boundary set.
3) **Empirical validation:** DDAL competitive performance is demonstrated empirically on benchmark data and compared with eight state-of-the-art active learning criteria. In particular, those constructed for deep learning.

II. Problem Setup

Let $\mathcal{D}$ be a probability distribution over $\mathbb{R}^d \times [C]$, where $[C] = \{1, ..., C\}$ and $C \geq 2$ indicates the number of classes. Our aim is to find a classifier $f : \mathbb{R}^d \to \Delta_C$ (Δ_C denotes the space of probability measures over $[C]$) that minimizes the classification error probability

$$\mathcal{E}(f) = \mathbb{P}_{(x,y)\sim\mathcal{D}}\left\{(\arg\max_{c\in[C]} f_c(x)) \neq y\right\}.$$

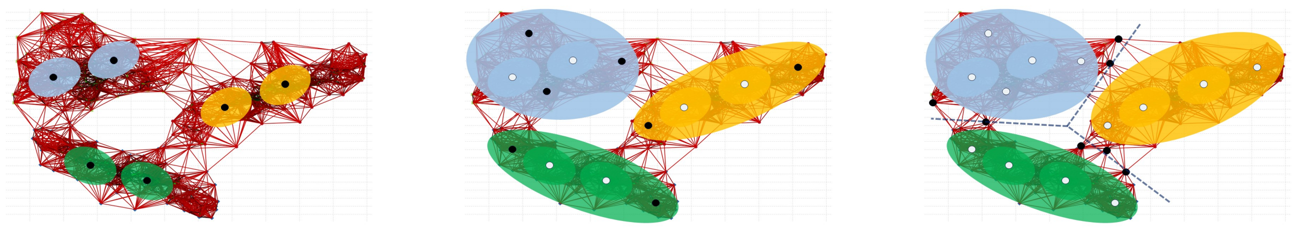


Figure 1: **Schematic evolution of diffusion-based active sampling on the graph $G(V, E, W)$ constructed on the penultimate layer**. The graph in red represents the constructed graph over all points $X_\ell \cup X_{\text{pool}}$ Black points - current samples, white points - previous samples, ovals - area of low uncertainty in class label for 3 different color coded labels. Left: active sampling in early exploration stage. Center: advanced exploration stage of sampling captures the label related clusters. Right: Refinement (exploitation) stage: black samples focus on the decision boundary between class where highest uncertainty prevails.

In deep learning, we consider the parametric functions

$$f_\theta = g_{\theta_{n+1}} \circ h^n_{\theta_n} \circ \cdots \circ h^1_{\theta_1},$$

where $h^i_{\theta_i} : \mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i}$ (with $d_0 = d$) and $g_{\theta_{n+1}} : \mathbb{R}^{d_n} \to \Delta_C$. The parameter $\theta = \{\theta_i\}_i$ defines our final model f_θ.

Active learning setting. In active learning, one is given a *pool* of input data points $X_{\text{pool}} = \{x_i\}_{i=1}^N$, and a budget Q to select for labeling. Optionally, a labelled subset $X_\ell \subset X_{\text{pool}}$ may also be given as input. In the following, we denote $X_u \doteq X_{\text{pool}} \setminus X_\ell$ as the set of unlabeled points in X_{pool}. The aim of active learning is to minimize the error $\mathcal{E}(f_\theta)$ while querying only Q points from X_{pool}, and training f_θ with them.

III. CLASSIFICATION VIA GRAPH DIFFUSION

We use the optimized latent representation to construct a KNN-based weighted proximity graph $G = (V, E)$, where

$$W_{ij} = m\left(-\frac{\rho(f^n_\theta(x_i), f^n_\theta(x_j))}{\sigma_{ij}}\right) I\{j \in N(i)\} \qquad (1)$$

with m as a similarity metric, and ρ as a distance metric. The metric $\rho(f^n_\theta(x_i), f^n_\theta(x_j)) = \|f^n_\theta(x_i) - f^n_\theta(x_j)\|_2^2$, $\sigma_{ij} = \max_{j \in N(i)} \rho(f^n_\theta(x_i), f^n_\theta(x_j))$, and $N(i)$ are the K neighbours of $f^n_\theta(x_i)$. The transition matrix is defined as

$$M \doteq D^{-1}W, \qquad (2)$$

where D is diagonal with $D_{ii} = \sum_j W_{ij}$.

Label diffusion can be considered as a Markov process to propagate the label information of X_l to X_u. Specifically, the one-step transition probability between states x_i and x_j is given by $p_{ij} = \mathbb{P}\{x_i \to x_j\} = M_{ij}$. We employ a random walk on G as a mean to assign a label to $x_i \in X_u$. The predicted label of x_i is associated with the probability of arriving to a labeled point $f^n_\theta(x)$ of class 1 (w.l.o.g.) after performing a t-step random walk starting at $f^n_\theta(x_i)$. Marking this probability as $p_t(y(x) = 1|i)$, it can be derived by the recursive relation

$$p_t(y(x) = 1|i) = \sum_j p_{t-1}(y(x) = 1|j)p_{ij}. \qquad (3)$$

More formally, we associate the random walk probability $p_t(y(x) = 1|i)$ with the classification probability $p(y(x_i) = 1|x_i)$. For labeled points $x \in X_\ell$, $p(y(x) = y|i) = 1$.

Denoting $2p_t(y(x) = 1|i) - 1$ by χ_i we observe that $\chi_i \in [-1, 1]$ and its sign can be used to generate binary labels. Denote

$$D = \begin{pmatrix} D_{ll} & 0 \\ 0 & D_{uu} \end{pmatrix}, W = \begin{pmatrix} W_{ll} & W_{lu} \\ W_{ul} & W_{uu} \end{pmatrix}.$$

In matrix form $\chi_u = [D^{-1}_{uu}W_{ul}|D^{-1}_{uu}W_{uu}]\begin{pmatrix} \chi_l \\ \chi_u \end{pmatrix}$.

Let the graph laplacian $L = D - W$ in the system, then

$$L_{uu}\chi_u = W_{ul}\chi_l \iff L_{uu}X_{uu} = -L_{ul}y_l. \qquad (4)$$

(4) above can be solved via the iteration:

$$\chi_i^{(t+1)} = \frac{1}{L_{uu,ij}}\left(-(L_{ul}y_l)_i - \sum_{j \neq i} L_{uu,ij}\chi_j^{(t)}\right). \qquad (5)$$

(5) is transducing a label χ_i^{t+1} to χ_i as a weighted average of the labels of its neighbors with the transition probabilities as weights. The labels are propagated to X_u gradually for t steps. At the t-th step,

$$\chi_{ic}^{(t)} = \begin{cases} 1 & \text{if } x_i \in X_l \ \& \ c = y_i \\ -1 & \text{if } x_i \in X_l \ \& \ c \neq y_i \\ 0 \ if \ t = 0, \ else \ (M\chi_{:,c}^{(t-1)})_i & \text{if } x_i \in X_u. \end{cases}$$

We provide the multi-class case in the supplementary material.

IV. ACTIVE LEARNING

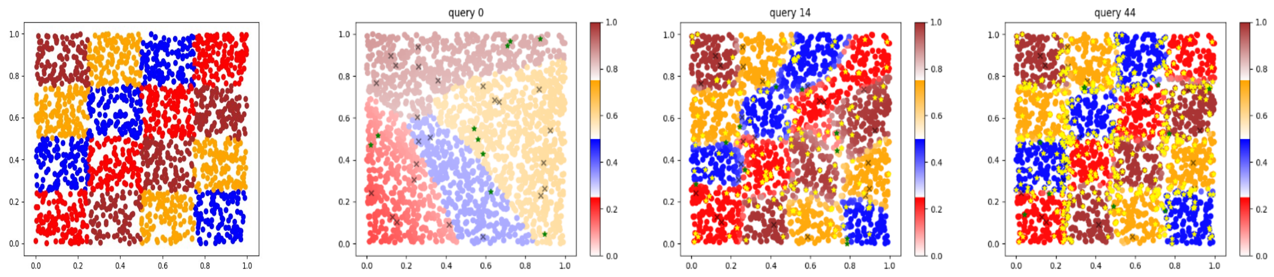
We begin with introducing our query selection criterion and prove its important transition from exploration to refinement.

A. Query criterion.

$\chi^{(T)}$ values can be used realize the *uncertainty* on whether vertex $i \in \{i_1, ..., i_{|pool|}\}$ belongs to class c. Specifically, Specifically, the magnitude can be used to select a new batch to query as

$$\hat{X} = \arg\min^B_{i \in X_u} \min_{c \in [C]} \left|\chi_{c,i}^{(T)}\right|, \qquad (6)$$

where $\min^B$ denotes the B smallest elements. The selection criterion in (6) fits the one-vs-all approach above. In supplementary material other variants are suggested.



Ground truth First query batch 14 query batches 44 query batches

Figure 2: Exploration & Refinement in Multiclass Checkerboard. Left to right: Ground truth, 1st batch (exploration, 30% accuracy), 14 batches (exploration, 82% accuracy) and 44 batches (Refinement, 95% accuracy). Color coding corresponds to label assignment and intensity to magnitude of weighted labels. Light yellow points represent so-far queried points, black points represent current query. Transition from exploring the distribution in (0-14 batches) to focus on boundaries (44) batches

B. *Exploration and Refinement.*

The query criterion (6) enables the exploration of the data set at early stages of active learning, and a switch to refinement when exploration has saturated. This observation is based on the convergence of the diffusion iterant χ^t to the second eigenvector ϕ_2 of the graph's Laplacian as $t \to \infty$. We show in our analysis below that the 2nd eigenfunction can be used to sample data for labeling with an optimal exploration-refinement trade-off.

To realize this, we observe that at early stages of label acquisition low magnitude entries in χ correspond to data points that are unreachable from the training set via diffusion and need to be explored. At later stages all unlabelled data points $f_\theta^n(X_u)$ are reachable via diffusion from the labelled set $f_\theta^n(X_l)$. At this stage low magnitude χ entries correspond to the transition between the two classes -1 and 1. These nodes capture the eigenvector's transition from negative to positive entries. Therefore, sampling these points corresponds to the refinement of the decision boundary. We provide the convergence result to the second eigenvector of the graph Laplacian in Lemma (1):

Lemma 1. *Let $\lambda_1, ..., \lambda_n$, $\phi_1, ..., \phi_n$ be the solutions to the system: $L\phi = (1 - \lambda)D\phi$. Then $\chi^{(t)}$ converges to ϕ_2 via the iteration (10) with M, as $t \to \infty$.*

The eigenvector ϕ_2 is a solution to the relaxed Minimal Normalized Cut problem (*MinNCut*) [5] in a graph $G(V, E, W)$ constructed from data representation in the hidden layer f_θ^n:

$$MinNCut = \arg\min_{h:h^T D\mathbf{1}=0} \frac{h^T L h}{h^T D h}. \tag{7}$$

Eq. (12) provides an additional view point: at the limit, the eigenfunction ϕ_2 is approximately piece-wise constant with opposite sign on each of the two blocks corresponding to the negative and positive labels [5]. Therefore the criterion (6) captures first the unexplored data points whose label magnitude is 0 (exactly). Once diffusion from the labelled set reaches all nodes the eigenfunction has full support. At this stage the criterion (6) switches to sampling at the eigenfunction's inflection point and refines the decision boundary.

Algorithm 1 DDAL

```
input: data $X_{\text{pool}} = \{x_i\}_{i=1}^N$, labeled subset $X_\ell^0$, model $f_\theta$ trained on $X_\ell^0$, $Q$, $B$, $M$, $T$
while budget $Q$ is not exhausted: do
    compute $G = (V, E, W)$ using $\{f_\theta^n(x_i)\}_{i=1}^N$
    for $i = 1$ to $R = \frac{Q}{B}$ do
        initialize $\chi^{(0)}$ according to $X_\ell^0 \cup \hat{X}_\ell^{i-1}$
        for $t = 1$ to $T$ do
            $\chi^{(t)} = M\chi^{(t-1)}$
            $\chi^{(t)} = \chi^{(t-1)}|_{(X_\ell^0 \cup \hat{X}_\ell^{i-1})}$
        end for
        query $\mathcal{S} = \arg\min^B_{k \in \hat{X}_u^i} \min_{c \in [C]} \left|\chi_{c,k}^{(T)}\right|$
        update $\hat{X}_\ell^i = \mathcal{S} \cup \hat{X}_\ell^{i-1}$
    end for
    retrain $f_\theta$ with $\hat{X}_\ell^i$
end while
```

We demonstrate switching from exploration to refinement on a checkerboard example - Fig. 2. Exploration is observed to become saturated, as the joint data-label distribution is sufficiently mapped. At that stage DDAL automatically switches to refining the decision boundaries. We provide the pseudo code of DDAL in algorithm 1.

We note that iterations of (5) are only run for a few time steps instead of until full convergence. In exploration stage labels are sampled to reduce the diffusion time to convergence by cutting the distance between the training points. In particular, in the exploration stage the 0-label magnitude corresponds to points unreachable from the training set, and therefore the distance to the nearest training point is reduced by at least T hops in the random walk with each exploratory label acquisition.

C. *Running time and parameter selection*

The running time analysis is composed of three parts. The first part includes the computation of the K-NN graph. The K-NN proximity search computational cost can be reduced from the naive search cost by using procedures for K-NN search based on KD [4] or ball trees [17]. Such methods have complexity $O(dN \log N)$, with d as the dimension of the space. Other alternatives include approximate search ([6]). In the second part, the diffusion vector χ is multiplied with the transition matrix. Addressing its sparsity as $O(KN)$ non-zero entries, this operation scales linearly in N as $O(TKN)$. T and K determine the level of confidence imposed by the diffused training set over the unlabeled set. Higher K imposes strong confidence in the current labeling hypothesis, but renders the diffusion more exhaustive. Similarly, large number of iterations T may result in an overly smoothed (and less informative) signal $\chi^{(T)}$. During exploration, large T imposes an hypothesis that may be locally correct but is far from being globally reliable. In our experiments, we use $T \simeq \log_K N$, with the intention to cover most of the graph via diffusion: assuming a diameter-balanced graph [19]. The cover requires $O(\log_K N)$ iterations, if the labelled set is small (i.e. $|X_l| \approx O(1)$, as typical in active learning settings). We use K sufficiently high to allow graph connectivity. This parameter selection leads to a diffusion process that scales as $O(KN \log_K N)$.

Finally, a batch of smallest soft-labels needs to be queried. This requires a quick-sort to be applied to the soft labels magnitude, which scales $O(N \log N)$. We conclude that the running time of DDAL is $O(dN \log N + KN \log_K N + N \log N)$, whereas for a constant K can be further simplified to $O(dN \log N)$. We note that the number of units in the penultimate layer d is typically small.

V. Sample Complexity Analysis

We analyze two fundamental proximity graph structures of the penultimate layer that model the two stages of exploration and refinement in the hidden layer graph representation. We address the binary setting to facilitate the analysis.

The first case addresses the early exploration stage, as the hidden layers representation is not sufficiently adapted to the class function, and therefore separation between classes is not detectable. We model this setting as a K-regular graph or a grid (similarly to the work of [7]). The binary class function divides the grid to two regions in which each class resides. Localization of the decision boundary is still not achieved by the learner at this exploratory stage, as we show that the query complexity of a full recovery of the graph-cut scales linearly in the size of the boundary set and logarithmic in N.

The second case captures a learning stage of a converging network with a larger training set at the mature refinement stage. In this phase, well-separated clusters emerge in the hidden layer representation, which correspond to label assignments.

A. The non-separable grid case.

Definition 1. A **d-dimensional weighted grid graph** is defined as the weighted graph $G(V, E, W)$ on the $\sqrt[d]{N} \times \dots \times \sqrt[d]{N}$ grid of N nodes constructed via the graph Cartesian product on the $\sqrt[d]{N}$ line graphs. The edge weights W_{ij} are fixed.

Definition 2. The **balancedness** β of a labelled set $C = \{(x_i, y_i)\}_{i=1}^N$ is defined as $\beta = \min_j \frac{|C_j|}{N}$.

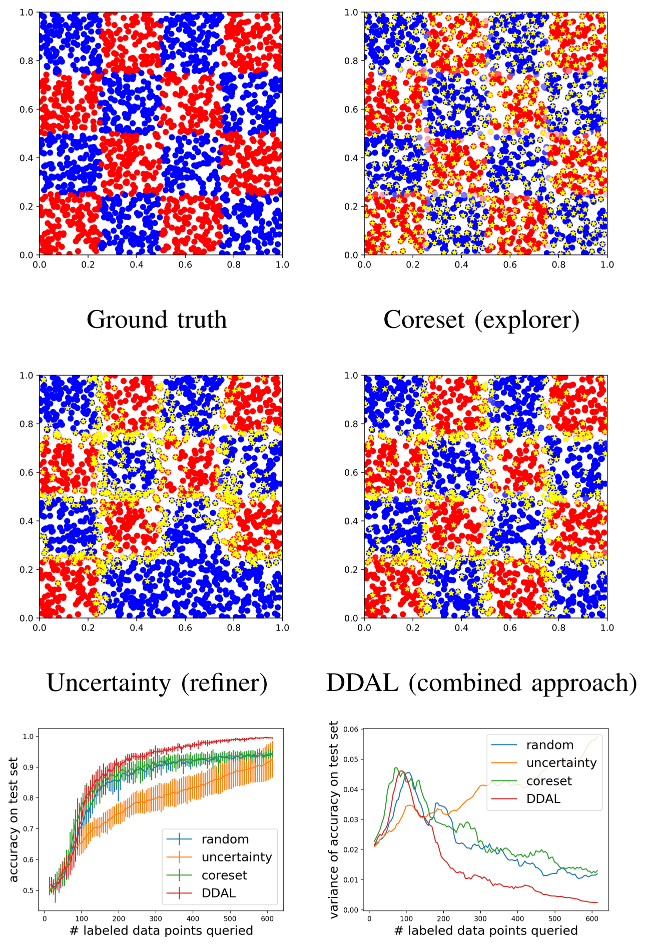


Figure 3: Exploration vs. refinement in the checkerboard example. **Top**: from left to right: *(1)* Binary checkerboard dataset *(2)* Points queried (in yellow) using: *coreset* exploratory criteria [21] *(3) uncertainty* refiner criterion [14] *(4) our* criterion. **Bottom**: (left) Corresponding curves of accuracy versus size of training set, and (right) accuracy variance demonstrates the advantage and robustness of the combined exploration-refinement approach in DDAL.

Theorem 1. *Let the graph $G(V, E, W)$ correspond to a d-dimensional regular grid, and assume that the grid is partitioned by the cut set C of size $|\partial C|$ with balancedness β. Then the sample complexity of BatchQuery for recovering the cut with probability at least 1 - δ for any $\delta > 0$ is*

$$\frac{\log \frac{1}{\beta\delta}}{\log \frac{1}{1-\beta}} + O(d \log_K N + |\partial C|). \tag{8}$$

B. The separable case.

At the advanced refinement stage the hidden layer representation has structure comprising separated clusters that correspond to different label assignments. The K-NN graph, in this case, has a minimal cut that corresponds to the decision boundary between the two classes. We analyze the query complexity of well-separated class samples via a Gaussian Mixture Model (GMM) $w_1\mathcal{N}(m_1, \Sigma_1) + w_2\mathcal{N}(m_2, \Sigma_2)$ modelling two densities in the hidden layer representation. The separation

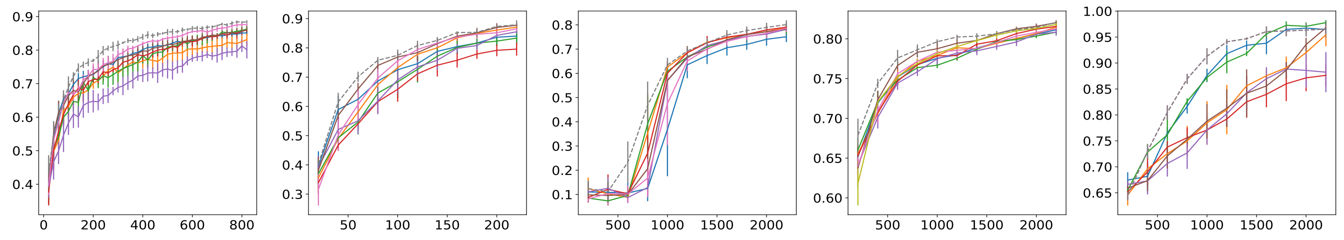


Figure 4: Accuracy vs. queried points. From left to right: MNIST (fully connected), MNIST (convolutional), SVHN, CIFAR10, and Openml155 datasets. The x-axis represent the labels queried, while the y-axis represents test set accuracy. Legend: (–) random, (–) uncertainty, (–) coreset, (–) bayes-entropy, (–) entropy, (–) margin, (–) bayes-uncertainty, (–) badge, (- -) diffusion.

S is defined as the minimal distance between the centroids, relative to the covariance:

$$S = \frac{\|\mu_i - \mu_j\|}{\sqrt{d}\left(\sqrt{\lambda_{\max}(\Sigma_i)} + \sqrt{\lambda_{\max}(\Sigma_j)}\right)}. \tag{9}$$

Theorem 2. *Consider a Gaussian Mixture Model of 2 well-separated ($S >> 1$) Gaussians of balancedness β. Let the graph $G(V, E, W)$ be the KNN graph constructed from the GMM samples with weights according to eq. (1). Then $BatchQuery$ will recover the decision boundary with probability $1-\delta$ by querying $\log\frac{1}{\beta\delta}\left(\log\frac{1}{1-\beta}\right)^{-1}$ points.*

VI. Experiments

Exploration vs. refinement: a toy example We consider the binary checkerboard example (Figure 3) to demonstrate the utility of refinement vs. exploration criteria [3]. The comparison demonstrates how different criteria tend to operate: Refinement-based criteria - *uncertainty* [14], tackle decision boundaries as soon as they individuate them. Lacking proper exploration, it results in a completely mis-classified region. On the other hand, exploratory criterion, such as *Coreset* [21], does not fully detect the decision boundary. Finally, DDAL operates in two phases: first, it explores the distribution to discover all decision boundary and then localizes them. Accuracy results and stability results in Figure 3-bottom demonstrate DDALs performance compared to other criteria.

Benchmark evaluation. In Figure 4 we provide a performance comparison with other active learning methods. We selected fundamental approaches: refinement (e.g. uncertainty [13] and entropy [9]), exploratory (e.g. Coreset [21]), and a combined approach (e.g. 'margin' [20, 15] and Badge [1, 2]). Our selections were guided by i) comparing the exploration-refinement trade-off between different active learning approaches, and ii) comparing deep active learning approaches. We experiment with the following benchmark classification problems: MNIST, CIFAR10, Openml155 [24], and SVHN. Additional experimental details are reported in the appendix. The advantage of DDAL is prominent during the early exploration and transition to refinement.

Robustness study. We examine the robustness of DDAL and state-of-the-art for the choice of batch size and number of epochs in Figure 5. For large batch size the criterion in (6) becomes more exploratory. Namely, it probes points with larger distance between them. This is more advantageous in the early stages of learning. Small batches may focus more on refinement of the decision boundary and may give better gains at later stages. Conversely, when the batches are small they are revisited more often by the optimizer in each of the active learning iterations, and therefore faster convergence is observed.

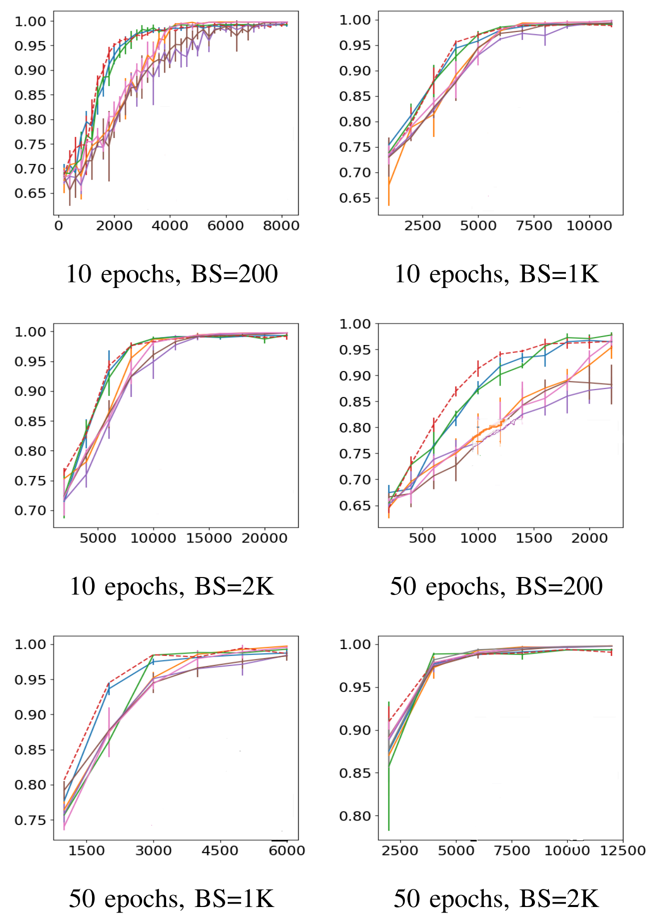


Figure 5: Query size & epochs vs accuracy in Openml 155. Legend: (–) random, (–) uncertainty, (–) coreset, (- -) diffusion, (–) bayes-entropy, (–) entropy, (–) bayes-uncertainty.

References

[1] Jordan T Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. *arXiv preprint arXiv:1906.03671*, 2019.

[2] Jordan T Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Project. https://github.com/JordanAsh/badge, 2019.

[3] El-Yaniv R. Luz K. Baram, Y. Online choice of active learning algorithms. *JMLR*, 5:255–291, 2004.

[4] J. Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[5] F. R. K Chung. *Spectral Graph Theory*. CBMS Regional Conference Series in Mathematics, 92 edition, 1996.

[6] Immorlica N. Indyk P. Mirrokni V. S Datar, M. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the 20th annual symposium on Computational geometry*, 2004.

[7] X. Zhu G. Dasarathy, R. Nowak. S2: An efficient graph based active learning algorithm with application to nonparametric classification. In *Proceedings of The 28th Conference on Learning Theory*, volume 40 of *Proceedings of Machine Learning Research*, pages 503–522. PMLR, 03–06 Jul 2015.

[8] Islam Riashat Ghahramani Zoubin Gal, Yarin. Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1183–1192. JMLR. org, 2017.

[9] Alex Holub, Pietro Perona, and Michael C Burl. Entropy-based active learning for object recognition. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8. IEEE, 2008.

[10] Szegedy C. Ioffe, S. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[11] A. Krishnamurthy J. Langford A. Agarwal J. T. Ash, Chicheng Z. Deep batch active learning by diverse, uncertain gradient lower bounds. In *International Conference on Learning Representations*, 2020.

[12] Ba J. Kingma, D. P. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[13] David D Lewis. A sequential algorithm for training text classifiers: Corrigendum and additional data. In *Acm Sigir Forum*, volume 29, pages 13–19. ACM New York, NY, USA, 1995.

[14] Gale W. A. Lewis, D. D. A sequential algorithm for training text classifiers. In *ACM SIGIR*, pages 3–12, 1994.

[15] Tong Luo, Kurt Kramer, Dmitry B Goldgof, Lawrence O Hall, Scott Samson, Andrew Remsen, Thomas Hopkins, and David Cohn. Active learning to recognize multiple types of plankton. *Journal of Machine Learning Research*, 6(4), 2005.

[16] H. H. Zhou M. Löffler, A. Y. Zhang. Optimality of spectral clustering for gmm, 2019.

[17] S. M Omohundro. *Five balltree construction algorithms*. International Computer Science Institute Berkeley, 1989.

[18] Gross S. Chintala S. Chanan G. Yang E. DeVito Z. Lin Z. Desmaison A. Antiga L. Lerer A. Paszke, A. Automatic differentiation in PyTorch. In *NIPS Workshop*, 2017.

[19] P. Sparl S. Miklavic. -distance-balanced graphs. *Discrete Applied Mathematics*, 244:143 – 154, 2018.

[20] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden markov models for information extraction. In *International Symposium on Intelligent Data Analysis*, pages 309–318. Springer, 2001.

[21] Savarese S. Sener, O. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

[22] B. Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012.

[23] Hazırbas C. Triebel R. Cremers D. Stark, F. Captcha recognition with active deep learning. In *Workshop new challenges in neural computation*, page 94, 2015.

[24] Jan van Rijn. Normalized version of the pokerhand data set.

[25] Ma Z. Nie F. Chang X. Hauptmann A. G. Yang, Y. Multi-class active learning by uncertainty sampling with diversity maximization. *Int. J. Comput. Vision*, 113(2):113–127, June 2015.

APPENDIX

TRAINING DETAILS

We report the architecture and hyper-parameters used in the experiments in Section VI for each of the data sets. We used for most of the data intensive experiments Ubuntu 16.04 on a machine with 2x Intel Xeon Gold 6126 CPU @2.6GHz/3.7GHz 12C/24T each, 2x512GB SSD for system (RAID0 = 512GB usable), and 4xNvidia GTX 1080Ti 11GB GPUs.

a) Active learning training: After each batch of queries the network is trained for a certain number of epochs, starting from the previous configuration. The reported accuracies are averaged over 5 runs. All the experiments were performed using PyTorch [18].

b) Checkerboard: We consider a pool data set of $|X_{\text{pool}}| = 2000$ labeled points drawn from the checkerboard distribution. We start with a training set X_ℓ^0 composed of 4 points randomly drawn from the pool set for each of the classes (so that $\left|X_\ell^0\right| = 8$) and train a feed-forward neural network. The accuracy and its variance are measured on a separate test data set of $N_{\text{test}} = 200$ points drawn from the checkerboard distribution in Fig. 2. All the results are averaged over 5 runs.

We trained a fully-connected network with 2 hidden layers of width 30 each. We optimized using SGD with batch size 1, learning rate 0.001 and momentum 0.9. We ran 100 epochs after each query. The experiments were run on a pool set of size 2000. The accuracy was evaluated on a separate test data set sampled from the same distribution (200 points).

c) MNIST: The fully-connected model we used had 2 hidden layers of width 100 and 50 respectively. The convolutional model used was composed by a convolutional layer with 16 channels and a kernel of size 5×5, a MaxPool layer with a kernel of size 2 and padding 2, and 2 hidden fully-connected layers of width 20 each. For both models, we optimized using Adam [12] with batch size 8 and learning rate 0.001. For both models, a BatchNorm [10] layer was added before each hidden fully-connected layer. We ran 100 epochs after each query. The experiments were run on a pool set formed by a (balanced) randomly selected subset of the training data set of size 10000. The accuracy was evaluated on the test data set (10000 points).

d) CIFAR10 : The network used was a VGG-16 architecture pre-trained on ImageNet. We took the convolutional part of such network and added 2 fully-connected layers of width 512 and 20 respectively. A dropout layer was added after each of these fully-connected layers. Only the fully-connected layers were trained. We optimized using Adam [12] with batch size 100. After each query the learning rate was initialized to 0.0003 and decayed by 0.5 every 30 epochs. We ran 100 epochs after each query. The full training data set (50000 points) was used as pool data set for the experiments. The accuracy was evaluated on the test data set (10000 points).

e) SVHN : The network used was a VGG-16 architecture. We optimized using SGD with batch size 50, learning rate 0.005, momentum 0.9 and weight-decay 0.0005. We ran 50 epochs after each query. The experiments were run on a pool set formed by a (balanced) randomly selected subset of the training data set of size 20000. The accuracy was evaluated on the test data set (26032 points).

f) Openml155: For the OpenML, we use a two-layer Perceptron with ReLU activations (MLP) as in [11]. The embedding dimensionality of the MLP is 1024, as more capacity helps the model fit training data. We fit models using cross-entropy loss and the Adam variant of SGD. We use a learning rate of 0.0001.

g) DDAL: The following parameters were used for the diffusion algorithm in the experiments presented in Section VI. For the experiment with the checkerboard data set (experiment in Figure 2), we used $T = 4$, $K = 10$ and $P = 1$. For the experiments with the MNIST data set (experiment in Figure 4), we used $T = 5$, $K = 10$ and $P = 1$. For the experiment with the CIFAR10 data set (experiment in Figure 4), we used $T = 4$, $K = 20$ and $P = 10$.

h) Bayesian criterion: In order to perform the active learning queries using the Bayesian criterion, we added a dropout layer after each hidden fully connected layer to the models with no dropout layers.

A. Variants of multi-class extension

In Section IV, we described a possible formulation to extend the diffusion-based active learning criterion to the multi-class setting. We propose in the following other possible formulations.

a) One-vs-all approach: In the one-vs-all setting described in Section IV, the batch is queried according to

$$\hat{X} = \arg\min_{i \in X_u}^B q\left(\chi_{:,i}^{(T)}\right)$$

for some chosen function q which measures a notion of uncertainty at point x_i, given the matrix $\chi^{(T)}$. In Section IV, we chose q to be

$$q\left(\chi_{:,i}^{(T)}\right) = \min_{c \in [C]} \left|\chi_{c,i}^{(T)}\right|, \ or \ q\left(\chi_{:,i}^{(T)}\right) = \left\|\chi_{:,i}^{(T)}\right\|_p$$

for some $p \in [1, \infty]$.

b) Multivariate diffusion approach: Moving from the one-vs-all approach, we can performs the query as follows, using the property that M is a stochastic matrix. For each data point x_i, we propagate a probability vector $\chi_i^{(t)} \in \Delta_C$. This vector can be initialized as

$$\chi_{i,c}^{(0)} = \begin{cases} 1 & \text{if } i \in X_\ell \text{ and } c = y_i \\ 0 & \text{if } i \in X_\ell \text{ and } c \neq y_i \\ \frac{1}{C} & \text{otherwise} \end{cases}$$

We can therefore diffuse the matrix aggregating the signal for all the points, $\chi^{(t)} \in \mathbb{R}^{N \times C}$, and diffuse it as in the binary case:

$$\chi^{(t)} = M\chi^{(t-1)}, \qquad \chi^{(t)}|_{X_\ell} = \chi^{(0)}|_{X_\ell}$$

Since M is stochastic, it holds that $\chi^{(t)} \in \Delta_C$ at each iteration t. Therefore we can interpret each vector $\chi_i^{(t)}$ as a probability vector of the data point x_i belonging to different classes, obtained by the diffusion above. It therefore makes sense to choose the points to query as

$$\hat{X} = \arg\min_{k \in X_u}^{B} q\left(\chi_i^{(T)}\right)$$

where $q : \Delta_C \to \mathbb{R}$ is some measure of uncertainty. Possible choices include:

- Uncertainty: $q(p) = p_{c^*}$, where $c^* = \arg\max_c p_c$;
- Margin: $q(p) = p_{c^*} - p_{c_2^*}$, where c^* is defined as above and $c_2^* = \arg\max_{c \in [C] \setminus \{c^*\}} p_c$;
- Negative entropy: $q(p) = \sum_{c \in [C]} p_c \log p_c$.

B. PROOF OF LEMMA 1

Proposition 1. *A general solution to the Jacobi iteration (5) after t iterations and with initial conditions specified by $\chi^{(0)}$ is given by*

$$\chi^{(t)} = (M)^t \chi^{(0)} = c_1 \lambda_1^t \phi_1 + ... + c_n \lambda_n^t \phi_n, \qquad (10)$$

where $\chi_{i:x_i \in X_l}^{(t)} = y_i$, for each $t \in \{0, ..., T\}$, and $c_1, ..., c_n$ are coefficients that are prescribed by the initial condition $\chi^{(0)} = c_1\phi_1 + ... + c_n\phi_n$, where $\lambda_1, ..., \lambda_n$ and $\phi_1, ..., \phi_n$ are the eigenvalues and eigenvectors of M.

Via a simple algebraic manipulation those eigenvectors are also shown to be the eigenvectors of the graph's Laplacian:

Proposition 2. *$\lambda_1, ..., \lambda_n$, $\phi_1, ..., \phi_n$ are also solutions to the system:*

$$L\phi = (1 - \lambda)D\phi, \qquad (11)$$

where $L = D - W$.

The eigenvector ϕ_2 is a solution to the relaxed Minimal Normalized Cut problem (*MinNCut*) [5] in a graph $G(V, E)$ constructed from data representation in the hidden layer f_θ^n:

$$MinNCut = \arg\min_{h: h^T D \mathbf{1} = 0} \frac{h^T L h}{h^T D h}. \qquad (12)$$

We follow on the proposition 1 eq. (10). We first note that $\chi^{(0)}$ has the sign of its non-zero coordinates as the subset of the coordinates of ϕ_2 (up to a sign permutation), and after each iteration they are restarted. Since ϕ_1 has constant sign (and magnitude) restarting with opposite signs causes c_1 to be suppressed to zero already at $t = 0$. Since $\phi_2 \perp \phi_i$ for $i = 3, ..., N$, there have to be coordinates $j_1, ..., j_k;\ j_k \in \{1, ..., N-1\}$, for which $sign(\phi_i(x_{j_m})) \neq sign(\phi_2(x_{j_m}))\ 1 \leq m \leq k$. We consider two cases:

1) $\chi^{(0)}$ contains such non-zero coordinates. For all eigenvectors $\phi_{i_1}, ..., \phi_{i_l}$ with such sign inequality we have that $c_2 > c_{i_l}$ for $i_l \in \{3, ..., N\}$, and since $\lambda_2 \geq \lambda_{i_l}$ we have that ϕ_2 is dominant. The restarting of χ prevents it from converging to 0 in case $c_2\lambda_2 < 1$, and we conclude our proof. For all other eigen-components (for which $\chi^{(0)}$ has 0 coordinates where $sign(\phi_i(x_{j_m})) \neq sign(\phi_2(x_{j_m}))\ 1 \leq m \leq k$) the following case also holds.
2) All coordinates $j_1, ..., j_k$ in $\chi^{(0)}$ are zero: Since for a well-separated GMM the kernel K is a 2-block stochastic matrix, point x_{j_m} will be transduced only from points x_l for which $y_{j_m} = y_l$. For every such point x_{j_m} there exist an iteration t' such that $\chi^{(t')}(x_{j_m})$ becomes non-zero and attains the sign of y_l. At this t' all eigen-components $\phi_i,\ i = 3, ..., N$ in the iterant $\chi^{(t')}$, which have coordinates such that $sign(\phi_i(x_{j_m})) \neq sign(\chi^{(t')}(x_{j_m}))\ 1 \leq m \leq k$, will be reduced. Once all such points are visited the dominant component will be ϕ_2, and the remaining components will converge to zero.

C. PROOF OF PROPOSITION 1

The eigenvectors of the kernel M form a complete basis and therefore any vector in $\mathbb{C}^N$ can be represented as their linear combination. What is left to prove is that any initial conditions can be matched. To this end we need to show that

$$Sc = \chi^{(0)} \qquad (13)$$

can be solved, where S is the matrix whose columns are the eigenvectors of B_J (the iteration matrix introduced in Section III for the iteration in eq. (5)). Since the eigenvectors are linearly independent S is non-singular and (10) always admits a solution.

D. PROOF OF PROPOSITION 2

Simple algebraic operations yield the result:

$$D^{-1}W\phi = \lambda I\phi \Leftrightarrow \phi K\phi = \lambda I\phi \Leftrightarrow W\phi = \lambda D\phi \Leftrightarrow$$
$$(D - L)\phi = \lambda D\phi \Leftrightarrow L\phi = D\phi - \lambda D\phi \Leftrightarrow L\phi = (1 - \lambda)D\phi$$

E. PROOF OF THEOREM 1

We note that eq. (8) is derived from several steps and after reducing constant factors. We elaborate these steps below:

1) The first term of (8) depends on the balancedness β, and accounts for the number of queries required to achieve a representative for every class. The ratio in (8) is a direct result of Lemma 4 in [7].
2) Since at every diffusion step K nodes are transduced with a label, K^T nodes will have a soft label after diffusing from a labelled node (assuming a connected graph). On the other hand, the approximate number of labelled nodes needed to cover the graph with soft labels can be derived in expectation from the ratio $\frac{N}{K^T}$ for connected graphs. When $T = O(\log_K N)$ all $O(N)$ unlabelled nodes are reachable in expectation in T iterations, and $O(1)$ labelled nodes are needed. Exploration involves $O(\frac{N}{K^T})$ queries to guarantee all nodes have non-zero labels via diffusion.
3) The search process is dictated by the minimal value of the diffusion soft label (criterion (6)) obtained after each propagation of the labels from a pair of nodes v_i v_j of opposite labels, to their neighbors along the shortest path between them. This propagation repeats until two nodes of opposite sign are discovered. Since the absolute value among those two nodes is expected to be minimal, our query criterion selects one of those nodes to be queried, and the process repeats until his opposite label neighbor is queried as well. For a regular graph grid in which all the transition probabilities are equal, the queried node will reside at the midpoint between v_i and v_j, thus resulting in $O(\log M)$ queries, where M is the length of the shortest path between them.

Since after exploration with $T = O(\log_K N)$ the maximal shortest distance between any two nodes v_i, v_j, of opposite label is at most $2K^T$ hops, we have that at most $\log 2K^T = 1 + T \log K = 1 + \log_K N \cdot \log K$ queries are required until the first two connected cut nodes are recovered. Since in a connected grid each node has $K = 2^d$ neighbors. We obtain at most $1 + d \log_K N$ queries are sufficient.

4) The next propagation will reach another cut node in a constant number of propagations, specifically for any grid in just 2 hops. Since this is a constant the following queries scale as the size of the cut - $|\partial C|$.

F. PROOF OF THEOREM 2

We start with auxiliary results (proved below) that show that the diffusion iteration converges to a minimal cut solution in the graph $G(V, E, W)$ derived from the top hidden layer, under the separation requirement. As a result, the decision boundary is recovered accurately because the GMM cluster assignments correspond to the ground truth labels.

We start with observing the general form of the solution of the iteration (5) as in Proposition 1:

$$\chi^{(t+1)} = M^{t+1}\chi^{(0)} = c_1\lambda_1^t\phi_1 + ... + c_n\lambda_n^t\phi_n, \tag{14}$$

where $\chi^{(t)}_{l:x_i \in X_l} = y_i$, for each $t \in \{0, ..., T\}$, and $c_1, ..., c_n$ are coefficients that are prescribed by the initial condition $\chi^{(0)}$: $\chi^{(0)} = c_1\phi_1 + ... + c_n\phi_n$, and $\lambda_1, ..., \lambda_n$ and $\phi_1, ..., \phi_n$ are the eigenvalues and eigenvectors of M.

Let $\lambda_1, ..., \lambda_n$ and $\phi_1, ..., \phi_n$ be the set of eigenvalues and eigenvectors of M ordered in the decreasing order of the eigenvalues. Using Proposition 2 and Lemma 1 we show the equivalence of the iteration solution as $t \to \infty$ to the Minimal Normalized Cut solution (*MinNCut*) [5].

As achieved by Lemma 1, the construction of the decision boundary can be done by following the convergence to the minimal cut solution. We therefore show that the sample complexity is dominated by the initial class sampling and the sampling in the exploration stage, which allows the labels to propagate through the whole graph for a given T.

1) At the initial stage data is queried until at least one sample of each label exists. Assuming that β is the measure of balancedness (as definition 2), the first component to consider is the complexity required to discover all classes, following on Lemma 4 in [7].
2) Next, we are looking to sample a set of labelled nodes such that when diffused T times all unlabelled nodes will have a non-zero soft label. As discussed above in the proof of Theorem 1, the expected number of queries is $O(\frac{N}{K^T})$ to guarantee coverage by diffusion for almost all nodes in T diffusion steps. On the other hand, setting $T = \log_K N$, renders this complexity as constant $O(1)$, independent of N, which can be neglected.
3) Once all such points are queried the minimal cut corresponding to the Gaussian clusters is obtained accurately according to Lemma 1, for sufficiently large T via the diffusion process. The optimality of the spectral cut solution here is supported by Theorem 1.1 in [16], which we give below for completion:

Theorem 3. *For N data points generated from a Gaussian Mixture model, if*

- *number of clusters is finite*
- *the sizes of clusters are in the same order*
- *the minimum distance among centers goes to infinity*
- *the dimension d is at most in the same order of N*

then with high probability, spectral clustering achieves the optimal clustering rate, which is

$$l(\hat{\chi}, \chi^*) = N \exp\Big(-(1 - o(1))\frac{\triangle^2}{8}\Big) \tag{15}$$

where $l(\cdot, \cdot)$ is the Humming loss function, and $\triangle$ is the distance between the centroids.

The theorem provides the final step, showing that ϕ_2 is an optimal solution for finding the decision boundary between the Guassians in the GMM.

We note that the assumptions of Theorem 3, in particular, the high separation between the class-clusters (here modeled by the GMM) are attained due to the highly trained network at the refinement stage. In this stage the penultimate layer is well trained and present a structured graph where members of different class are separated.