Wrangling with the paper “Unsupervised Space Partitioning for Nearest Neibhour Search”

Michael Darmanis*
Stathis Kotsis*
mdarm@di.uoa.gr
efstathkot@di.uoa.gr
National and Kapodistrian University of Athens
M149 - Database Systems, Spring 2023

ABSTRACT

This report presents a series of extensions to the paper ”Unsupervised Space Partitioning for Approximate Nearest Neighbour Search”. The original focus of the paper is to develop an efficient approach for approximate nearest neighbour search in high-dimensional spaces, which is a critical problem in various domains such as image retrieval, recommendation systems, and natural language processing. The improvements described in this report aim at further optimising the efficiency and accuracy of the approach. First, Principal Component Analysis (PCA) is introduced as a preprocessing step to reduce the dimensionality of the feature space and retain the most informative components. Second, the Mahalanobis distance metric is used to account for correlations between variables, making it more suitable for high-dimensional data than the traditional Euclidean distance. Third, a convolutional neural network is implemented as an additional model for the ensembling mechanism and integrated into the existing structure. Fourth, the ensembling process is modified to incorporate different model types, including linear models, neural networks and convolutional neural networks (CNNs), while adding seamless integration for any type of potential future model, allowing for greater flexibility in model selection. Fifthly, product vector quantization will be implemented and added as an option for more fine-grained unsupervised neighbour search on top of the existing model. Finally, an attempt to integrate the Hierarchical Navigable Small Worlds (HNSW) algorithm is documented, although this extension is only partially implemented due to time constraints. The report provides the rationale, mathematical formulations, and initial Python code for each enhancement. It also provides space to document experimental results comparing the original and improved settings. Through these improvements, the report contributes to the ongoing effort to optimise approximate nearest neighbour search.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; *Learning paradigms*; Image processing algorithms; • **Mathematics of computing** → *Mathematical software performance*; Approximation algorithms.

KEYWORDS

approximate nearest neighbor search; unsupervised space partitioning; principal component analysis; mahalanobis distance; ensemble models; convolutional neural networks; hierarchical navigable small worlds; high-dimensional data

*Both authors contributed equally to this project.

1 INTRODUCTION

This report summarises the improvements made to the paper entitled “Unsupervised Space Partitioning for Approximate Nearest Neighbour Search”. The aim of these improvements was to increase the performance and capabilities of the proposed approach. The improvements include the use of PCA as a pre-processing stage, the adoption of the Mahalanobis distance metric, and modifications to the ensembling process to allow for the incorporation of linear, neural, and CNN models. In addition, the report documents an attempt at improvement involving partial implementation of the Hierarchical Navigable Small Worlds (HNSW) approach. The report will outline the rationale behind each improvement and provide a space to record the results of experiments conducted based on the original settings.

2 IMPLEMENTED IMPROVEMENTS

2.1 Principal Component Analysis (PCA) Preprocessing

Principal Component Analysis (PCA) [11] was used as a preprocessing step to reduce the dimensionality of the feature space. By retaining the principal components that explain the most variance in the data, it is possible to reduce computational complexity without losing critical information. This reduction in dimensionality is particularly beneficial for high-dimensional datasets such as images, where each pixel is considered a feature. PCA can help filter out noise and irrelevant features, leading to better generalisation and faster training times.

The mathematical formulation of PCA is as follows

$$Y = X \times W \tag{1}$$

where Y is the matrix of principal components, X is the original data matrix, and W is the matrix of loadings or weights.

For datasets such as MNIST and SIFT, which contain high-dimensional data points representing images, the use of PCA as a preprocessing step is expected to be advantageous. In MNIST, each image is 28x28 pixels, resulting in 784 dimensions, and in SIFT, each image is represented by a 128-dimensional feature vector. PCA can significantly reduce the number of dimensions while retaining the features essential for classification or nearest neighbour search.

In addition, PCA can sometimes improve the performance of certain algorithms by removing correlations between features, which can be beneficial for algorithms that assume feature independence.

It’s important to note that the choice of the number of principal components to retain is a critical decision. In our implementation, we decided to keep the components that retained 95% of the original variance.

2.2 Mahalanobis Distance Metric

The Mahalanobis distance metric was introduced as an alternative to the Euclidean distance commonly used in various machine learning algorithms. The Mahalanobis distance takes into account correlations between variables and variance along different dimensions, making it more appropriate for high-dimensional data where features may not be independent.

The Mahalanobis distance between two points x and y is defined as

$$D_M(x, y) = \sqrt{(x - y)^T \times S^{-1} \times (x - y)} \tag{2}$$

where S is the covariance matrix of the data.

Unlike the Euclidean distance, which treats each dimension equally, the Mahalanobis distance takes into account the dispersion and correlation of data in different dimensions. This can be particularly useful in scenarios where the size and distribution of features vary, as it standardises the data based on their distribution.

For datasets such as MNIST and SIFT, where the features represent pixel intensities or local image descriptors, the distribution and correlations of the features are important aspects to consider. The Mahalanobis distance can capture the intrinsic geometry of such datasets more accurately compared to the Euclidean distance.

Furthermore, although more computationally intensive, the Mahalanobis distance is scale-invariant and correlates well with human perception of similarity. It's particularly effective in applications such as image recognition, anomaly detection and clustering, where understanding the underlying geometry of the data is critical.

2.3 Ensembling Process with Different Model Types

Ensembling is a broad category of machine learning techniques that combine multiple models to make predictions. Those models are called base learners. If all the models in the ensemble are of the same type then they are called homogeneous ensembles while they are called heterogeneous learners if different learning algorithms are used. The three main categories of Ensembling are bagging, stacking and boosting.Bagging mainly involves feeding different samples of a dataset into base learners in parallel and taking usually the average of the predictions as the output of the model.The average of the predictions is only one possible method to obtain the final results. Weighted averaging and voting are also viable options. With Stacking, various different base learners fit on the same data samples and the output of the base learners is fed into a Meta Learning Model with the initial dataset labels as the targets. The meta learner decides how to combine the models' predictions.Finally boosting[10] is based on sequential models. In particular each base learner makes predictions on the same data and each subsequent model tries to correct the mistakes of previous models by assigning bigger weights to the datapoints wrongly predicted. The final output is a weighted average of the predictions from each model according to it's predictive power. In essence, boosting tries to convert weak learners, which perform slightly better than random guessing to strong learners, models with high accuracy[12].

In the original work the ensemble model is based on Boosted Search Forest[6] which uses AdaBoost[9] for the correction of wrong partitions predictions in a sequential manner. Each model focuses on the points which were wrongly partitioned in the previous model and tries to improve those predictions based on a loss function.

The ensemble process has been modified to allow different types of models to be included. This change allows greater flexibility in the type of models that can be included in the ensemble. In particular, instead of training tree nodes of the same model type, the implementation has been extended to support the construction and training of linear regression models, convolutional and neural networks. By combining these different types of models, the ensemble can potentially achieve higher performance than any single model alone. This is particularly useful for datasets such as MNIST, which contains images of handwritten digits, and SIFT, which contains local image descriptors. CNNs can be effective at capturing spatial patterns in images, while linear models and neural networks can capture other types of relationships in the data: for example, CNNs can learn to recognise the strokes in handwritten digits, while a neural network can capture how these strokes combine to form different numbers. Linear models can provide a baseline that's less prone to overfitting. Linear models are relatively simple and interpretable, but may not capture complex patterns in the data. Neural networks are more flexible and can model non-linear relationships, but are prone to overfitting on small data sets. Convolutional Neural Networks (CNNs) are particularly effective for image data as they can automatically learn to recognise patterns in the spatial structure of the data.

The process of ensemble training with different models can be seen in Figure 1. The first base learner, a Neural Network here, is fed the points in the dataset and does some predictions, then the datapoints are weighted and the mispredicted are assigned higher weights. After that they are fed to a cnn model which corrects some mistakes from the previous model. Following the same process, the final model is then able to classify all the partitions correctly.

Furthermore, we argue that the support of different models could be used to design a Stacking Ensembling method, as an alternative to Boosting, where the final predictions and the task of optimally combining the models is done by a Meta Learner, when the complexity of the dataset is high and simple averaging schemes do not produce the desired results. We'll leave the design of such a model and its integration to future work.

2.4 CNN Implementation and Integration

The ensemble was extended to include Convolutional Neural Networks (CNNs), which are particularly adept at handling image data by efficiently capturing spatial structures[4]. This inclusion could potentially be beneficial for the MNIST dataset, with its grey-scale images of handwritten digits, as CNNs excel at recognising intricate spatial patterns such as strokes and lines. Similarly, for the SIFT dataset, which deals with feature descriptors for image matching, CNNs are able to identify key features in images. In particular, the CNN model is designed to work for the MNIST dataset and it's variants (Fashion-MNIST), but extension to other datasets is trivial

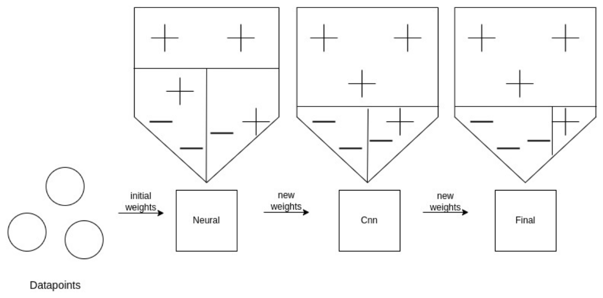

Figure 1: This figure is a simplification of the main scheme used for training the base learners for our task, which is similar to a classification task.

and can be easily achieved with just a few changes to the implementation, provided that the feature dimensions can be reshaped into a square matrix. The convolutional neural network architecture used consists of three convolutional layers and two linear layers.The input to the convolutional neural network is a tensor of shape (batch size,1,28,28), which is a batch of the square image matrices of the MNIST dataset. The first convolutional layer applies a convolution with 6 kernels, kernel size 3x3 and stride 1. This is followed by ReLu activation, a max-pooling layer with kernel size 2x2 and stride 2, and a batch normalisation layer. The second convolutional layer uses 16 kernels of size 3x3 and stride 1, followed by ReLu activation and max pooling of kernel size 2x2. The final convolutional layer uses 120 kernels of size 3x3 followed by ReLu. The output is then flattened and fed into a linear layer with ReLu activation. This produces a vector of size n_hidden for each batch, where n_hidden is specified by the user. The default value is 84.

You can see a rough sketch of the CNN architecture, at Figure 5 of Appendix A.1, and the number of layers and parameters in each layer.

2.5 Ensembling with Product Vector Quantization Pipeline

Since the main task of the unsupervised partitioning model is to divide the space into multiple partitions, each of which is likely to contain potential neighbours, the task of retrieving the top-k neighbours of query points from the generated bins can alternatively be performed using a simple product quantization. Product quantization [3] is the process of compressing data into a smaller space. This is achieved by encoding each data vector into a smaller encoding, a symbolic representation. This method drastically reduces the memory consumption of the data vectors. In order to better fine-tune the top-k neighbours, we can therefore use a two-step pipeline where

(1)The Unsupervised Space partitioning method first divides the space into multiple partitions, reducing the search space for finding the top-k neighbours.

(2)The points in each produced bin are encoded and inserted into an index using a quantization process. After the index is created the top-k neighbours for each point in the bin can be retrieved.

The quantization process is as follows : The points in each partition are split into subvectors and for each subvector a number of cluster centroids are produced. The number of centroids and the dimensions of the subvectors can be easily parametrized by the user. Then the distances of the centroids for each subvector are calculated and stored since they will be used as an index later. The centroids are then fit using a clustering Algorithm like K-means and each subvector is assigned to it’s closest centroid, thus building a codebook which includes all the data points’ unique encodings. These are the unique symbolic representations of all the data points. Since we now have the symbolic representation of the datapoints and the distances of the centroids, finding the top-k neighbours for each data point is easy.The query point’s subvectors distances with all the other points’ subvectors in the codebook are retrieved using a lookup operation in the stored distances’ tables and added together to form the total distance of the query point with any other point. The top-k smallest distances define the top-k nearest neighbours. Since the encoding of each data point is very small in comparison to the initial data point’s dimensions, this method is characterized by very low memory usage and is great when low memory availability is expected. Testing the method on our dataset it seems that the real bottleneck is the number of cluster centroids and the number of samples in each bin since they must be compared with each other to store the distances. In our case 1000 query points from the test dataset were fed into the quantization process after the space partitioning took place and the top-10 neighbours were queried. The results for candidate set sizes of 3000-4000 points show that the method can be used as an additional step, to index the points of each returned bin and find the nearest neighbours of each one of them after the distances have been calculated and stored but optimization steps are required in order to tackle the bottlenecks. Such optimization methods are available[**?**]opmethods but they are out of the scope of this work. The time required for finding the top-10 nearest neighbours for varying number of queries and quantization parameters using the simple quantization approach are presented in 1 .

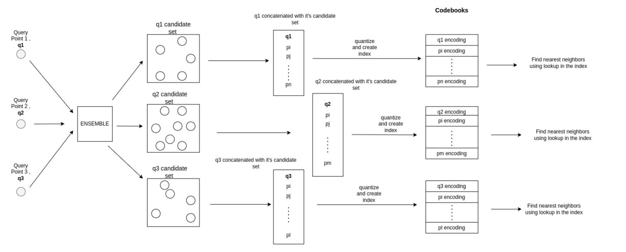
Codebooks
q1 concatenated with it's candidate set
q1 candidate set
Query Point 1 , q1
q1
pi
pj
pn
quantize and create index
q1 encoding
pi encoding
pn encoding
Find nearest neighbors using lookup in the index
q2 concatenated with it's candidate set
q2 candidate set
Query Point 2 , q2
ENSEMBLE
q2
pi
pj
pm
quantize and create index
q2 encoding
pi encoding
pm encoding
Find nearest neighbors using lookup in the index
q3 concatenated with it's candidate set
q3 candidate set
Query Point 3 , q3
q3
pi
pj
pl
quantize and create index
q3 encoding
pi encoding
pl encoding
Find nearest neighbors using lookup in the index

Figure 2: Overview of the 2 step pipeline feeding the results of the Ensemble to the quantization process and then querying for the nearest neighbours.

3 EXPERIMENTAL SETUP & RESULTS

In this section, we present the experimental setup and results obtained from various modifications made to the baseline model presented in the original paper. The experiments had the same settings as the original paper, using the same datasets and evaluation

Table 1: Query time for finding the top-10 nearest neighbours for varying number of query points, of dimensions 784, 4 centroids per subvector and 16 subvectors, in an average candidate size of 3000

# Query points	Time (m)
100	1.85
500	5.92
1000	15.7

metrics and protocols[2]. Below, we enumerate the different modifications:

(1) **Baseline Model:** As a reference, we first established a baseline by using the original models presented in the paper, which includes Neural Networks and Logistic Regression.
(2) **Principal Component Analysis (PCA) Preprocessing:** We applied PCA as a preprocessing step to reduce the dimensionality of the feature space. This experiment aimed to understand the effect of dimensionality reduction on the model's performance.
(3) **Multimodel Ensembling:** We modified the ensembling process to incorporate different model types, including linear models and neural networks (not convolutional neural networks). This aimed to observe the effects of using a diverse set of models in the ensemble on the performance metrics.
(4) **Mahalanobis Distance Metric:** We adopted the Mahalanobis distance metric as an alternative to the Euclidean distance. This experiment aimed to understand the impact of accounting for correlations between variables in high-dimensional data on the model's performance.
(5) **Convolutional Neural Networks (CNNs):** In this experiment, we introduced Convolutional Neural Networks (CNNs) for feature extraction from raw data. The experiment involved the creation of a CNN architecture with several convolutional and pooling layers, followed by fully connected layers. The objective was to explore the efficacy of CNNs in capturing spatial patterns in the data and evaluate their performance compared to traditional Neural Networks. It's important to note that we did not apply CNNs on SIFT features because padding was required, and we had time limitations.

3.1 Results and Discussion

3.1.1 *RNN-recall.* In terms of quality and accuracy of partitioning, it's interesting that PCA outperformed the original method for the 16-bin partition, especially given the relatively small feature space (see Figure 3a). However, the performance of PCA for MNIST was poor. This could be due to PCA's tendency to capture global structures, which may not be as effective when dealing with high-dimensional data such as MNIST. On the contrary, for the 256 partition scheme (see Figures 3c and 3d), all implementations showed a degradation in performance. In particular, Mahalanobis distance generated a single large bin containing almost the entire dataset, as shown in Figures 3 and 4. This could be because the Mahalanobis distance takes into account the covariance of the data and may not be able to distinguish individual clusters in high-dimensional space, leading to ineffective partitioning. The results suggest that the modifications are not generalisable and are not recommended for large space partitions.

The multi-model ensemble also performed as expected, although it had a similar accuracy to the original model in the SIFT dataset (see Figure 3a). One of the likely reasons for this is the inherent complexity and high dimensionality of the datasets. When different models with their own assumptions and biases are ensembled, the combined model may not necessarily account for the structure of the data effectively. In addition, ensembling can sometimes lead to increased computational overhead, which may have negated any performance gains.

One reason for this may be that the loss function does not effectively capture the complexity of high-dimensional data or the covariance structure. This is evident in the degradation of the results for the 256 partition scheme, where the implementations performed poorly in partitioning the space. A potential improvement could be to include additional terms in the loss function (see Appendix A.1) that take into account the geometric and distributional properties of the data, or to explore alternative distance metrics and data representation techniques that are more appropriate for high-dimensional spaces.

3.1.2 *Average query execution time.* In terms of average query execution time, all methods showed faster response times for RNN-recall, especially with the 16-bin partition (see Table 2). This is expected for PCA as it reduces the dimensionality of the feature space, resulting in less computational overhead. Similarly, the Mahalanobis distance may be faster as it may effectively eliminate less important dimensions, thus reducing the computational time. The table shows that for MNIST with 1 bin, PCA and Mahalanobis had query times of 0.30 and 0.41, respectively, compared to 0.67 for the original method, an improvement of about 55% and 39%, respectively. For CNN, the query times were even lower at 0.23 for MNIST with 1 bin, which is an improvement of about 66% compared to the original method. However, CNN was not applied to SIFT due to padding requirements and time constraints.

4 CONCLUSION

In this report, several extensions to the original approach presented in "Unsupervised Space Partitioning for Approximate Nearest Neighbour Search" were thoroughly investigated. The extensions included: Principal Component Analysis (PCA), Mahalanobis distance metric, multi-model ensembling, Convolutional Neural Networks (CNNs), and a product quantization approach; all of which aimed to improve the performance of the model. The product quantization approach was specifically added as an additional step in the two-step pipeline after the model ensembling, with the goal of creating low memory indexes for the data points in each partition and performing a more fine-tuned search for the K nearest neighbours in each bin.

However, the experimental results exhibited inconsistent performance. PCA demonstrated potential in certain configurations but was not as effective with high-dimensional datasets such as MNIST. The Mahalanobis distance metric encountered difficulties

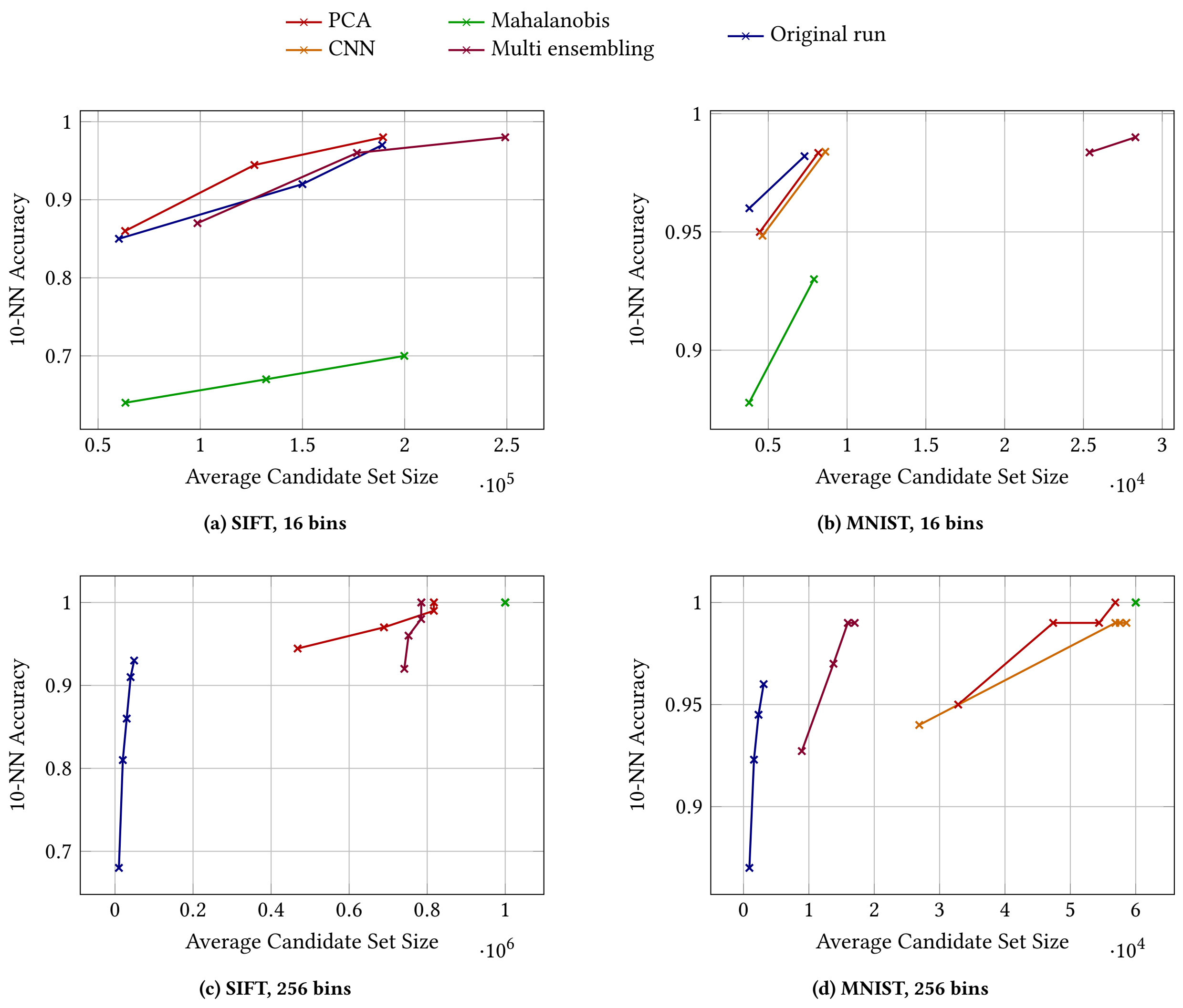

Figure 3: Comparing our method to the original space-partitioning baseline. X-axis: how many candidates are in the set. Y-axis: accuracy of finding the 10 nearest neighbours (higher and more to the left is better). We combined three models, but only show the best one to keep the plot simple.

in handling high-dimensional spaces and often generated overly generalized partitions. CNNs indicated proficiency in decreasing query execution times but were constrained by specific data requirements such as the necessity for padding. With regard to multi-model ensembles, it is important to recognize that ensembling does not necessarily lead to increased accuracy if the individual models are already optimized. In this context, the ensemble approach did not yield the anticipated improvement, likely due to the complexity involved in effectively integrating models for high-dimensional data.

Future research could explore adaptive techniques, alternative loss functions, and hybrid approaches to address the inherent challenges of unsupervised space partitioning. This study could serve as a starting point for understanding the potential and limitations of different preprocessing techniques, and lays the groundwork for innovative solutions in the area of approximate nearest neighbour search.

REFERENCES

[1] Juan Aguaron and Jose Maria Moreno-Jimenez. 2003. The geometric consistency index: Approximated thresholds. *European journal of operational research* 147, 1 (2003), 137–145. https://doi.org/10.1016/S0377-2217(02)00255-2

[2] Abrar Fahim, Mohammed Eunus Ali, and Muhammad Aamir Cheema. 2023. Unsupervised Space Partitioning for Nearest Neighbor Search. In *Proceedings 26th International Conference on Extending Database Technology, EDBT 2023, Ioannina, Greece, March 28-31, 2023*, Julia Stoyanovich, Jens Teubner, Nikos Mamoulis, Evaggelia Pitoura, Jan Mühlig, Katja Hose, Sourav S. Bhowmick, and Matteo Lissandrini (Eds.). OpenProceedings.org, 351–363. https://doi.org/10.48786/edbt.2023.28

[3] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2011. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 1 (2011), 117–128. https://doi.org/10.1109/TPAMI.2010.57

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. ImageNet Classification with Deep Convolutional Neural Networks. 60, 6 (2017). https:

Table 2: Average query times for a 16-bin partition on MNIST and SIFT datasets. Measurements' uncertainties, for a 64% confidence interval, are provided in parentheses.

	MNIST		SIFT		
	1 bin	2 bins	1 bin	2 bins	3 bins
Original Method	0.67	0.80	0.90	0.96	1.24
	(0.08)	(0.11)	(0.15)	(0.17)	(0.22)
PCA	0.30	0.22	0.25	0.36	0.42
	(0.05)	(0.04)	(0.03)	(0.14)	(0.06)
Mahalanobis	0.41	0.21	0.32	0.36	0.42
	(0.07)	(0.06)	(0.06)	(0.10)	(0.04)
CNN	0.23	0.25	-	-	-
	(0.07)	(0.03)	(-)	(-)	(-)
Multi Ensembling	0.30	0.25	0.25	0.36	3.42
	(0.15)	(0.04)	(0.02)	(0.08)	(0.04)

//doi.org/10.1145/3065386

[5] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. 2014. *Mining of Massive Datasets* (2 ed.). Cambridge University Press. https://doi.org/10.1017/CBO9781139924801

[6] Zhen Li, Huazhong Ning, Liangliang Cao, Tong Zhang, Yihong Gong, and Thomas S. Huang. 2011. Learning to Search Efficiently in High Dimensions. In *Proceedings of the 24th International Conference on Neural Information Processing Systems* (Granada, Spain) *(NIPS'11)*. Curran Associates Inc., Red Hook, NY, USA, 1710–1718.

[7] Prasanta Chandra Mahalanobis. 2018. On the generalized distance in statistics. *Sankhyā: The Indian Journal of Statistics, Series A (2008-)* 80 (2018), S1–S7. https://doi.org/10.1007/s13171-019-00164-5

[8] Yu. A. Malkov and D. A. Yashunin. 2018. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. https://doi.org/10.48550/arXiv.1603.09320 arXiv:1603.09320 [cs.DS]

[9] Robert E. Schapire. 2013. *Explaining adaboost.* Springer Berlin Heidelberg, 37–52. https://doi.org/10.1007/978-3-642-41136-6_5 Publisher Copyright: © Springer-Verlag Berlin Heidelberg 2013..

[10] Robert E. Schapire. 2003. The Boosting Approach to Machine Learning An Overview.

[11] Svante Wold, Kim Esbensen, and Paul Geladi. 1987. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems* 2, 1 (1987), 37–52. https://doi.org/10.1016/0169-7439(87)80084-9 Proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists.

[12] Zhi-Hua Zhou. 2012. *Ensemble Methods: Foundations and Algorithms* (1st ed.). Chapman & Hall/CRC. https://doi.org/10.5555/2381019

A FURTHER ATTEMPTED IMPROVEMENTS

A.1 Loss function

In our initial approach, the loss function comprised of two terms: accuracy-related loss and bins distribution-related loss. However, to further improve the network's performance, we attempted to introduced an additional term to account for geometric consistency[1]. This bodes well with the assumption that both dataset and queries come from the same distribution. The updated loss function is as follows:

$$L = \eta \cdot L_{\text{bins}} + L_{\text{diff\_sum}} + \lambda \cdot L_{\text{geom}}$$

where L is the total loss, η is a scaling factor, L_{bins} is the bins distribution-related loss, $L_{\text{diff\_sum}}$ is the accuracy-related loss, λ is a scaling factor for the geometric consistency loss, and L_{geom} is the geometric consistency loss.

(1) **Accuracy-related loss, $L_{\text{diff\_sum}}$:**
This loss ensures that the output of the network correctly classifies the input data points. We compute a cross-entropy loss between the predicted output distribution and the ground truth. The loss is then weighted by a set of pre-defined weights. Mathematically, $L_{\text{diff\_sum}}$ is computed as:

$$L_{\text{diff\_sum}} = \frac{1}{N}\sum_{i=1}^{N} \text{weights}_i \cdot \Big(-\text{one\_hot}(\text{maj\_bins}_i) \cdot \log(\text{outputs}_i + 10^{-9}) \Big)$$

where N is the batch size, weights are the predefined weights, one_hot is the one-hot encoding function, majority_bins are the bins in which the majority of points are, and outputs are the predicted output distributions.

(2) **Bins distribution-related loss, L_{bins}:**
This loss term encourages the network to have a more uniform distribution of data points across the output bins. It is computed by taking the sum of the top k values in each bin of the output distribution, where k is equal to the batch size divided by the number of bins (n_{bins}):

$$L_{\text{bins}} = -\frac{1}{N}\sum_{j=1}^{n_{\text{bins}}}\sum_{i=1}^{k} \text{outputs}_{i,j}$$

where N is the batch size and outputs are the predicted output distributions.

(3) **Geometric consistency loss, $L_{\mathbf{geom}}$** (Not implemented but provided code):
This loss ensures that the geometric relationships between the input data points are preserved in the output space. It is computed as the mean squared error between the pairwise distances of the input data points and the pairwise distances of the output data points:

$$L_{\text{geom}} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \left(\text{dist}_{\text{output}}(i, j) - \text{dist}_{\text{input}}(i, j)\right)^2$$

where N is the batch size, and $\text{dist}_{\text{output}}$ and $\text{dist}_{\text{input}}$ are the pairwise distances in the output and input spaces respectively.

The addition of the Geometric consistency term was not successfully implemented due to time constraints (see `../src/loss_fn.py`, the commented-out section).

A.2 Hierarchical Navigable Small Worlds (HNSW)

An effort has been made to integrate the Hierarchical Navigable Small Worlds (HNSW) algorithm into our approach (see `../hnsw-pytorch/model.py` path of the repository). The HNSW algorithm is a highly efficient method for approximate nearest neighbour search, especially in high-dimensional spaces. It works by constructing a hierarchy of navigable small-world graphs, where each layer of the hierarchy corresponds to a sparser sample of the data. This structure allows a fast and effective search for the nearest neighbours of a query point.

HNSW is based on the small-world phenomenon, which states that most pairs of nodes in a real-world graph can be connected by a relatively short path of edges. The algorithm exploits this principle by constructing multiple layers of graphs, where the bottom layer contains all the data points, and each subsequent layer is a sparser sample of the points in the layer below. The search process starts at the top layer and navigates down the hierarchy until the nearest neighbours are found in the bottom layer.

However, the actual insertion and search logic has not been fully implemented due to time constraints. Completing the implementation and integrating HNSW into the main ANN algorithms could potentially improve the efficiency of nearest neighbour searches, which is crucial for large and high-dimensional datasets, such as SIFT and MNIST.

A.3 CNN Architecture

Received 25 June 2023

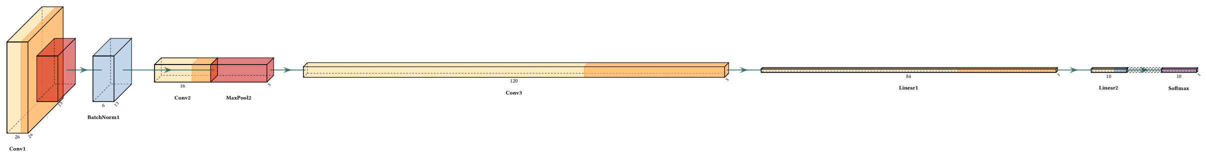

Figure 5: CNN architecture