Image Classification Using Graph Autoencoders

PROJECT REPORT
SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENT
FOR THE AWARD OF THE DEGREE
OF
BACHELORS OF TECHNOLOGY
IN
COMPUTER ENGINEERING

Submitted by:

TANISHQ ARORA (2K20/CO/455)
VIVEK RANJAN (2K20/CO/494)
VRUSHANK NANDESHWAR (2K20/CO/495)

UNDER THE SUPERVISION OF
DR ANURAG GOEL

DEPARTMENT OF COMPUTER ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(FORMERLY Delhi College of Engineering)
Bawana Road, Delhi-110042

DECEMBER 2023

CANDIDATE'S DECLARATION

We, Tanishq Arora (2K20/CO/455), Vivek Ranjan (2K20/CO/494) and Vrushank Nandeshwar (2K20/CO/495), students pursuing Bachelors of Technology in Computer Engineering, hereby, declare that our project 'Image Classification Using Graph Autoencoders', submitted to the Department of Computer Engineering, Delhi Technological University, Delhi for the fulfilment of the requirement for being awarded the degree of Bachelors of Technology in Computer Engineering is original and not plagiarised from any source, and that we have cited all the references. This work has not earlier formed the foundation for awarding any Degree or Diploma or similar title.

TANISHQ ARORA	**VIVEK RANJAN**	**VRUSHANK NANDESHWAR**
(2K20/CO/455)	**(2K20/CO/494)**	**(2K20/CO/495)**

Place: Delhi

Date: 15 December 2023

CERTIFICATE

We hereby certify that the project report titled ‘Image Classification Using Graph Autoencoders’ submitted by Tanishq Arora (2K20/CO/455), Vivek Ranjan (2K20/CO/494) and Vrushank Nandeshwar (2K20/CO/495), pursuing Bachelors of Technology in Computer Engineering from Delhi Technological University, Delhi, in partial fulfilment of needed requirements for awarding of Bachelors of Technology in Computer Engineering, is a record of project work carried out by the students under my supervision. To the best of our knowledge, the work has not been submitted in part or full for any other Degree or Diploma to this University elsewhere.

Dr Anurag Goel
SUPERVISOR

Place: Delhi
Date: 12th December 2023

ABSTRACT

This paper discusses the prevalent methods for image classification as well as the use of graph autoencoders as a means of image classification. Scene understanding contains the task of systematically converting the image to a graph representation of the image representing the objects as nodes and edges as the relationships between them. The proposed model uses the lower-dimension embedding obtained from the graph autoencoder for classification. The performance of existing image classification methods like k-NN, Support Vector Machine (SVM), Decision Trees, Artificial Neural Networks were compared with each other and with the performance with Graph Autoencoders.

Each classification technique was implemented on three datasets to remove dataset-based bias in the results for any technique. The results are compared on the basis of their accuracy, precision, recall and f1 score. These are predefined metrics to create an empirical study on the performance of each method.

ACKNOWLEDGEMENT

We are grateful to Prof. Vinod Kumar (Head of Department, Department of Computer Engineering), Dr. Anurag Goel (Assistant Professor, Department of Computer Engineering), and all of the faculty members of the Department of Computer Engineering, Delhi Technological University for their tremendous support and guidance in completing the project that we undertook. This effort was made possible thanks to their guidance. We would also like to express our deepest gratitude to all of the faculty members of the Department of Computer Engineering, Delhi Technological University for their unwavering support and important insights.

We are also appreciative to the university for providing us with well-equipped laboratories, cutting-edge infrastructure, and great testing facilities. These materials were critical in aiding our study and experimentation, helping us to complete our tasks effortlessly and efficiently. We would also like to thank our lab assistants, seniors, and peer group for their constant support and assistance. Their willingness to share their expertise and experiences has been extremely valuable. Finally, we would like to thank everyone who has helped directly or indirectly with the successful completion of our thesis. Their encouragement, support, and advice have been essential, and we are eternally thankful.

CONTENTS

Image Classification Using Graph Autoencoders **1**
CANDIDATE'S DECLARATION 2
CERTIFICATE 3
ABSTRACT 4
ACKNOWLEDGEMENT 5
CONTENTS 6
LIST OF FIGURES 8
LIST OF TABLES 9
LIST OF SYMBOLS, ABBREVIATIONS AND NOMENCLATURE 10
CHAPTER 1 11
INTRODUCTION 11
1.1 Overview 11
1.2 Motivation 11
1.3 Problem Statement 11
CHAPTER 2 12
DATASETS 12
2.1 Datasets 12
2.1.1 CIFAR10 12
2.1.2 MNIST 13
2.1.3 Fashion MNIST 13
CHAPTER 3 14
DATA PRE-PROCESSING 14
3.1 Data Loading and Grayscale Conversion 14
3.2 Feature Extraction 14
3.2.1 Histogram of Oriented Gradients 14
3.2.2 Graph Autoencoder Embedding to Latent Vector 15
3.3 Image to Graph Representation 15
CHAPTER 4 17
CLASSIFICATION USING EXISTING METHODS 17
4.1 Supervised Image Classification 17
4.1.1 Decision Trees 17
4.1.2 k-Nearest Neighbour 18
4.1.2 Support Vector Machine 18
4.1.4 Convolution Neural Network 19

CHAPTER 5 20
CLASSIFICATION USING GRAPH AUTOENCODERS 20
5.1 Graph Representation 20
5.2 Graph Autoencoders 20
5.2.1 Autoencoders 20
5.2.2 Embedding 21
5.2.3 Loss Function 21
CHAPTER 6 22
CONCLUSION AND FUTURE SCOPE 22
6.1 Performance Metrics 22
6.1.1 Accuracy 22
6.1.2 Precision 22
6.1.3 Recall 22
6.1.4 f1 Score 22
6.2 Results 23
6.2.1 Decision Tree 23
6.2.2 k-Nearest Neighbours 23
6.2.3 Support Vector Machine 23
6.2.4 Convolutional Neural Network 24
6.3 Conclusion 24
6.4 Future Scope 25
6.4.1 Contrastive Learning 25
REFERENCES 26

LIST OF FIGURES

Fig 2.1 Sample of CIFAR10 dataset 13
Fig 2.2 Sample of MNIST dataset 13
Fig 2.3 Sample of fashion MNIST dataset 13
Fig 3.1 Image representation of output from HOGDescriptor 15
Fig 3.2 Sample views of graphs generated from images 17
Fig 4.1 Sample Image Classification Decision Tree 18
Fig 4.2 Mathematical formula for calculating distance 19
Fig 4.3 Decision boundaries in Support Vector Machines 20
Fig 4.4 Convolutional Neural Network layers 20
Fig 5.1 Standard Autoencoder flow 22
Fig 6.1 Result generated by Convolutional Neural Network using the MNIST dataset 25

LIST OF TABLES

Table 6.1 Decision Tree Classification Result 24
Table 6.2 k-Nearest Neighbours Classification Result 24
Table 6.3 Support Vector Machine Classification Result 24
Table 6.4 Convolutional Neural Network Classification Result 25

LIST OF SYMBOLS, ABBREVIATIONS AND NOMENCLATURE

MNIST	Modified National Institute of Standards and Technology database
OpenCV	Python library for computer vision
COLOR_BGR2GRAY	Function from OpenCV library to convert images to grayscale
HOG	Histogram of Oriented Gradients
KNN	k-Nearest Neighbours
SVM	Support Vector Machine
CNN	Convolutional Neural Network
Embedding/Latent Space	The output of encoder from autoencoder

CHAPTER 1

INTRODUCTION

1.1 Overview

The level of automation in the modern world could not be brought about without the help of computer vision. One of the main focuses of computer vision is Image Classification. The primary objective of this project was to gauge the state of image classification currently and develop a new method of image classification using graph autoencoders to compare its performance with already existing techniques.

Objectives -

- To research and implement existing image classification techniques
- To compare their performance with each other
- To implement image classification using graph autoencoders
- To compare its performance to previous techniques

1.2 Motivation

Image classification is an integral part of computer vision, with applications ranging from using your phone to identify an object to self-driving vehicle technology. With the rise in automation requirements, image classification will play an important role in allowing for rapid growth.
Graph autoencoders could bring more information from the image in a lower dimension, making image classification more efficient and even more accurate.

1.3 Problem Statement

To review existing methods for image classification using common metrics and to develop a novel classification method using graph autoencoders. The new autoencoder should be compared to previous methods using the metrics for comparison.

CHAPTER 2

DATASETS

2.1 Datasets

The initial concern for any machine learning endeavour is being able to find large amounts of high-quality data. In image classification tasks, the data in question is the set of images on which to train the classification models. Since the problem of image classification is an old one, there are many datasets freely available for public use.

In the dataset selection process, the main concerns were the variety of the images and the size of the dataset. Both criteria needed to be within the bounds of our computational capabilities. Images near the size of 32 pixels in height and width were preferred for this specific application.

2.1.1 CIFAR10

This is a collection of 60,000 labelled colour images of size 32x32 pixels. The images are divided into 10 classes of everyday objects.

This dataset was compiled and is maintained by Alex Krizhevsky et al., The University of Toronto [1].

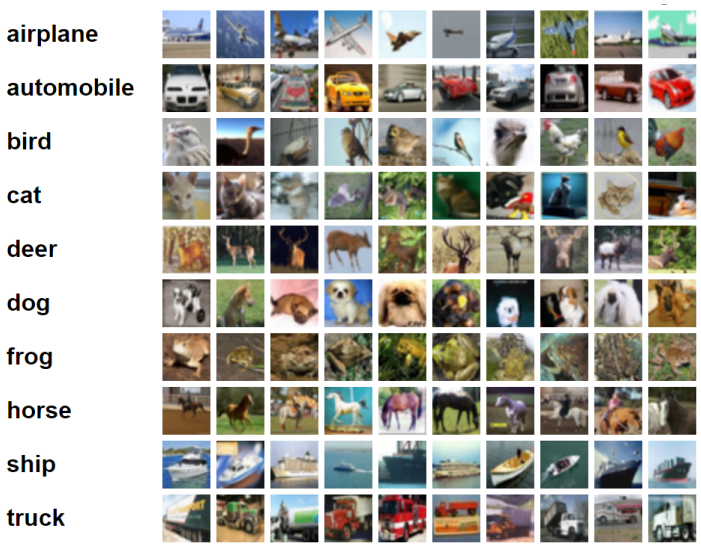

Fig 2.1 Sample of CIFAR10 dataset

2.1.2 MNIST

A collection of 70,000 labelled grayscale images of size 28x28 pixels. Images are of handwritten digits (0-9) and are divided into 10 classes [2].

This dataset was created and is maintained by Yann LeCun from Courant Institute with her associates.

Fig 2.2 Sample of MNIST dataset

2.1.3 Fashion MNIST

A collection of 70,000 labelled grayscale images of size 28x28 pixels [3]. Images are of common fashion items and are divided into 10 classes of wearable items. This dataset was created by Xiao, Han 2017.

Fig 2.3 Sample of fashion MNIST dataset

CHAPTER 3

DATA PRE-PROCESSING

3.1 Data Loading and Grayscale Conversion

Data is loaded from the Keras library in Python. The load_data() function returns a tuple of tuples containing the train test splits of the images and their labels.

The CIFAR10 dataset used contains colour images while both MNIST datasets contain grayscale images. In order to reduce computation time and complexity of the models used, CIFAR10 images were converted to grayscale. This was done using the COLOR_BGR2GRAY() function from the OpenCV library in Python. Further every model requires the size of the images which can be extracted from individual images from the train/test split data. These images may need to be resized in order to bring them to a workable dimension, but in this case, they are already of a manageable size.

3.2 Feature Extraction

In the task of image classification, the models used depend on their similarity to other images in the same class as well as their difference from images in different classes. Grayscale images are represented by a matrix of values representing pixel blackness levels (0 black and 255 white). This leads to the problem of being able to identify patterns or features to classify on.

The purpose of using graph autoencoders is to be able to train a graph encoder to lower the dimensionality and in turn actually extract the relevant and contributing features from the image and therefore lead to a better classification model.

3.2.1 Histogram of Oriented Gradients

HOG is a method to extract features from an image. It uses the count of gradient orientations in local groups in order to extract features in the form of a vector descriptor from the image [4]. The HOGDescriptor() function was used from the OpenCV library.

HOG gives the sharp changes in the gradient in the image, effectively operating as an edge detector. It combines the values of the Gradient matrix and the Orientation matrix. The result

obtained from a HOGDescriptor(), is an array of floating point numbers which represent the image's local gradient information to be used as the feature vector.

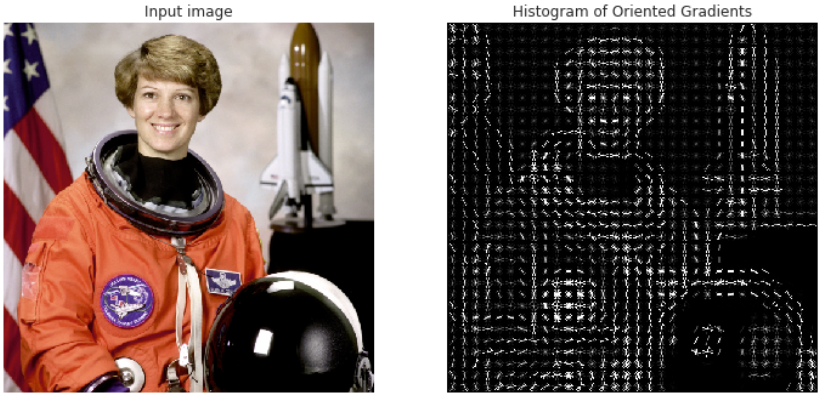

Fig 3.1 Image representation of output from HOGDescriptor

3.2.2 Graph Autoencoder Embedding to Latent Vector

The graph autoencoder takes a graphical representation of an image which it uses as the input layer for a neural network which internally hides the 2D convolutional layers. The output from this first neural network is called the embedding or the Latent Space [5].

This latent vector representation can be used as the feature for the classification using neural networks. The training of the latent vector representation to be accurate, meaningful and representative of the actual image is done by reconstructing the latent vector representation to the full image using a decoder neural network.

3.3 Image to Graph Representation

The main challenge in applying graph autoencoders to images, is being able to create a graph from the image. This graph must contain semantic information regarding the image and the positioning of objects inside the image. Neural Networks, a sort of deep learning method, can be used for this process [6]. This doesn't eliminate the option of using HOGDescriptor or other feature extraction techniques. Once the features have been extracted using a HOGDescriptor, these features can be converted to a regular graph using the networkx library The tradeoff between using a neural network for graph generation and using a HOGDescriptor like feature extraction technique is the computational complexity to the resulting extracted feature graph.

The deep learning technique would provide superior semantic information extracted and the regular computer vision techniques would provide for weak semantic representation.

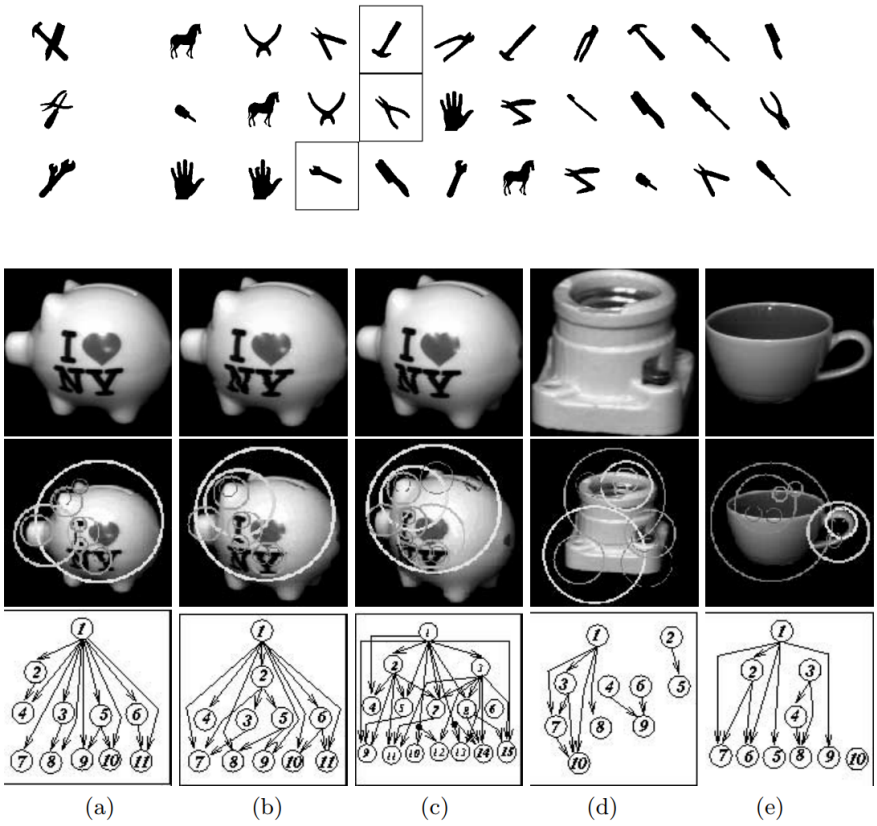

Fig 3.2 Sample views of graphs generated from images [14]

CHAPTER 4

CLASSIFICATION USING EXISTING METHODS

Some traditional image classification techniques have been discussed below. The results of these techniques can then be used to come up with a Graph Autoencoder model.

4.1 Supervised Image Classification

These techniques use labelled data to train the classifier that can predict the class of new images.

4.1.1 Decision Trees

Decision trees contain decision nodes. Each node is a sort of filter applied to the image. In a binary decision tree, the node identifies boolean values of its decision factor and then classifies the image in question into a certain group of images only based on that decision factor. Combining all the features of a class unique image, creates a tree of decisions that can be traversed through by selecting the subtree which corresponds to the decision value of the current image in the current decision parameter.

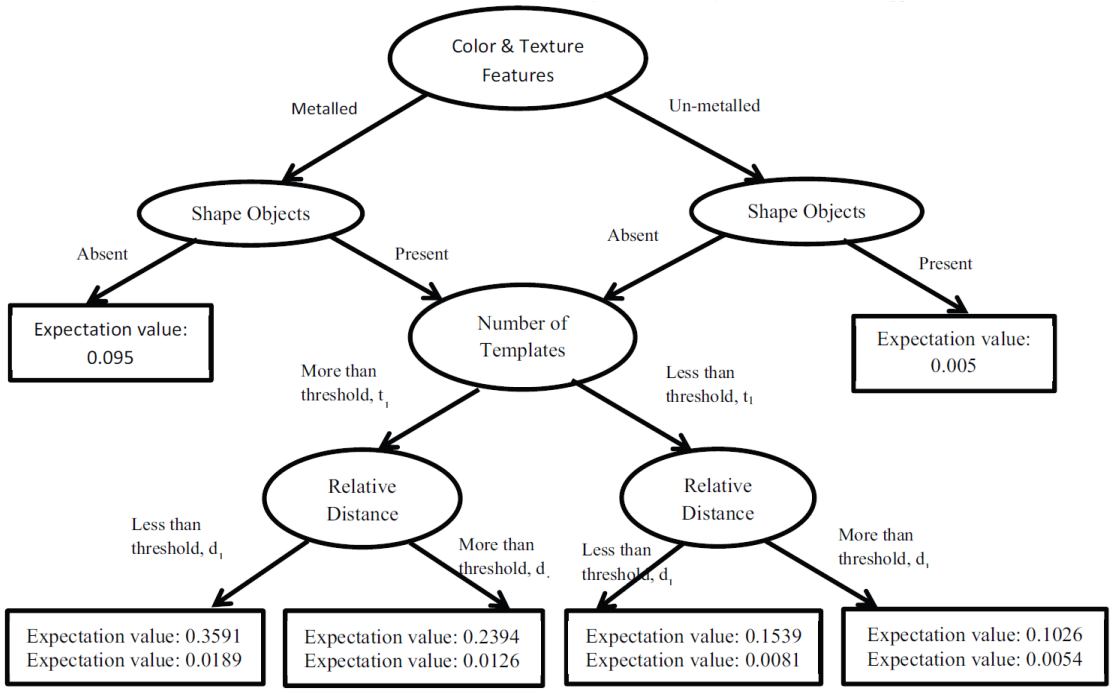

Fig 4.1 Sample Image Classification Decision Tree [7]

4.1.2 k-Nearest Neighbour

k-nearest neighbour (k-NN) is an intuitive method for image classification. It works by finding the k closest images in the training dataset to a given query image, based on a distance metric such as Euclidean distance or cosine similarity. The query image is then assigned the label that is the majority classification among its k nearest neighbours [8]. The Euclidean distance for n-dimensional data is given as:

$$\text{Euclidean distance} = \sqrt{\sum_{i=1}^{n} \left(x_i - y_i\right)^2}$$

Fig 4.2 Mathematical formula for calculating distance

In the context of an image, once the features are extracted into a feature vector, the distance is calculated on those feature values.

4.1.2 Support Vector Machine

SVM is a supervised algorithm that works by finding the optimal hyperplane that separates data pints of different classes. SVMs use the data points that are closest to the hyperplane and are called as support vectors. The hyperplane is chosen so that the distance between the closest points of different classes is maximised, also known as the margin of the model. SVM can also handle high-dimensional and nonlinear data, and use different kernel functions to transform the data into a suitable space.

Some of the Kernels are: Linear, RBF(Radial Basis Function), and Polynomial Kernel

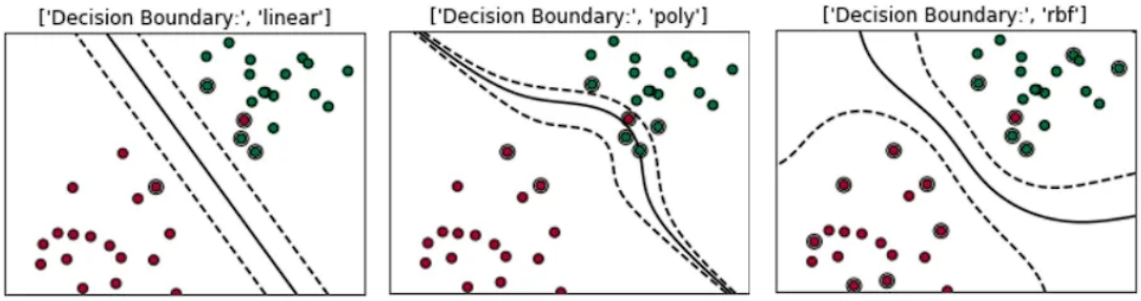

Fig 4.3 Decision boundaries in Support Vector Machines

For image classification, SVM uses the feature vector to calculate the optimal hyperlane for classifying the images into the classes.

4.1.4 Convolution Neural Network

A Convolutional Neural Network, which is a type of neural network that can process images and extract features from them. CNNs are widely used for image classification [10], due to their ability to work on matrices (representations of images). CNNs learn hierarchical features from images, such as edges, shapes, textures, and objects, by applying convolutional filters, pooling layers and fully connected layers with different or same activation functions to perform classification. An activation function is simply a function that takes a floating point number input in the range of 0-1 and outputs a binary value based on its distribution. An example of such a function would be the sigmoid function.

With the help of CNNs the dimensions of the image are reduced as after each iteration a new matrix like structure is generated which consists of the features extracted from the input in the current iteration hence making the overall classification model more efficient. CNNs can classify images with high accuracy and efficiency and can handle complex and diverse data.

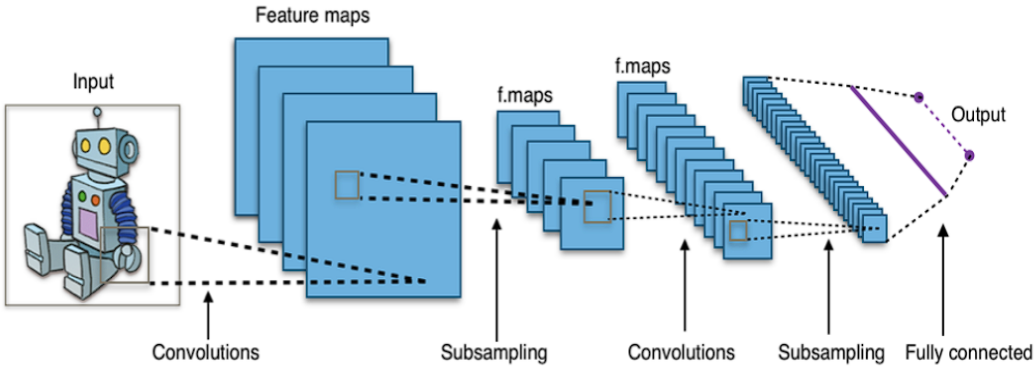

Fig 4.4 Convolutional Neural Network layers

CHAPTER 5

CLASSIFICATION USING GRAPH AUTOENCODERS

5.1 Graph Representation

Graphs are an effective way of representing data that consists of entities and their relationships. The graph data structure has been applied to many complex and heterogeneous datasets such as social networks, citation networks, knowledge bases, etc. Graph Autoencoders are a type of neural network model that can learn to encode and decode graph-structured data. Graph autoencoders can be used for image classification by extracting features from the images, converting them into a graph-based structure and using them as an input for a classifier. Graph autoencoders can learn a low-dimensional and robust representation of the images, which can reduce the dimensionality and noise of the data and improve the performance of the classifier.

5.2 Graph Autoencoders

The beauty of using graph autoencoders for image classification is that the latent vector representation, which is an intermediate data state in the graph autoencoder process, is just an example of a feature extraction technique. This allows for its use to train models with high accuracy.

To understand the working of graph autoencoders, an understanding of regular autoencoders is necessary.

5.2.1 Autoencoders

The traditional autoencoder, as depicted below, is a type of neural network that comprises two main components: an encoder and a decoder. The encoder's role is to take a data point, denoted as X, and transform it into a lower-dimensional representation, referred to as the embedding Z. The decoder then takes this embedding Z and attempts to reconstruct the original input, producing an output referred to as $\widehat{X}$. The resemblance between the input X and the output $\widehat{X}$ is

largely dependent on the quality of the embedding. The accuracy of an autoencoder is measured by the similarity between the data points X and $\widehat{X}$:

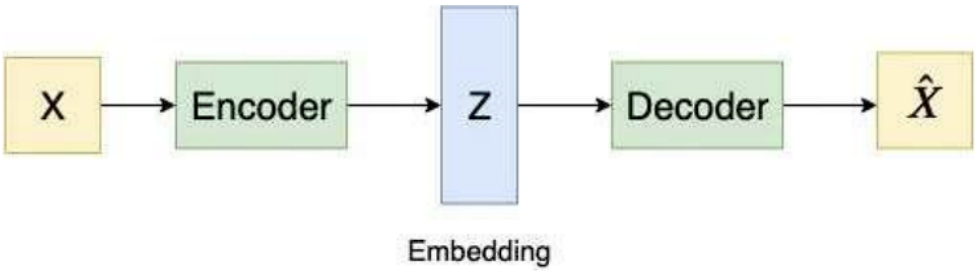

Fig 5.1 Standard Autoencoder flow

5.2.2 Embedding

How we can use the lower dimension embedding is very interesting as there can be several use cases. The most straightforward use case would be space optimization as storing lower dimension embeddings would use less space compared to storing their pixel intensities. It could also save computational power as the input dimensions are lower. Hence using the embeddings as an input for the final image classifier would make the overall model efficient in terms of time and space, and less computationally complex as well.

5.2.3 Loss Function

The loss function is a measure of the autoencoder performance i.e. how well it can reconstruct the input data from the latent representation. The loss function is usually composed of two terms: a reconstruction loss and a regularisation loss.

The reconstruction loss is the combined difference between the input to the encoder and the reconstructed output from the decoder. The regularisation loss on the other hand imposes some constraints on the latent representation, such as sparsity, smoothness etc., and penalises the deviation between the two.

The loss function is used to train the autoencoder by creating a minimization term loss. With each iteration of training the autoencoder, the loss function is minimised to give the best overall result, and hence it affects the quality and properties of the latent representation and the reconstruction.

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 Performance Metrics

Any model trained for a classification task needs to be empirically measured in order to compare its performance to other similar models. The performance of a model depends on the type of dataset used and the viability of the model in the context of that data, the amount of data available and the tuning of the parameters in the model.

To measure the performance, predefined metrics need to be established and used systematically and uniformly across the different results obtained from the models trained.

6.1.1 Accuracy

Mathematically this is defined as the ratio of the number of data points correctly classified to the total number of data points classified. The accuracy is computed for each of the classes separately and then these values are combined to form an average accuracy.

6.1.2 Precision

The ratio of the number of correctly classified data points for a class to the total number of data points classified to a particular class. This has to be calculated for every class separately and then combined.

6.1.3 Recall

Similar to the precision with a key difference, that recall is the ratio of the number of correctly classified data points for a class to the total number of actual data points in the class. In precision, the comparison is to the number of data points classified to a certain class, in recall, it is the number of data points actually in the class.

6.1.4 f1 Score

It is the harmonic mean of recall and precision. It measures the combined effect of both similar metrics

6.2 Results

The three datasets used were loaded and then each of the models were trained using the corresponding test data. The resulting model was tested using the labelled values from the dataset which were not used for training (test set) and the results were expressed as tables of different performance metrics previously discussed.

6.2.1 Decision Tree

Dataset	**Average Accuracy**
cifar10	31%
MNIST	91%
Fashion MNIST	76%

Table 6.1 Decision Tree Classification Result

6.2.2 k-Nearest Neighbours

Dataset	**Average Accuracy**
cifar10	54%
MNIST	99%
Fashion MNIST	87%

Table 6.2 k-Nearest Neighbours Classification Result

6.2.3 Support Vector Machine

Dataset	**Average Accuracy**
cifar10	66%
MNIST	99%
Fashion MNIST	90%

Table 6.2 Support Vector Machine Classification Result

6.2.4 Convolutional Neural Network

Dataset	Average Accuracy
cifar10	55%
MNIST	98%
Fashion MNIST	88%

Table 6.4 Convolutional Neural Network Classification Result

	precision	recall	f1-score	support
0	0.97	1.00	0.98	980
1	0.99	1.00	0.99	1135
2	0.99	0.98	0.98	1032
3	0.98	0.99	0.98	1010
4	0.99	0.98	0.98	982
5	0.99	0.98	0.98	892
6	0.99	0.99	0.99	958
7	0.98	0.98	0.98	1028
8	0.99	0.97	0.98	974
9	0.97	0.98	0.97	1009
accuracy			0.98	10000
macro avg	0.98	0.98	0.98	10000
weighted avg	0.98	0.98	0.98	10000

Fig 6.1 Sample result generated by Convolutional Neural Network using the MNIST dataset

6.3 Conclusion

One of the first observed effects were of the parameters of Histogram of Oriented Gradients. Any slight variation in these parameters resulted in an extreme change in the resulting classification accuracy.

The MNIST dataset was not a very challenging dataset in terms of accurate predictability. This led to the accuracy of its classification being greater than 90% in all of the models. The cifar10 had a more varied set of images leading to a much lower success rate in classification.

The use of graph autoencoders for image classification would require highly accurate graph representation of images. These representations can be generated by using any of these image classification techniques as object recognition algorithms. Further the graph can be connected using edges in order to form the structure of the image in a graphical form,

6.4 Future Scope

The graph autoencoder with image-to-graph conversion is to be fully implemented. The datasets being large in terms of the number of images and their resolution and the task of converting an image to a graph is a very computationally intensive one and has led to difficulties and delays in this process.

6.4.1 Contrastive Learning

In the current use of the graph autoencoders, the latent vector representation of the image is being used as a lower dimension and compressed representation of the information contained in the image. To make full use of the graph autoencoder, the reconstructed image can also be used. The image will be considered as a positive pair for contrastive learning.

Contrastive learning is a classification technique that uses a set of n pairs of images out of which one is a positive pair (pair of images from the same class) and n-1 are the negative pairs (pairs of images from different classes) to classify incoming images. This is a type of self-supervised technique for image classification. It uses the contrastive loss function which represents how similar the pairs are.

REFERENCES

[1] A. Krizhevsky, “Learning Multiple Layers of Features from Tiny Images,” M.Sc Thesis, University of Toronto, 2009.

[2] LeCun, Y. (1998). The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/.

[3] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms,” 2017.

[4] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 2005, pp. 886-893 vol. 1, doi: 10.1109/CVPR.2005.177.

[5] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, “Adversarially Regularized Graph Autoencoder for Graph Embedding,” 2018.

[6] Newell, Alejandro, and Jia Deng. Pixels to Graphs by Associative Embedding. 2017.

[7] C. Agarwal and A. Sharma, "Image understanding using decision tree based machine learning," ICIMU 2011 : Proceedings of the 5th international Conference on Information Technology & Multimedia, Kuala Lumpur, Malaysia, 2011, pp. 1-8, doi: 10.1109/ICIMU.2011.6122757.

[8] Amato, Giuseppe, and Fabrizio Falchi. "kNN based image classification relying on local feature similarity." In Proceedings of the Third International Conference on Similarity Search and Applications, pp. 101-108. 2010.

[9] Sun, Xiaowu, Lizhen Liu, Hanshi Wang, Wei Song, and Jingli Lu. "Image classification via support vector machine." In 2015 4th International Conference on Computer Science and Network Technology (ICCSNT), vol. 1, pp. 485-489. IEEE, 2015.

[10] Chen, Leiyu, Shaobo Li, Qiang Bai, Jing Yang, Sanlong Jiang, and Yanming Miao. "Review of image classification algorithms based on convolutional neural networks." Remote Sensing 13, no. 22 (2021): 4712.

[11] Fan, Shaohua, Xiao Wang, Chuan Shi, Emiao Lu, Ken Lin, and Bai Wang. "One2multi graph autoencoder for multi-view graph clustering." In proceedings of the web conference 2020, pp. 3070-3076. 2020.

[12] Hou, Zhenyu, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. "Graphmae: Self-supervised masked graph autoencoders." In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 594-604. 2022.

[13] Lu, Chenyang, and Gijs Dubbelman. "Image-Graph-Image Translation via Auto-Encoding." arXiv preprint arXiv:2012.05975 (2020).

[14] Shokoufandeh, Ali, and Sven Dickinson. "Graph-theoretical methods in computer vision." In Summer School on Theoretical Aspects of Computer Science, pp. 148-174. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000.