Active Metric Learning and Classification using Similarity Queries (Supplementary Material)

Namrata Nadagouda **Austin Xu** **Mark A. Davenport**

School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia, USA

A PLACKETT-LUCE MODEL DETAILS

The Plackett-Luce model is derived from an assumption, the Luce's choice axiom [Luce, 1959], also known as the Independence of Irrelevant Alternatives (IIA), which states that the presence of other items in a choice set do not change the relative probabilities of choosing items in the set. This is a reasonable assumption in our setting. This model belongs to a family of discrete choice models which are commonly used to describe situations where a selection is made from a set of options. Such scenarios are encountered widely in the fields of economics [Train, 2009], for example, to explain the choice made by a company on whether or not to launch a product into the market, in psychology [Tversky and Kahneman, 1981] to interpret the choices made by humans in every day situations and, more recently, in computer science [Rosenfeld et al., 2019] to model choices made by a user in online platforms.

B METRIC LEARNING

In this section, we provide precise experimental details and highlight additional metric learning experimental results for both DML and non-parametric embedding learning via MDS.

B.1 DEEP METRIC LEARNING

Neural network architectures and learning rates. For the DML experiments, we utilize the following network architectures and learning rates for the three datasets. We utilize networks consisting only of fully connected layers with ReLU nonlinearities inserted between all layers.

- **Mahalanobis Metric Dataset:** Fully connected layers of sizes 32, 48, and 10, respectively. Learning rate: 0.0001
- **Food73 Dataset:** Fully connected layers of sizes 12, 12, and 12, respectively. Learning rate: 0.0005
- **Graduate Admissions Dataset:** Fully connected layers of sizes 16, 12, and 10, respectively. Learning rate: 0.0001

We utilize the same learning rate for re-training models across all methods (random, Info-NN, Batch-Euclidean/Centroid).

Experiment parameters. In all experiments, we utilized a value of $\mu = 0.00001$ for the probability model and utilized 20 initialization triplets. Batch sizes of 10 (synthetic), 30 (food and graduate admissions) are used. Furthermore, for Info-NN experiments, we utilize the following values for hyperparameters σ^2 (distance distribution variance), n_s (number of samples used to compute mutual information), B and B' (number of top most informative queries selected per batch):

- **Synthetic Mahalanobis Metric Dataset:** $\sigma^2 = 1$, $n_s = 100$, $B' = 10 = B$
- **Food73 Dataset:** $\sigma^2 = 6.5$, $n_s = 1,000$, $B' = 5$
- **Graduate Admissions Datset:** $\sigma^2 = 10$, $n_s = 1,330$ $(= 10N)$, $B' = 5$

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

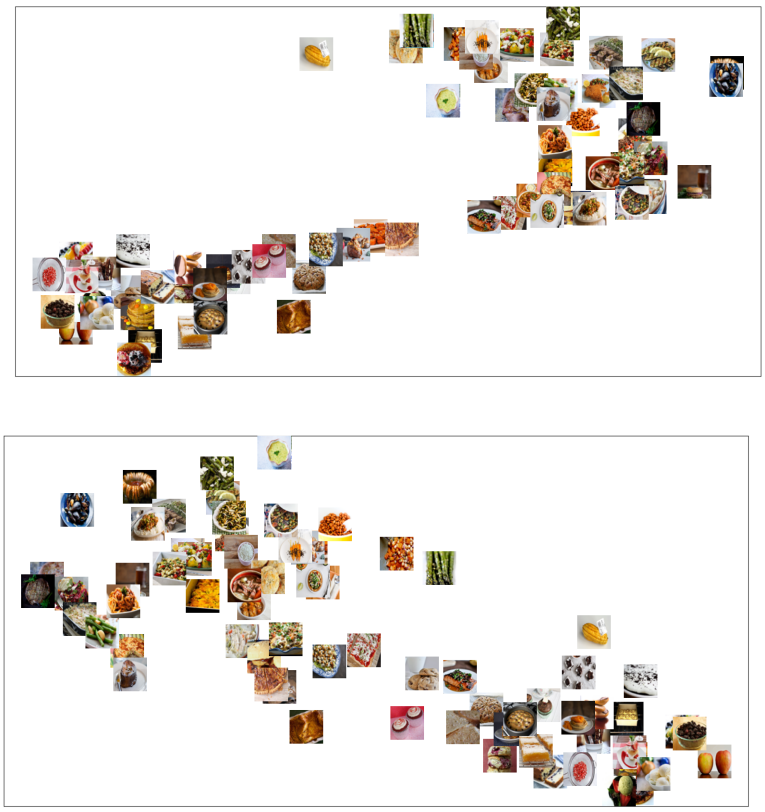

Figure 1: Visualization of food embedding learned using queries selected with Batch-Centroid (top) and Batch-Euclidean (bottom) generated using t-SNE [Maaten and Hinton, 2008].

As reported in the main paper, we used batch sizes of $10, 30$, and 30 for the Mahalanobis, food, and admissions datasets respectively. These batch sizes are the sizes of the NN queries collected. For any method using triplets, the batch size is doubled, resulting in batch sizes of $20, 60$, and 60, respectively. This is done so we can compare both on a per-query and per-triplet basis. To set such parameters, a coarse grid search was performed to find the best performing parameters.

We compared our method against two baselines found in Kumari et al. [2020]. These baselines follow the same general approach of weighting informativeness (measured using entropy) and diversity (measured using various metrics such as the Euclidean distance of all permutations of the triplet or the centroid of the three points selected in the triplet) for an *overcomplete* batch size. We utilize an overcompleteness factor of 3, which indicates that for a batch of B triplets, the $3B$ most informative triplets are identified. The informativeness of the $3B$ triplets are then weighted by the informativeness, and the top B triplets are then presented to the oracle. From studies performed by Kumari et al. [2020], anything above a factor of 2 exhibits roughly the same performance.

Additional embedding visualizations. Models used to generate all embedding visualizations, including those shown in the main paper, used the same number of triplets. We present an additional visualization of the Food73 dataset embedding learned with the Batch-Centroid and Batch-Euclidean methods in Fig. 1. In comparison to the embedding learned with Info-NN (Fig. 3 in main paper), the embedding learned with Batch-Centroid after the same number of triplets does a poorer job of grouping together vegetables, unlike the Info-NN embedding.

We also present a visualization of the embedding learned via Batch-Euclidean on the Graduate Admissions dataset in Fig. 2. Comparing embeddings learned with Info-NN and Batch-Centroid (Fig. 5 in main paper) and Batch-Euclidean, it is clear that Info-NN selects queries that more closely group highly ranked candidates together. However, none of the methods visualized are able to completely cluster candidate tiers distinctly; for all three methods, admitted students (fellowship and non-fellowship) are intermingled with candidates in the first and second rejection tiers.

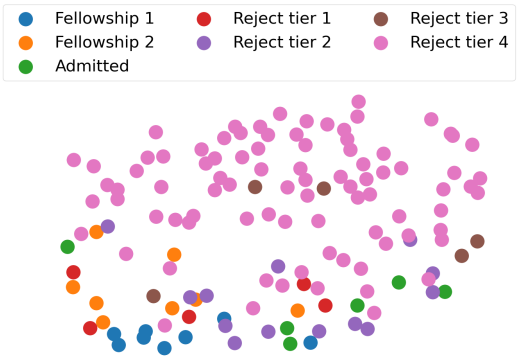

Figure 2: Visualization of admissions embedding learned using queries selected with Batch-Euclidean generated using t-SNE [Maaten and Hinton, 2008].

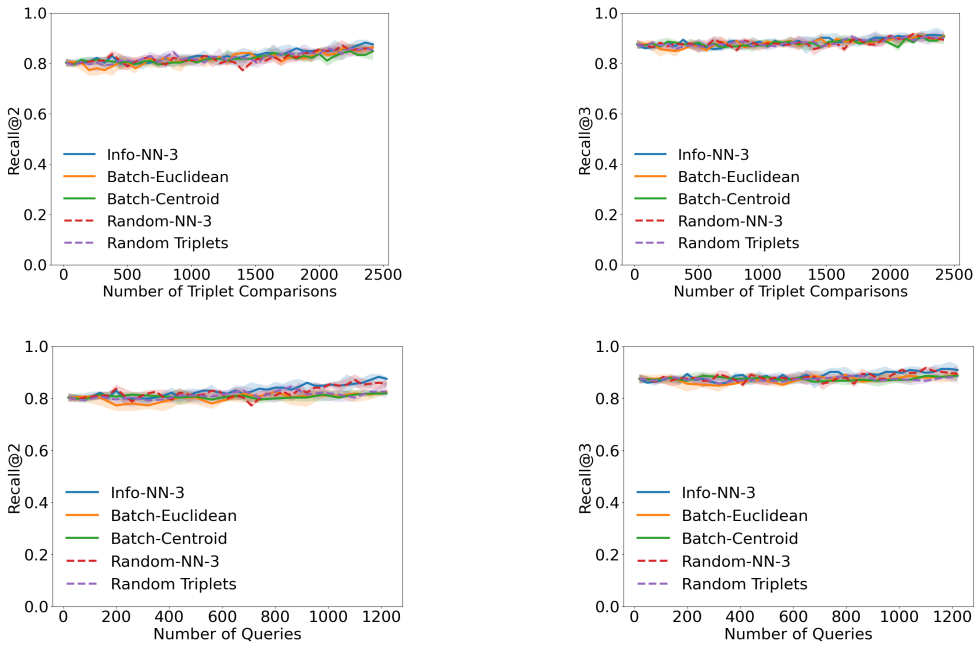

Figure 3: Per-triplet (top) and per-query (bottom) comparison for Info-NN against other methods. Recall@2 (left) and Recall@3 (right).

Additional results on Graduate Admissions dataset. Results for additional values of K for Recall@K and TopFraction@K are presented in Fig. 3 and Fig. 4, respectively. On both a per-triplet and per-query basis, Info-NN is performs the best for all values of K. We note that for Recall@K for larger values of K, all methods perform roughly the same and perform well. This is because the dataset contains a large number of tier 4 rejections, which every method is able to successfully group together, inflating the Recall@K value. Thus, we believe that the TopFraction@K results do a better job of illustrating how the method does in selecting queries that group admitted or more highly ranked candidates together.

B.2 MDS EMBEDDING LEARNING

We perform a set of experiments which utilize MDS to learn representations of the items. In particular, we use this opportunity to compare the performance of NN queries against a more complex ranking query [Canal et al., 2020]. When comparing against ranking queries, it is important to note that **we expect both actively selected and randomly selected ranking queries to outperform a nearest neighbor query of the same size on a per-query basis**, as there is a discrepancy in the amount of information each query contains. All experiments were performed on a 2019 MacBook Pro, 2.6 GHz 6-Core Intel i7, 16 GB RAM.

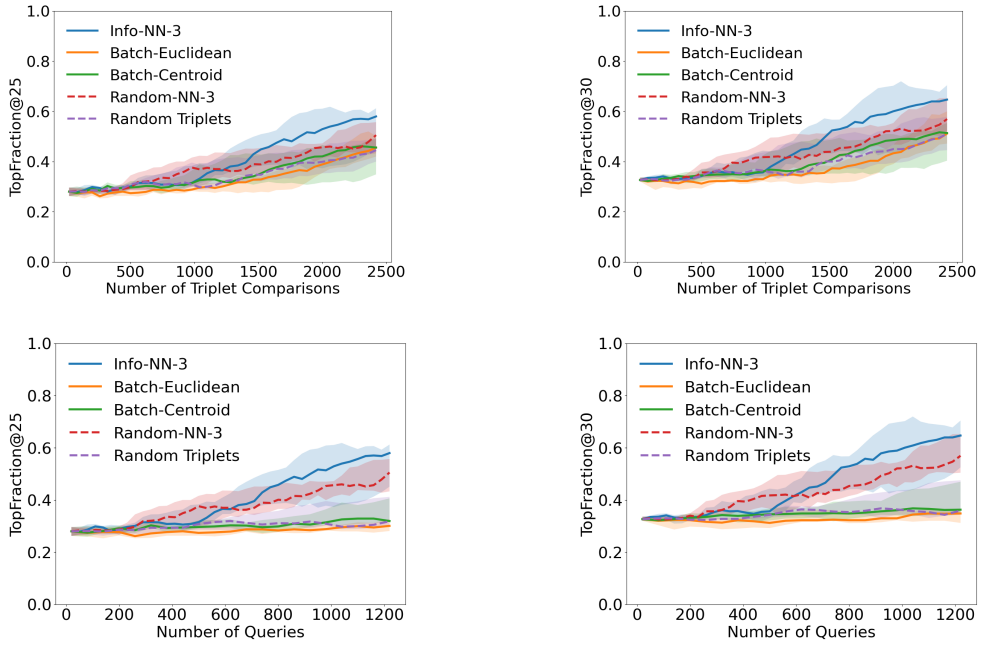

Figure 4: Per-triplet (top) and per-query (bottom) comparison for Info-NN against other methods. TopFraction@25 (left) and TopFraction@30 (right).

Data generation. In each simulation, the ground truth embedding consists of points drawn independently from a multivariate Normal distribution with mean $\mathbf{0}$ and covariance matrix $\boldsymbol{I}$. We utilize a deterministic oracle, which orders the items based on their true distances from the selected reference object and generate a new initialization embedding with entries drawn uniformly at random from $[0, 1]$ for every trial.

Experiment parameters. For both the Info-NN vs. Random-NN and Info-NN vs. Ranking experiments, we utilize a diminishing μ parameter. For each active learning iteration $k \in \{1, \ldots, K\}$, we set $\mu = D_{\max}(0.99)^k$, where $D_{\max}$ is the maximum pairwise distance in the current estimate of the embedding. As presented in Tamuz et al. [2011], the μ parameter can be thought of as a margin. With a diminishing μ, we are enforcing a stricter margin in the earlier stages of learning, when our estimate of the embedding is poor. As the number active learning cycles increases, our estimate of the embedding should improve, thus lessening the need for a larger margin. Multiple other options for μ were considered, such as setting μ to a constant or the maximum of all pairwise distances, but we found that the diminishing μ worked well for the MDS synthetic embedding learning experiments. We utilized step size of $\alpha = 0.5$ for probabilistic MDS. This parameter was not finely tuned. We observed similar performance as long as α is reasonably small ($\alpha < 1$).

Probabilistic multidimensional scaling. To fit an embedding using nearest neighbor or ranking queries, we first decompose the query response into a set of paired comparisons and store these paired comparisons in $\mathcal{S}$. A nearest neighbor query of size M as $M - 1$ paired comparisons and similarly, ranking query of size M can be decomposed into $\frac{M(M-1)}{2}$ paired comparisons. Thus, the active embedding technique framework is general enough to accommodate both query types. We then utilize a version of the probabilistic multidimensional scaling (MDS) approach presented in Tamuz et al. [2011]. Starting with some input embedding $\boldsymbol{Z}$, we perform a fixed number of gradient descent iterations with a fixed step size α (not necessarily to convergence) on the empirical log-loss

$$\ell_{\mathcal{S}}(\boldsymbol{Z}) = \frac{1}{|\mathcal{S}|} \sum_{i=1}^{|\mathcal{S}|} \log \frac{1}{p_{Q_i}},$$

where for $Q_i = r_i \cup \{t_i^{(1)}, t_i^{(2)}\} \in \mathcal{S}$

$$p_{Q_i}(Y_i = 1 \mid D_{Q_i}) = \frac{(D_{i1}^2 + \mu)^{-1}}{(D_{i1}^2 + \mu)^{-1} + (D_{i2}^2 + \mu)^{-1}}.$$

with $D_{Q_i} \coloneqq \{D_{i1}, D_{i2}\}$, $D_{i1} \coloneqq D_{r_i, t_i^1}$, and $D_{i_2} \coloneqq D_{r_i, t_i^2}$. That is, we perform updates of the form $\boldsymbol{Z} = \boldsymbol{Z} - \alpha \nabla \ell_{\mathcal{S}}(\boldsymbol{Z})$.

Algorithm 1 Info-NN-M: Active Embedding Technique

Require: Embedding $\boldsymbol{Z}_{\text{init}} \in \mathbb{R}^{D \times N}$, query length M, number of active learning cycles K, burn-in period K_0, number of samples n_s, number of MDS iterations K_{MDS}, MDS step size α

```
S ← {}
for k = 1, ..., K_0 do
    Q_k ← query of size M drawn uniformly at random
    y_k ← oracle response to Q_k
    S ← S ∪ (y_k, Q_k)
end for
Z_0 ← probabilisticMDS(Z_init, S, K_MDS, α)
for k = 1, ..., K do
    (I, Q) ← {} (Store highest MI value and corresponding query for all references)
    for j = 1, ..., N do
        Q_j ← Set of all queries of size M for which to compute MI with j as reference item
        I_j ← Info-NN-distances(Z, Q_j, n_s) (Compute MI for each query)
        (I, Q) ← (I, Q) ∪ (max I_j, arg max I_j) (Store query in Q_j with highest MI)
    end for
    Q_{j*} ← Query in (I, Q) with highest corresponding value in I
    y_{j*} ← Oracle response to Q_{j*}
    S ← S ∪ (y_{j*}, Q_{j*})
    Z_k ← probabilisticMDS(Z_{k-1}, S, K_MDS, α)
end for
```

Our active embedding strategy, utilizing probabilistic MDS, is as follows: Starting with an initial embedding $\boldsymbol{Z}_0$, we initialize our algorithm by running probabilistic MDS on $\boldsymbol{Z}_0$ with K_0 randomly drawn queries to obtain $\boldsymbol{Z}_1$. At each iteration $k > 0$, we alternate between the following:

1. Fix each column in $\boldsymbol{Z}_k$ as the reference data point, run Info-NN to find the query that maximizes mutual information with respect to the reference, and choose the query with the maximum mutual information over all N reference data points.

2. Solicit a response from the oracle for the chosen query, append the paired comparison decomposition to $\mathcal{S}$, and apply probabilistic MDS to $\boldsymbol{Z}_k$ with the updated $\mathcal{S}$ to obtain $\boldsymbol{Z}_{k+1}$.

The full procedure can be found in Alg. 1

Evaluation metrics. To quantify the performance of our approach, we examine how well our recovered embedding preserves the rank ordering of the items. To do so, we use the Kendall's Tau rank correlation coefficient [Kendall, 1938]. To capture the holistic quality of the learned embedding, we set each object as the reference object, rank all other items based on distance to the reference object, and compute the Kendall's Tau between that item and the ranking induced by the ground-truth embedding with the same reference object. We then define the *aggregate Kendall's Tau* as the mean of all of these Kendall's Tau coefficients. In our simulations we consider multiple trials and we report the median aggregate Kendall's Tau and the 25% and 75% quantiles.

For the following experiments, *Info-Ranking-M* means the active selection method in Canal et al. [2020] was used to select ranking queries each with a set T_i of size M.

Info-NN vs. Random-NN. In the first simulation, we quantify the improvement in using the adaptive algorithm over randomly selected nearest neighbor queries. In particular, we fix $N = 20$, $D = 2$ or $D = 5$, use $K_0 = 20$ initial random queries, and examine the performance for queries of sizes $M = 2, 3, 4$, and 5.

As shown in Fig. 5, for all query sizes the learned embedding is significantly better when queries are selected actively rather than at random. Notably, Info-NN-3 queries exceed the performance of randomly selected size 4 and 5 queries despite being smaller. Randomly selected nearest neighbor queries of sizes $3, 4$, and 5 all performed similarly, indicating that randomly selected queries contain redundant information that cannot be overcome solely by increasing the query size.

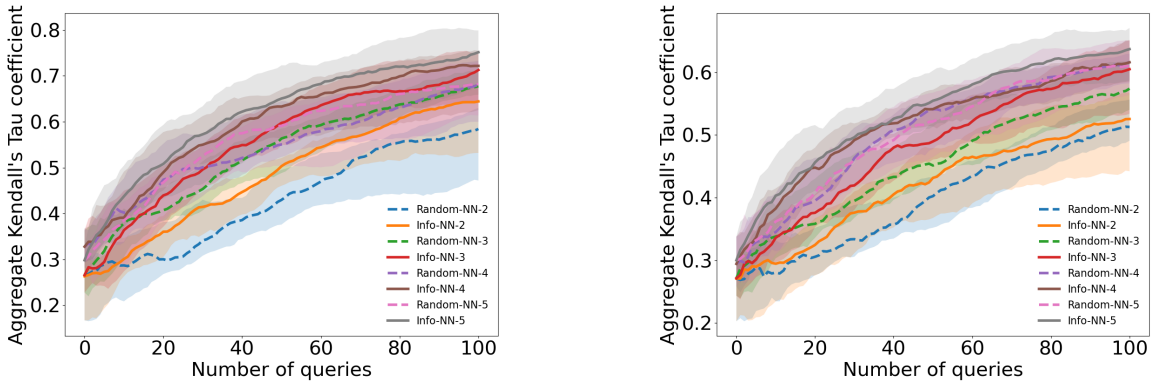

Figure 5: Comparison of actively selected nearest neighbor queries and randomly selected nearest neighbor queries for $D = 2$ (left) and $D = 5$ (right). Info-NN outperforms randomly selected queries in all cases, even outperforming randomly selected queries of larger size in some cases. Gradient step parameters: 500 iterations, step size = 0.5.

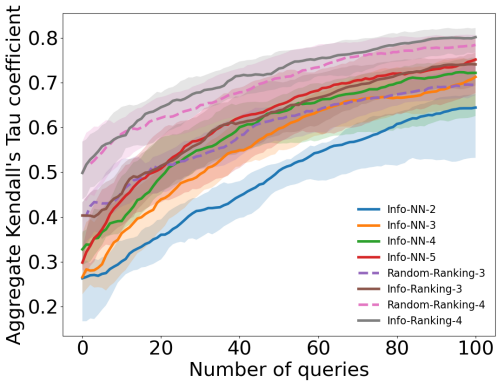

Figure 6: Comparison of actively selected nearest neighbor queries and actively selected and randomly selected ranking queries. Info-NN performs is competitive with a randomly selected ranking query of the same size. Gradient step parameters: 500 iterations, step size = 0.5.

Info-NN vs. Ranking. In the second simulation, we compare the performance of actively selected nearest neighbor queries against ranking queries [Canal et al., 2020]. We observe that nearest neighbor queries perform competitively to ranking queries, as illustrated in Fig. 6. Again, we fix $N = 20$, $D = 2$, utilize $K_0 = 20$ initial random queries, and examine the performance of Info-NN queries of sizes $3, 4$, and 5 and ranking queries of sizes 3 and 4.

We observe that the nearest neighbor query exhibits similar performance to randomly selected ranking queries, despite the ranking queries containing twice as many paired comparisons as a nearest neighbor query. Info-NN-3 queries are able to match randomly selected ranking queries of the same size, while Info-NN-4 queries exceed the performance of randomly selected ranking queries of size 3, while almost matching the performance of actively selected size 3 ranking queries. Employing Info-NN can nearly compensate for the difference in information between nearest neighbor and ranking queries, highlighting an advantage in the trade-off between complexity and "information density" (the number of triplets contained in one query).

B.3 ACTIVE SELECTION COMPUTATIONAL COMPARISON

While our mutual information computation strategy is similar, utilizing NN queries results in computational advantages when compared to the ranking query used in Canal et al. [2020]. To compare the time discrepancy between computing mutual information for ranking and nearest neighbor queries, we perform 10 iterations of our embedding technique, and record the amount of time it takes to compute the mutual information for each object as the reference object. We then report the average and standard deviation of the times taken. We use the same parameters for each active learning algorithm, such

Table 1: Timing results, in seconds, for computing mutual information for nearest neighbor and ranking queries. Experiments performed on 2019 MacBook Pro, 2.6 GHz 6-Core Intel i7, 16 GB RAM.

	$M = 2$	$M = 3$	$M = 4$
NN	0.0265 ± 0.0036	0.1509 ± 0.0044	0.6634 ± 0.0812
Ranking	0.6605 ± 0.0583	8.5394 ± 0.3400	175.0046 ± 93.2602

as number of queries to consider and number of distance samples generated. In Table 1, we report the average amount of time it takes to compute the mutual information for a given reference object for differently sized queries in actively selecting nearest neighbor queries using Alg. 1 and the method presented in Canal et al. [2020]. The drastic discrepancy in timing between the two methods is due primarily to the fact that the nearest neighbor mutual information computation does not require computation for all possible permutations of the set of M items, whereas the ranking query does.

C CLASSIFICATION

C.1 ALGORITHMS

A description of the active classification framework and the complete Info-NN query strategy utilized to select samples for labeling, is below.

Algorithm 2 Active Learning for Classification

Require: Dataset $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^{N}$, batch size B, number of classes C, number of samples n_s
  $\mathcal{L}_0 \leftarrow \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{j}$ initial (balanced) labeled dataset
  $\mathcal{U}_0 \leftarrow \{\boldsymbol{x}_i\}_{i=j+1}^{N}$
  $M_0 \leftarrow$ Model trained on $\mathcal{L}_0$
  for $k = 1, \ldots, K$ **do**
    $\mathcal{B}_k \leftarrow$ Info-NN-$m(M_{k-1}, \mathcal{L}_{k-1}, \mathcal{U}_{k-1}, B, C, n_s)$
    $\mathcal{L}_k \leftarrow \mathcal{L}_{k-1} \cup \{(x_i, y_i) : x_i \in \mathcal{B}_k\}$
    $\mathcal{U}_k \leftarrow \mathcal{U}_{k-1} \backslash \mathcal{B}_k$
    $M_k \leftarrow$ Model trained on $\mathcal{L}_k$
  end for

Algorithm 3 Info-NN-M

Require: Model $\mathcal{M}$, labeled set $\mathcal{L}$, unlabeled set $\mathcal{U}$, batch size B, number of classes C, number of samples n_s
  $\boldsymbol{Z}_{\mathcal{L}}$ = Compute Embedding ($\mathcal{L}$)
  $\boldsymbol{Z}_{\mathcal{U}}$ = Compute Embedding ($\mathcal{U}$)
  $Q \leftarrow \{\}$ (Set of candidate queries)
  for $u \in \boldsymbol{Z}_{\mathcal{U}}$ **do**
    $\text{NN}_u \leftarrow$ Top M nearest neighbors
    $Q_u \leftarrow u \cup \text{NN}_u$
    $Q \leftarrow Q \cup Q_u$
  end for
  $I \leftarrow$Info-NN-distances$(\boldsymbol{Z}_{\mathcal{U}}, Q, n_s)$
  $G(\mathcal{U}) \leftarrow$ K-Means Clustering ($\mathcal{U}, \mathcal{L}$)
  $\mathcal{B} \leftarrow$ unlabeled samples corresponding to top values of I from every cluster

C.2 EXPERIMENTAL DETAILS

Computational infrastructure The experiments were performed on a combination of three desktop machines with the following configurations:

1. A 3.80GHz 16-Core Intel $i7 - 9800X$ CPU and an Nvidia Quadro RTX 5000 GPU

2. A 2.10GHz 20-core Intel Xeon Gold 6230 CPU and four Nvidia Quadro RTX 6000 GPUs

Datasets. Below are the details of the real world datasets used on classification experiments.

- MNIST [LeCun et al., 1998] is a dataset of black and white images of handwritten digits belonging to 10 classes and consists $60,000$ training samples and $10,000$ test samples.
- CIFAR-10 [Krizhevsky et al., 2009] is a dataset consisting of colour images belonging to 10 classes with $50,000$ training samples and $10,000$ test samples.
- SVHN [Netzer et al., 2011] consists of digits (10 classes) from natural scene RGB images with $73,257$ training samples and we use $10,000$ samples for testing the accuracy of the learned models.

Baselines. The details of the baseline active labeling methods used are as follows.

- BatchBALD: Samples are selected according to the algorithm described in Kirsch et al. [2019]. The algorithm uses Monte-Carlo (MC) sampling to compute joint probabilities of the different labeling configurations in a batch of samples which is very memory intensive. This requires the pool of unlabeled data to be sub-sampled in order for the computations to be feasible. The number of MC samples for the computations and the size of the pool set was determined by the memory associated with the GPUs. We use 10^3 MC samples and the sizes of the pool set used were $20,000$ for MNIST and $5,000$ for both CIFAR-10 and SVHN respectively. We would like to note here that we did not perform an extensive experimentation to determine an optimal configuration of the number of MC samples and size of the pool set but decided a configuration based on the settings that did not result in running out of GPU memory.
- K-Center: Optimal samples that achieve the desired coverage, based on the distances in the embedding space learned by the network, are selected. This method is based on the algorithm described in Sener and Savarese [2017].
- MaxEntropy: The top unlabeled samples with the maximum entropy, computed based on the class probabilities predicted by the model, are chosen.
- Random: A batch of samples is drawn at random from the pool for labeling.

Models and training methodology. In all the experiments, the models are trained from scratch at every active learning cycle. The performance reported is measured on a holdout test set comprising of $10,000$ samples in all the experiments.

MNIST: For experiments on the MNIST dataset, we use a model similar to the one used in Kirsch et al. [2019]. Specifically, we use a CNN consisting of two convolutional blocks followed by two fully connected layers. The two convolutional blocks consist of 32 and 64 filters of kernel size 5, each followed by layers of dropout, max-pooling and relu units. The two fully connected layers, of size 128 and 10 respectively, also have a dropout unit between them. We use a probability of 0.5 for all dropout units.

The data inputs to the model are normalized and batch sizes of 64 and 128 are used while training and testing respectively. We use the Adam optimizer with a learning rate of 0.001. Since the size of the labeled set used in these experiments is small compared to the entire dataset, we use early stopping to ensure that the model does not overfit to the training data. We use a validation set of size 100 consisting of 10 samples from every class selected at random and we stop training after 10 consecutive epochs of increasing validation loss.

CIFAR-10 and SVHN: For both the datasets, we use a ResNet-18 [He et al., 2016] to conduct the experiments. While training, the data inputs are normalized along with augmentation techniques consisting of random cropping with an output size of 32 and a padding of 4 and random horizontal flipping. The model is trained for 250 epochs using the Adam optimizer with a learning rate of 0.001 in combination with the cosine annealing scheduler. A batch size of 128 is used for both training and testing.

Mutual Information estimation for Info-NN: The parameter μ is set equal to the maximum value of the inter-sample distances in the embedding space. The variance for the normal distribution of distances is set as the variance of all the distances in the embedding space and 100 samples from the distributions are used for inference. These values were found to work well in all the experiments and an extensive and a systemic search for these hyperparameters was not performed.

C.3 ADDITIONAL RESULTS

Performance plots. We compare the performance of Info-NN with and without clustering (top b samples are selected solely based on informativeness) on MNIST. The results are illustrated in Fig. 7 where we can observe the improved

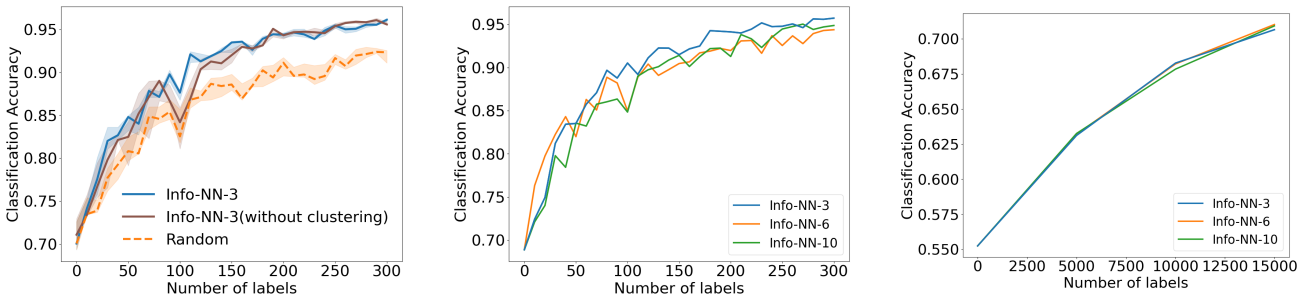

Figure 7: Active classification experiments:Comparison of the performances of Info-NN with and without clustering using a batch size of 3 on the MNIST dataset (left). Performance comparison between Info-NN queries of different lengths on MNIST (center) and CIFAR-10 (right) datasets.

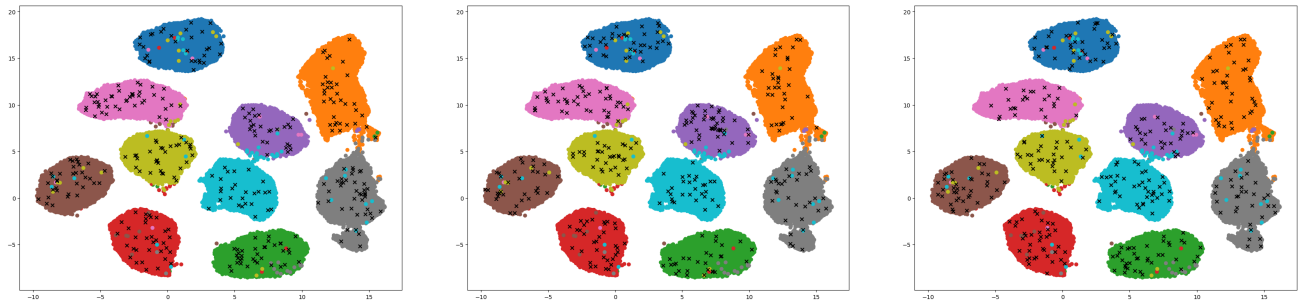

Figure 8: Visualization of samples selected on MNIST with MaxEntropy (left), Info-NN-3 (center) and K-Center (right) querying strategies, generated using UMAP [McInnes et al., 2018]. Each of the blobs correspond to one among the 10 classes and the samples selected are indicated by black crosses.

performance realized by Info-NN when combined with clustering.

Also, we conducted experiments on MNIST and CIFAR-10 datasets to determine the optimal query length for Info-NN. In Fig.7, we can observe that queries of length 3 resulted in the best performance on MNIST, significantly outperforming queries of longer lengths. On CIFAR-10, while all of them seem to exhibit a similar performance, queries of length 3 outperform the others consistently. Thus, we use queries of length 3 in all the experiments with supervised classification.

Visualizations. The samples selected by different active methods are illustrated in Fig. 8. We can observe that MaxEntropy tends to select redundant informative samples indicated by clusters of black crosses and K-Center selects samples to ensure diversity indicated by the more distributed placement of the selected samples. In the case of Info-NN, we see a combination of clustered and distributed samples likely selecting both informative and diverse samples.

References

Gregory Canal, Stefano Fenu, and Christopher Rozell. Active ordinal querying for tuplewise similarity learning. In *AAAI*, pages 3332–3340, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Maurice Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

Andreas Kirsch, Joost van Amersfoort, and Yarin Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. In *Advances in Neural Information Processing Systems*, pages 7026–7037, 2019.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Priyadarshini Kumari, Ritesh Goru, Siddhartha Chaudhuri, and Subhasis Chaudhuri. Batch decorrelation for active metric learning. In *IJCAI*, 2020.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

R Duncan Luce. Individual choice behavior, 1959, 1959.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov): 2579–2605, 2008.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

Nir Rosenfeld, Kojin Oshiba, and Yaron Singer. Predicting choice with set-dependent aggregation. *arXiv preprint arXiv:1906.06365*, 2019.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

Omer Tamuz, Ce Liu, Serge Belongie, Ohad Shamir, and Adam Tauman Kalai. Adaptively learning the crowd kernel. *arXiv preprint arXiv:1105.1033*, 2011.

Kenneth E Train. *Discrete choice methods with simulation*. Cambridge university press, 2009.

Amos Tversky and Daniel Kahneman. The framing of decisions and the psychology of choice. *science*, 211(4481):453–458, 1981.