University College London

COMP0123 Complex Networks and Web (2023/24)

Coursework 2

January 2024

A Network Approach to Image Classification using Community Detection Algorithms and Image Embeddings

Abstract

This report investigates the application of network community detection in image classification, utilising the well-known MNIST dataset as a benchmark. It explores the construction of image networks through two distinct methods of feature vector extraction, namely raw pixel values and image embeddings via EfficientNet. Central to the study is the application of the Louvain method for community detection and a voting mechanism for label assignment. Key findings highlight that image networks constructed using image embeddings more accurately capture community structures and yield classification results comparable to supervised K-Nearest Neighbor algorithms. Despite limitations, including computational resource constraints and the simplistic nature of the MNIST dataset, the results are encouraging and pave the way for future explorations into integrating community detection with image classification.

Contents

1 Background

Image classification is a fundamental and critical task in many real world applications, e.g. road object detection in automatic cars and medical image analysis. This essential problem in the field of computer vision has been mainly studied using emerging deep learning techniques, in particular Convolution Neural Networks (CNNs) are the dominant methods adopted [6]. While CNNs and other deep learning approaches have significant performance in accuracy and speed of image classification, they often require substantial computational resources. Also, these models often possess low interpretability due to their complex architectures [17].

Alternatively, network science diverges as a distinctive branch for understanding complex structure and interactions. Community detection is one of the central aspects of network analysis, which target to group nodes in a network based on the density of their interconnectivity. Such detection algorithms were applied in social network analysis, recommendation systems and also biological networks. In recent years, this method has seen promising results in topic modeling [2, 12], a hot and demanding techniques for grouping a collection of document according to their content.

Given the success of community detection in topic modeling, this report explores the potential of applying similar techniques to the subject of image classification. Specifically, it investigates the creation of network representations of an image dataset, where each image is considered as a node and similarities between raw images are the edges. Furthermore, image embedding, a powerful technique in deep learning that transforms raw image to a feature vector, was employed to transform the image dataset. The network properties of the two network created using raw pixels and extracted features are analysed. In addition, the performace of Louvain modularity method for community detection on these networks is also assessed.

2 Research Questions

In the study, we focused on two primary directions to study the application of network analysis in image classification. These areas are summarised through the following research questions:

2.1 How can we effectively represent images as a network for network analysis?

The first research question addresses the challenge of converting visual data into a network format that accurately captures the inherent relationships and features within images. A robust network representation is crucial for applying analytical techniques, such as community detection algorithms, and can significantly impact the outcomes of image classification tasks. This part of the study seeks to explore various methodologies for embedding images into nodes within a network, determining the most effective ways to establish edges based on image similarity metrics, and understanding how different network structures affect the overall performance and interpretability of the results. By investigating this question, we aim to identify and formulate a network representation that is both computationally efficient and effective in capturing the complex patterns inherent in visual data.

2.2 What is the performance impact of applying community detection algorithms to image classification tasks?

The second research question focuses on evaluating the effectiveness of community detection algorithms when applied to the field of image classification. Community detection has shown promising results in various domains, including social network analysis and topic modeling [2, 12], but its application and performance in image classification are not yet fully understood. This study aims to empirically assess various community detection methods, with a particular focus on the Louvain method, to determine their accuracy, efficiency, and scalability when used with network representations of images. We will explore how the algorithm partition the image networks into communities and the subsequent impact on classification performance. This investigation will provide preliminary insights into the potential benefits of incorporating community detection into image classification workflows.

3 Literature Survey

In this section, we review some literature that are closely related to the work involved in this report.

3.1 Topic Modelling using Network Community Detection

Topic modelling is an unsupervised learning task which involves clustering unclassified documents in a corpus according to the text content. Traditional methodologies include Probabilistic Latent Semantic Analysis (PLSA), Latent Dirichlet Allocation (LDA) and Non-negative Matrix Factorization (NMF). These methods come with the inherent requirement of pre-specifying the number of topics and often suffer when applied to corpora consisting of short texts [2, 7]. In view of these challenges, recent studies have explored the application of network community detection algorithms as an alternative approach to topic modeling [2, 4].

The contrustion of the network representation of a corpus usually requires two steps. Firstly, each document is transformed to a feature vector which corresponding to a node. Then, the pairwise similiarity matrix of these nodes is computed and used as the adjacency matrix to construct a graph. There is a variety of ways to produce a feature vector from text, which typically falls into two main categories [12]: word occurrence and word embedding. Word occurrence methods, such as Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF), rely on the frequency and distribution of words within the document to construct feature vectors, emphasizing the presence and importance of specific terms. On the other hand, word embedding techniques like Word2Vec [11] or GloVe [13] provide a dense representation, capturing not just the occurrence but also the context of words.

Community detection can therefore be applied to the constructed network. The Louvain algorithm is of the unsupervised algorithms for community detection that can operate in time linear to the network size [5]. In [12], Mu et al. provided a thorough empirical study on using the Louvain method for topic modelling on Twitter. Their results showed that some twitter network graphs constructed based on word occurrence and word embeddings out-performed the baseline LDA method.

3.2 Modern Approaches to Image Classification

As outlined in the comprehensive review by Chen et al. [6], modern approaches to image classification have evolved significantly with the development of various powerful Convolution Neural Network (CNN) architectures. The following review provides some of the significant advancements in this domain.

Powerful networks

The field of image classification has been greatly advanced by deep learning, particularly through the development of powerful network architectures. Among these, Inception (also known as GoogLeNet) [15] and Residual Networks (ResNet) [9] have been influential. Inception networks are known employing multiple sized convolutional filters within the same level to capture information at various scales. They significantly increase the width of the network and introduce a lot of computational efficiency and flexibility in capturing image details. On the other hand, ResNet introduced a concept of skip connections to allow training of very deep networks by allowing feature reuse and ameliorating the vanishing gradient problem. These connections essentially allow the network to learn residual functions with reference to the layer inputs, making it easier to train deeper networks.

Emergence of smaller and efficient networks

While networks like Inception and ResNet have led to significant performance improvements, they often require considerable computational resources, which is not always feasible to operate in small mobile devices. This led to the development of more efficient architectures like EfficientNet [16]. EfficientNet, a more recent architecture, scales up CNNs in a more structured way. It uses a compound coefficient to uniformly scale network width, depth, and resolution, which leads to better performance and efficiency. These networks provide a new way to scale up CNNs that require significantly fewer parameters and computational resources than previous models like Inception and ResNet, without compromising on performance. EfficientNet's architecture is based on the observation that balancing network depth, width, and resolution can lead to better performance.

3.3 Image Embeddings

Image embedding (or, image feature extraction) is a technique that involves translating visual information of an image to a real-value vector, so-called *image feature vector* [10]. A model that performs such transformation on images is referred as *image encoder*. A usual practice of composing an image encoder is to remove the final classification layer of a pretrained neural network model. The output vector from the layer before the classification layer is therefore the feature vector of the input image. An image encoder built on a pretrained ResNet model was mentioned in [14].

4 Methodology

4.1 The MNIST Dataset

In our study, we used the well-known MNIST dataset in the field of computer vision for image regconition task. This dataset contains labeled images of handwritten digits, ranging from 0 to 9, and serves as a benchmark for evaluating various image classification techniques [3]. The original MNIST dataset consists of 60,000 training images and 10,000 testing images, each a 28x28 pixel grayscale representation of a single handwritten digit. The pixel values range from 0 to 255, with 255 indicating the brightest value.

Due to the constraints of computational resources, our study employed a reduced subset of this dataset: specifically, we used only 2,000 images for training and 500 images for testing. To maintain the representativeness of the original dataset, we ensured that the class distribution remains uniform across both the reduced train set and test set, with each class of digit proportionally represented. This approach allowed us to conduct meaningful experiments and analyses while accommodating the limitations of our computational setup. Ten example images from the reduced subset and their corresponding labels are shown in Figure 1.

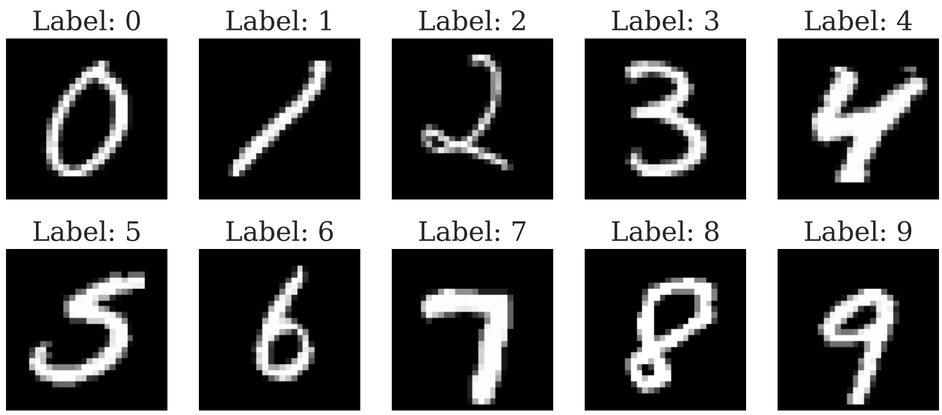

Figure 1: *The first image of each digit in the reduced subset from the original MNIST dataset with the corresponding labels.*

4.2 Constructing the Images Network

4.2.1 Extracting Feature Vectors as Nodes

To create a network linking each image in our dataset, we defined each node in the network using the feature vector of the image. Two methods were employed in our study to obtain the feature vector of an image.

Method 1: Raw Pixel Values

Each 28x28 2-D image was flattened to from a 1-D vector with 784 elements. The vector was formed by stacking the pixel values row by row. Feature vectors produced by this method are referred as ***raw features*** for the rest of this report.

Method 2: Image Embedding with EfficientNet

We utilised the pretrained baseline model `EfficientNet-B0` of EfficientNet [16] to create an image encoder that produces image embeddings as feature vectors. Detail procedures are illustrated in Algorithm 1. Feature vectors produced by this method are referred as ***latent features*** for the rest of this report.

Algorithm 1 Obtain Image Embeddings using Pretrained `EfficientNet-B0`

Input: Reduced training set of MNIST (2000 images)
Output: Feature Vectors for training and test sets

```
Initialise the pretrained EfficientNet-B0 model
Prepend a new preprocessing layer to transform the MNIST image to fit the
  original model input. This layer broadcasts the images to 3 channels and
  performs standardisation.
Remove the last classification layer (original targets consists of 1000 classes).
Append a new fully connected linear layer with output dimension = 10.
Define cross categorical entropy as the loss function.
Define Adam optimizer for training.
for epoch in 1 ... epochs do
    for each batch in Training set do
        Forward pass through the model
        Compute loss
        Backpropagate error and update model weights
Remove the newly added linear layer to obtain the image encoder.
Forward pass the training and test set through the encoder to obtain the latent
  features of each set.
```

4.2.2 Define the Edges using Distance Metrics

To define edges between nodes, we compared two different distance metrics, cosine distance and euclidean distance. For any two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$, the computation of each metric is as follows:

$$CosineDistance(\mathbf{u}, \mathbf{v}) = 1 - \frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}|| \; ||\mathbf{v}||}$$

$$EuclideanDistance(\mathbf{u}, \mathbf{v}) = \sqrt{||\mathbf{u} - \mathbf{v}||^2}$$

Using each of these metrics we can obtain two pairwise distance matrix $D_{\text{cos}}, D_{\text{euc}} \in \mathbb{R}^{\texttt{\#train}+\texttt{\#test}}$ between all training and test images, where `#train` and `#test` are the number of training and test images respectively. The first `#train` rows and columns refer to each training image while the remaining `#test` rows and columns refer to each test image. The element-wise reciprocal matrix of D was therefore used as the weighted adjacency matrix to establish linkages between nodes. However, one pitfall of the network is that we would obtain a complete graph, where community information is not clear since all the nodes are connected even though edges are weighted. To address the problem, we constructed two other types of graphs based on the pairwise distance matrix D.

Type 1: Distances Threshold Network

Instead of adding every pairwise distances as edges, they are added only if they are below a selected threshold value. In our experiment, we varied the value of threshold and compared the performance of different threshold graphs for image classification.

Type 2: K-Nearest Neighbors Network

In this type of graphs, only edges that connect each node to its K nearest neighbors according to the distance metric are added to the graph. In our experiment, graphs constructed with K range from 1 to 50 were evaluated.

4.3 Community Detection and Label Assignment

The Louvain method was adopted for community detection. Since it is an unsupervised algorithm for clustering nodes, a label assignment mechanism is required to determine the label (the written digit) of each node (image) in a cluster. In our study, the dataset are labelled and a voting mechanism is adopted to assign label to the test image in each cluster. Such mechanism allowed us to incorporate the Louvain method in our supervised image classification task.

This mechanism operates within each community identified by the Louvain method. For every community, we examine the labels of the training image it contains and identify the most common label. This majority label is then assigned to all the test data points in the same community. By leveraging the underlying structure discovered by the Louvain method, this voting mechanism provides a simple way to extend labels to unlabeled test data.

4.4 Model Selection

In constructing our model, we introduced several variating factors such as the node feature vectors and the distance metric. To choose the set of factors that provides the optimal performance in classifying our test images. *Stratified K-fold cross validation* was applied to select the optimal factors (or parameters). Cross validation is a common strategy in machine learning for estimating the generalisability of a model and selecting the optimal model parameters. In stratified K-fold cross validation, the training dataset are split into K equal parts, with each part having the same number of data from each class. Using one part as the validation set, the model is trained on the remaining parts and evaluated the validation set. The average of the K metrics obtained is used for model comparison. In our study, 5 folds cross validation was performed and the accuracy on the validation set was used as the evaluation metric. Table 1 listed the parameters and the corresponding range of optimal options or values we searched from.

Parameter	**Range**
feature vectors	raw features / latent featuers
distance metric	cosine / euclidean
threshold (for Distances Threshold Network)	$5^{th}, 10^{th}, ..., 100^{th}$ percentiles of the pair-wise distances distribution
k (for K-Nearest Neighbors Network)	1 to 50

Table 1: *Parameters of our classification model using network community detection and the corresponding range of optimal values searched from.*

4.5 Baseline Models for Comparision

In the pursuit of evaluating the effectiveness of our network community detection-based model, we establish a comparative analysis using two baseline models: K-Nearest Neighbor (KNN) and EfficientNet Classifier.

The K-Nearest Neighbor (KNN) modelis a non-parametric supervised learning algorithm that classifies data points based on the majority vote of its K nearest neighbors. The optimal parameters for the KNN model was search using 5 folds stratified cross validation of the parameters listed in Table 2. For each set of feature vectors the optimal parameters found are illustrated in Table 3.

Parameter	**Range**
feature vectors	raw features / latent featuers
distance metric	cosine / euclidean
k	1 to 50
Vote weight	uniform / weighted according to inverse distance

Table 2: *Parameters of the K nearest neighbors model and the corresponding range of optimal values searched from.*

Feature Vector	K	Distance Metric	Vote weight	Mean Validation Accuracy	Denote as
Raw	6	cosine	weighted	0.925	RawKNN6
Latent	4	cosine	uniform	0.996	LatentKNN4

Table 3: *Optimal K Nearest Neighbors model for each set of features vectors.*

On the other hand, the EfficientNet Classifier, was obtained using Algorithm 1, where we trained the image encoder. By keeping the last classification layer, the model output the a 10-dimension vector which indicates the confidence score of predicting each class. The predicted label is therefore the class with the highest confidence score.

4.6 Tools and Softwares

Refer to Appendix A for the tools and softwares used in this study.

5 Results

5.1 Cross Validation of Community Detection Model

The cross validation results of our community detection model constructed using different factors are presented in Figure 2. The result generally show that our model performs better using latent features and cosine distance to construct the image network. Every models constructed using the latent features showed better mean validation accuracy then those using raw features, suggesting that the learnt latent features are more capable in capturing similiarty between the images. Such similarites can be reflected in the network even keep only the top 5% shortest edges or including edges from the two nearest neighbors only.

The combining effect of using latent features and cosine distance can be understood through Figure 3. The figure shows the distribution of the distance metrics for each type of feature vectors. Using the latent features with cosine distance as the metric, fewer nodes are close to each other, only nodes that possess high resemblance are close to each other.

Table 4 listed the best performing graphs in cross validation for each type of graphs each feature vectors. The network properties of these graphs are presented in Table 5. Note that graph created by the K Nearest Neighbor scheme have significantly less edges. Also, the high modularities of these graph are understood, since only nearest node are linked. It is surprising to observed that the LatentGthres20 graph has such a high value of modularity, as the graph is densely connected with the average node degree around 20% of the total number of nodes. The low modularity of RawGthres95 reflects that the raw feature vectors are not as informative as the latent feature vectors.

Feature Vector	Type of Graph	Threshold	K	Mean Validation Accuracy	Denote as
Raw	Distance Threshold	95^{th} percentile	/	0.868	RawGthres95
Raw	K-Nearest Neighbors	/	5	0.898	RawGknn5
Latent	Distance Threshold	20^{th} percentile	/	0.996	LatentGthres20
Latent	K-Nearest Neighbors	/	16	0.996	LatentGknn16

Table 4: *Graphs with highest mean validation accuracy for each type of graphs and each feature vectors.*

Graph	#Nodes	#Edges	Avg. Degree	Max Degree	Diameter	#Clusters	Modularity
RawGthres95	2500	2967562	2374.05	2499	2	145	0.027238
RawGknn5	2500	9456	7.56	26	14	20	0.829063
LatentGthres20	2500	624750	499.80	944	3	11	0.688238
LatentGknn16	2500	27850	22.28	53	15	16	0.893717

Table 5: *Properties of graphs with highest mean validation accuracy for each type of graphs and each feature vectors.*

5.2 Performance Comparison

Table 6 summarised the accuracy on testing set of our best community detection model and the baseline models. Notice that the supervised KNN algorithm performs much better than our model when using the raw features. With only the raw features, the EfficientNet classifier out performs every other models. However, using the latent features encoded by the EfficientNet, the other models gain significant improments and perform even better than the original EfficientNet classifier. Using the latent features, our models show comparable performance with the supervised KNN model.

Model	**Feature Vectors**	**Accuracy**
RawGthres95	Raw	0.844
RawGknn5	Raw	0.866
RawKNN6	Raw	0.930
EfficientNet Classifier	Raw	0.966
LatentGthres20	Latent	**0.982**
LatentGknn16	Latent	0.974
LatentKNN4	Latent	0.980

Table 6: *Comparison of our best community detection models against the baseline models, in terms of accuracy on the test set.*

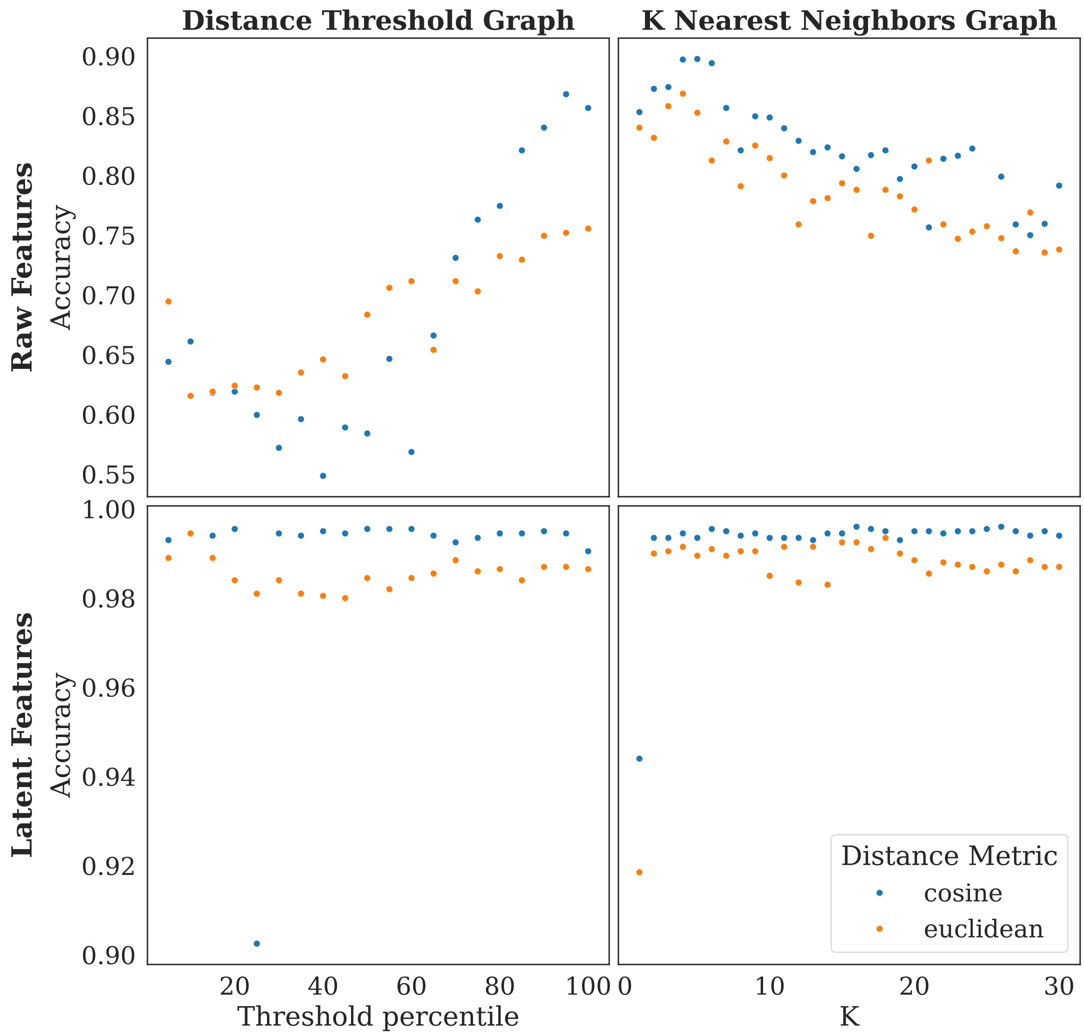

Figure 2: *Cross validation results comparing community detection model performance. The plot displays mean validation accuracy on the Y-axis across different model training approaches. Models bulit using raw features are shown in the first row, and those bulit using latent features are in the second row. The first column corresponds to models where the graph is constructed based on a distance threshold, and the second column represents models with graphs constructed using the K nearest neighbor approach. Within each plot, blue points indicate models constructed using the cosine distance, and orange points represent models constructed using the euclidean distance. This comparative visualization aids in assessing the impact of feature representation and graph construction method on the overall accuracy of the models.*

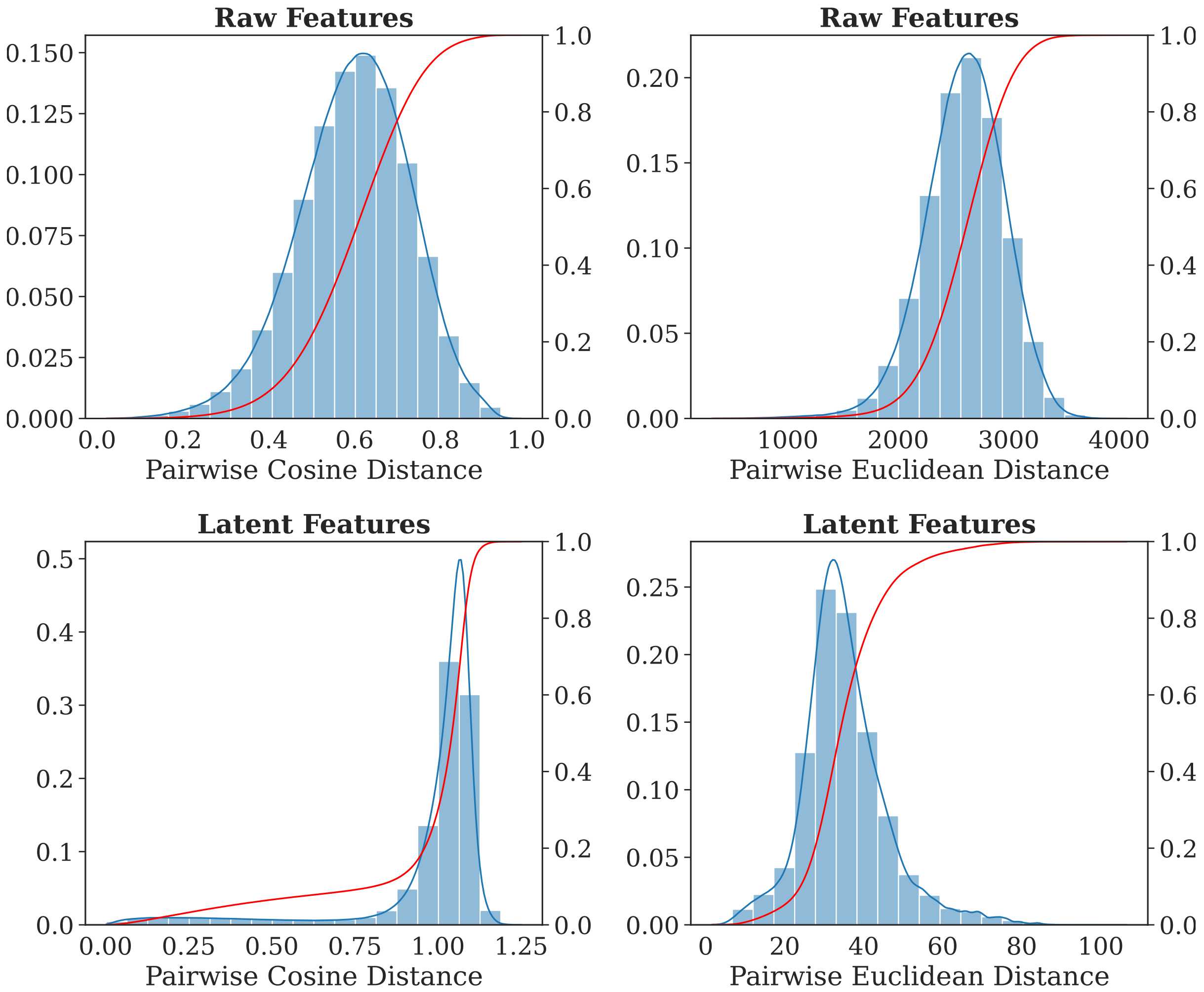

Figure 3: *Distribution of pairwise distances in between different feature vectors. The blue curve represent the empirical probability distribution function while the red curse represents the empirical cumulative distribution function.*

6 Discussion

6.1 Summary

In this study, we explored construction of network graphs using image embeddings to capture community structures in image data. Key findings indicate that such an approach effectively disclose the grouping within the data, potentially better than traditional methods. Specifically, the community structures identified through our network graph approach, when built upon image embeddings, demonstrated promising results, comparable to those achieved by supervised K Nearest Neighbor (KNN) algorithms. This observation is particularly encouraging, as it suggests that unsupervised community detection, combined with sophisticated image embeddings and a majority voting mechanism, can serve as a viable strategy in image classification tasks. The success observed in utilising image embeddings to construct network graphs that accurately reflect community structures. This study's outcomes signal a potential direction for future research and application in integrating community detection with image classification

6.2 Limitations

The study encounters certain limitations that must be acknowledged. Primarily, our limited computational capacity posed a significant constraint on the use of a reduced subset of data for analysis. This limitation might have affected the comprehensiveness of our findings and the generalizability of the model, as a larger and more varied dataset could might yield more insights. Additionally, the choice of the MNIST dataset, is another limitation. These low-resolution and grayscale images of digits, do not capture the complexity present in real-world images. While MNIST serves as a good starting point for understanding the model behavior, it does not fully challenge or reveal the capabilities of the model in situation involving higher-resolution, colored, or more patterned images that are more representative of real-world applications.

6.3 Future Work

Two future research directions include:

- Investigation on the hierarchical structure of the clusters revealed by community detection algorithms. In our study, even for the best model, does not exactly partition the network in to the number of classes. Further exploration on the hierarchical structure might reveal more insights of constructing better network representation of an image dataset.

- Other possible representation of the image network, for example a bipartite graph where all pixels are in one of the group while all images are in another group. Community detection on a bipartites graph might be employed on such network.

References

[1] Rubayyi Alghamdi and Khalid Alfalqi. A survey of topic modeling in text mining. *Int. J. Adv. Comput. Sci. Appl.(IJACSA)*, 6(1), 2015.

[2] Giambattista Amati, Simone Angelini, Antonio Cruciani, Gianmarco Fusco, Giancarlo Gaudino, Daniele Pasquini, and Paola Vocca. Topic modeling by community detection algorithms. In *Proceedings of the 2021 Workshop on Open Challenges in Online Social Networks*, OASIS '21, pages 15–20, New York, NY, USA, 2021. Association for Computing Machinery.

[3] Alejandro Baldominos, Yago Saez, and Pedro Isasi. A survey of handwritten character recognition with mnist and emnist. *Applied Sciences*, 9(15), 2019.

[4] Sylvio Barbon Jr, Gabriel Marques Tavares, and Guilherme Sakaji Kido. Artificial and natural topic detection in online social networks. *iSys - Brazilian Journal of Information Systems*, 10(1):80–98, Mar. 2017.

[5] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, oct 2008.

[6] Leiyu Chen, Shaobo Li, Qiang Bai, Jing Yang, Sanlong Jiang, and Yanming Miao. Review of image classification algorithms based on convolutional neural networks. *Remote Sensing*, 13(22), 2021.

[7] Xueqi Cheng, Xiaohui Yan, Yanyan Lan, and Jiafeng Guo. Btm: Topic modeling over short texts. *IEEE Transactions on Knowledge and Data Engineering*, 26(12):2928–2941, 2014.

[8] Martin Gerlach, Tiago P. Peixoto, and Eduardo G. Altmann. A network approach to topic models. *Science Advances*, 4(7):eaaq1360, 2018.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[10] Afshan Latif, Aqsa Rasheed, Umer Sajid, Jameel Ahmed, Nouman Ali, Naeem Iqbal Ratyal, Bushra Zafar, Saadat Hanif Dar, Muhammad Sajid, and Tehmina Khalil. Content-based image retrieval and feature extraction: A comprehensive review. *Mathematical Problems in Engineering*, 2019:9658350, Aug 2019.

[11] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*, 2013.

[12] Wenchuan Mu, Kwan Hui Lim, Junhua Liu, Shanika Karunasekera, Lucia Falzon, and Aaron Harwood. A clustering-based topic model using word networks and word embeddings. *Journal of Big Data*, 9(1):38, April 2022.

[13] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[14] K. Shuster, S. Humeau, H. Hu, A. Bordes, and J. Weston. Engaging image captioning via personality. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12508–12518, Los Alamitos, CA, USA, jun 2019. IEEE Computer Society.

[15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[16] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. 2019.

[17] Yu Zhang, Peter Tiňo, Aleš Leonardis, and Ke Tang. A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5(5):726–742, 2021.

A Tools and Softwares

The following tools and softwares were used for this study:

Python
Python 3 is the main programming lanuguage used in this study.

Numpy
Numpy is a python package used for implementing fast mathematical operations.

NetworkX
NetworkX is a python package used for implementing network related operation such as graph construction and community detection.

Scikit-learn
Scikit-learn is a python package used for performing cross validation and the supervised K Nearest Neighbors algorithms. Distance metrics computation was also performed using scikit-learn.

Pytorch
Pytorch is deep learning toolkit for python. It is used for getting the MNIST dataset and the pretrained EfficientNet. The retraining flow and image embedding were done using Pytorch.

Seaborn
Seaborn is a python package for flexible data visualisation. It was used to produce plots in this report.