

MESH SEGMENTATION FROM SPARSE FACE LABELS USING GRAPH
CONVOLUTIONAL NEURAL NETWORKS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ÖNDER İLKE SEVER

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

JANUARY 2020

Approval of the thesis:

**MESH SEGMENTATION FROM SPARSE FACE LABELS USING GRAPH
CONVOLUTIONAL NEURAL NETWORKS**

submitted by **ÖNDER İLKE SEVER** in partial fulfillment of the requirements for
the degree of **Master of Science in Computer Engineering Department, Middle
East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering**

Assoc. Prof. Dr. Yusuf Sahillioğlu
Supervisor, **Computer Engineering, METU**

Assoc. Prof. Dr. Sinan Kalkan
Co-supervisor, **Computer Engineering, METU**

Examining Committee Members:

Assist. Prof. Dr. Emre Akbaş
Computer Engineering, METU

Assoc. Prof. Dr. Yusuf Sahillioğlu
Computer Engineering, METU

Assist. Prof. Dr. Ufuk Çelikcan
Computer Engineering, Hacettepe University

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Önder İlke SEVER

Signature :

ABSTRACT

MESH SEGMENTATION FROM SPARSE FACE LABELS USING GRAPH CONVOLUTIONAL NEURAL NETWORKS

SEVER, Önder İlke

M.S., Department of Computer Engineering

Supervisor: Assoc. Prof. Dr. Yusuf Sahillioğlu

Co-Supervisor : Assoc. Prof. Dr. Sinan Kalkan

January 2020, 48 pages

The marked improvements in deep learning influence almost every area of computer science. The mesh segmentation problem in computer graphics has been an active research area and keep abreast of the trend of deep learning developments.

The mesh segmentation has a central role in multiple application areas for 3D objects. It is chiefly used to produce the object structure in order to manipulate the object or analyze the components of it. These operations are primitive, and that primitiveness causes a variety of application areas. The variation in application areas induce a variety of priority deviations over time and memory usage.

In this thesis, we solve the mesh segmentation problem by using Graph Convolutional Neural Networks. Our method uses a semi-supervised approach for which the mesh objects are sparsely labeled, and the results are the formed segments. We consider a mesh object as a graph by using their connectedness over the faces, and having the mesh in 3D lets us create geometrically logical features for our network. Using the neighborhood information is maintained by the Graph Convolutional Neural

Networks, which is a pretty new concept, and the application on the sparsely labeled mesh segmentation is novel to our work. By using the briefly summarized method, we reach competitive results compared to state-of-art mesh segmentation methods.

Keywords: 3D, mesh, segmentation, semi-supervised learning, graph convolutional neural networks

ÖZ

GRAFİKSEL EVRİŞİMLİ SINIR AĞLARINI KULLANARAK YÜZEYLERİ SEYREK ETİKETLENMİŞ NESNE BÖLÜTLEMESİ

SEVER, Önder İlke

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Doç. Dr. Yusuf Sahillioğlu

Ortak Tez Yöneticisi : Doç. Dr. Sinan Kalkan

Ocak 2020 , 48 sayfa

Derin öğrenmedeki belirgin gelişmeler Bilgisayar Bilimlerinin neredeyse her alanını etkilemektedir. Bilgisayar grafiklerinde nesne bölütlemesi aktif bir araştırma alanı olmuştur ve derin öğrenme gelişmeleri trendini yakından takip etmektedir.

Nesne bölütlemesi, 3 boyutlu nesnelere ile ilgili birçok uygulama alanında merkezi bir role sahiptir. Nesneyi manipüle etmek veya bileşenlerini analiz etmek için esas olarak nesne yapısını üretmek için kullanılır. Bu işlem çok temel bir role sahiptir ve bu temel olma özelliği uygulama alanlarına çeşitliliğe neden olur. Uygulama alanlarındaki çeşitlilik, zaman ve bellek kullanımı üzerinde öncelik sapmalarına yol açmaktadır.

Bu tezde, Grafikselle Evrişimli Sinir Ağlarını kullanarak Nesne Bölütlemesi problemini çözüyoruz. Metodumuz, bölgesel olarak bölütleri etiketlenmiş objeleri alarak, nesnenin bölütlenmiş halini sonuç olarak üreten yarı denetimli bir yöntemdir. Bölütlenecek nesnenin ağ bilgisini kullanarak, yüzeylerin bağlantıları ile bir grafik olarak düşünüyoruz, ağın 3 boyutlu düzlemde olması sinir ağımız için geometrik olarak mantıksal özellikler oluşturmamızı sağlıyor. Komşuluk bilgilerinin kullanılması,

oldukça yeni bir kavram olan Grafiksel Evrişimli Sinir Ağları tarafından korunmakta olup, bu yeni sinir ağlarının kullanılarak nesne bölütlemesi işleminin uygulanması; ilk kez bu çalışmada bizim tarafımızdan gerçekleştirilmiştir. Kısaca özetlenen yöntemi kullanarak, alanlarında en iyi kabul edilen nesne bölütlemesi yöntemleriyle kıyaslanabilecek sonuçlara ulaşıyoruz.

Anahtar Kelimeler: 3B, nesne, bölütleme, yarı gözetimli öğrenme, grafiksel evrişimli sinir ağları

Dedicated to you, Mom.

ACKNOWLEDGMENTS

Firstly, I would like to thank my thesis advisors Assoc. Prof. Dr. Yusuf Sahilliođlu and co-advisor Assoc. Prof. Dr. Sinan Kalkan for their surveillance, guidance, and continuous support.

I would like to thank my family for enabling me to achieve every single success of mine by creating opportunities that they never had. I appreciate your sacrifices every single day.

I would like to thank my warm-hearted sister Pınar Simge Sever, my upcoming brother-in-law Utkucan K ulekçi, and our wild but adorable cat Aşkuş for their emotional support and endless tolerance while sharing the home with me throughout my thesis process.

I would like to thank my beloved friend Beng ucan Kasaplı for standing with me whenever I felt alone or stuck about anything during my master’s degree program. I am grateful for my luck.

I would like to thank my dear friend and colleague Emir Kaan Perek for his sincere helps through my thesis process from a to izzard.

I would like to thank the family of Ankawiser; Erkut Alakuş, Emir Kaan Perek, and Onur Tosun for allowing me to proceed with my education and tolerating me during this period.

I would like to thank the House Bulbasaur members Baran Kemer and Batuhan Akbaş, for the motivational events and their kind friendship whenever I needed them through my master’s degree program.

Finally, this thesis work is supported by the Scientific and Technological Research Council of Turkey (T UB ITAK) under the project EEEAG-215E255.

TABLE OF CONTENTS

ABSTRACT	v
ÖZ	vii
ACKNOWLEDGMENTS	x
TABLE OF CONTENTS	xi
LIST OF TABLES	xiii
LIST OF FIGURES	xv
LIST OF ALGORITHMS	xvii
LIST OF ABBREVIATIONS	xviii
CHAPTERS	
1 INTRODUCTION	1
1.1 Problem Definition	2
1.2 Motivation	2
1.3 Method Overview and Contributions	3
1.4 The Outline of the Thesis	4
2 BACKGROUND AND RELATED WORK	5
2.1 Artificial Neural Networks	5
2.2 Mesh Segmentation	8

3	SEGMENTATION MODEL	11
3.1	Pre-Processing Stage	11
3.1.1	Generating Labeled Nodes	11
3.1.2	Generating Features	12
3.2	Model Architecture	12
3.2.1	Input and Output	12
3.2.2	Loss Functions	13
3.2.3	Graph Convolutional Layers	14
3.3	Sparsely Labeled Training Stage	15
4	EXPERIMENTS AND RESULTS	19
4.1	Dataset	19
4.2	Experiments	20
4.2.1	Labeled Face Ratio	20
4.2.2	Network Topology	24
4.2.3	Adjacency Impact on the Loss function	28
4.2.4	Activation Function	32
4.2.5	Dropout	34
4.2.6	Batch Size	36
4.3	Evaluation	39
5	CONCLUSIONS AND DISCUSSION	41
5.1	Future Work	42
	REFERENCES	43

LIST OF TABLES

TABLES

Table 4.1	Architectural details of the changing labeled face ratio experiment.	21
Table 4.2	Best accuracy values for different label ratios.	22
Table 4.3	Visual results of Labeled Face Ratio experiment.	23
Table 4.4	Different network topologies for the experiment.	24
Table 4.5	Parameters for network topology experiment.	24
Table 4.6	Best accuracy values for different network topologies.	25
Table 4.7	Visual results of Network Topology experiment (1).	27
Table 4.8	Visual results of Network Topology experiment(2).	28
Table 4.9	Architectural details of the changing adjacency impact on the loss function experiment.	28
Table 4.10	Best accuracy values for different adjacency impacts (1).	29
Table 4.11	Best accuracy values for different adjacency impacts (2).	30
Table 4.12	Visual results of Adjacency Impact on the Loss function experiment (1).	31
Table 4.13	Visual results of Adjacency Impact on the Loss function experiment (2).	32
Table 4.14	Architectural details of the different activation functions experi- ment.	32

Table 4.15 Best accuracy values for different activation functions.	33
Table 4.16 Visual results of Activation Function experiment.	34
Table 4.17 Architectural details of the changing dropout ratio experiment. . . .	35
Table 4.18 Best accuracy values for different dropout values.	36
Table 4.19 Visual results of Dropout experiment.	36
Table 4.20 Architectural details of the changing batch size experiment.	37
Table 4.21 Best accuracy values for different batch sizes.	38
Table 4.22 Visual results of Batch Size experiment.	39
Table 4.23 Accuracy comparison of our method for human category of 3D shapes in Princeton Segmentation Benchmark with different labeled face ratios.	40
Table 4.24 The Rand Index scores of segmentation for the human category of 3D shapes in Princeton Segmentation Benchmark with different methods. . .	40

LIST OF FIGURES

FIGURES

Figure 1.1	Examples of segmentation of 3D objects (Figure Source: [1]) . . .	1
Figure 1.2	A graph constructed from labeled instances x_1 , x_2 and unlabeled instances. The label of unlabeled instance x_3 will be affected more by the label of x_1 , which is closer on the graph, than by the label of x_2 , which is farther away, even though x_2 is closer in Euclidean distance. (Figure Source: [2])	2
Figure 1.3	Example scans of all 10 subjects showing the range of ages and body shapes. (Figure Source: [3])	4
Figure 2.1	Neuron networks: (a) brain; (b) neuron network (c) neuron connecting structure (d) neuron structure (e) neuron network architecture (Figure Source: [4])	6
Figure 2.2	CNN image classification pipeline. (Figure Source: [5])	7
Figure 2.3	Taxonomy of mesh segmentation algorithms. (Figure Source: [6])	8
Figure 4.1	Loss plot of different label ratios in the network.	21
Figure 4.2	Accuracy plot of different label ratios in the network.	21
Figure 4.3	Loss plot of different network topologies.	25
Figure 4.4	Accuracy plot of different network topologies.	25
Figure 4.5	Loss plot of different adjacency impact on the loss function in the network.	29

Figure 4.6	Accuracy plot of different adjacency impact on the loss function in the network.	29
Figure 4.7	Loss plot of different activation functions in the network.	33
Figure 4.8	Accuracy plot of different activation functions in the network.	33
Figure 4.9	Loss plot of different dropout values in the network.	35
Figure 4.10	Accuracy plot of different dropout values in the network.	35
Figure 4.11	Loss plot of different batch sizes in the network.	37
Figure 4.12	Accuracy plot of different batch sizes in the network.	38

LIST OF ALGORITHMS

ALGORITHMS

Algorithm 1	Training Segmentation Network	17
-------------	---	----

LIST OF ABBREVIATIONS

3D	3 Dimensional
ML	Machine Learning
AI	Artificial Intelligence
CNN	Convolutional Neural Network
GCN	Graph Convolutional Network

CHAPTER 1

INTRODUCTION

Segmentation is one of the vital research areas in computer graphics. It is the process of breaking objects into smaller and serviceable parts. Many various solutions [7, 8, 9] were generated to this problem in the past few years. However, as the new demands in the area rise, so does the need for new solutions.

In this thesis, we will work with 3-dimensional objects that are represented as meshes. Mesh segmentation is essential in the graphics area, considering it has the primary role in animation, skeleton extraction, texture adding, and so forth. The objects our model will be dealing with has at least one limbs and diverted from human objects to octopus object. However, in this thesis, we will work with the human type of objects as a prototype.

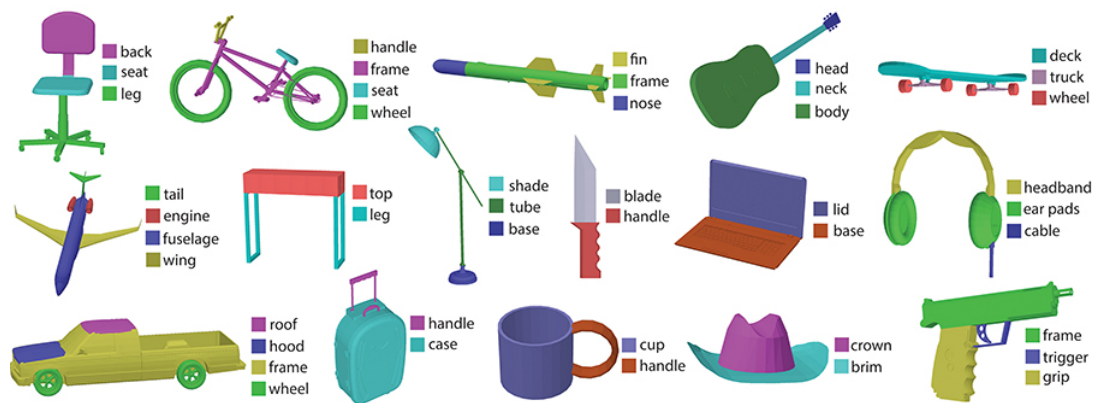


Figure 1.1: Examples of segmentation of 3D objects (Figure Source: [1])

1.1 Problem Definition

Objects represented as meshes can be viewed as a union of faces which are constructed from vertices and edges. As all faces share their edges with adjacent faces, there exists a graphical structure on the object. The problem is to cluster faces with logical adjacents. When looked at this way, the problem is not different from the clustering people on a social media platform. There is a version of a network in both problems, and this network should be analyzed before splitting into the pieces.

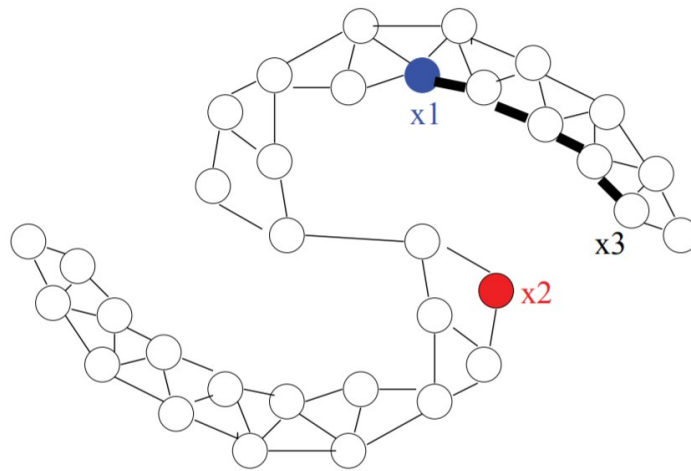


Figure 1.2: A graph constructed from labeled instances x_1 , x_2 and unlabeled instances. The label of unlabeled instance x_3 will be affected more by the label of x_1 , which is closer on the graph, than by the label of x_2 , which is farther away, even though x_2 is closer in Euclidean distance. (Figure Source: [2])

1.2 Motivation

As the number on machine learning applications increases day by day, new solutions that are suitable for the graphical problems are proposed. The primary motivation in this thesis is to use network information in the mesh more effectively so that fewer features can be used than most of the existing solutions.

Another motivation is finding a hybrid solution that is neither fully supervised nor

unsupervised so that in the minimal information, better results will be achieved. In many pattern classification problems, the acquisition of labelled training data is costly and/or time consuming, whereas unlabelled samples can be obtained easily. Semisupervised algorithms that learn from both labelled and unlabelled samples have been the focus of much research in the last few years; a comprehensive review up to 2005 can be found in [10], while more recent references include [11, 2, 12, 13, 14].

1.3 Method Overview and Contributions

Our method is a semi-supervised mesh segmentation process, which is based on a Graph Convolutional Neural Network. Graph Convolutional Neural Networks is the version of a Convolutional Neural Network which is capable of using the structure behind the graph [15].

The inputs of our method are models and some of the ground truth labels. After the training process, the outputs of our model are segmentation clusters for the given test objects. Not having pre-training for the input models makes our model end-to-end and straightforward.

The contributions in our work for the given problem are:

- Using Graph Convolutional Neural Networks in a semi-supervised mesh segmentation problem and analysis is novel to our work.
- Our model expects the whole mesh object as an input. Other segmentation models using learning techniques use single polygon of the mesh object. This technique enables our learning algorithm to consider the object, instead of the polygon.
- Our model creates the necessary features of an object by itself with the capability of Convolutional Neural Networks. This contribution makes our algorithm simple and, at the same time, more inclusive. Also, using Graph Convolutional Neural Networks, we use neighborhood information. That information allows the network to consider neighborhood labels and their features in each iteration.

- Our technique is trainable for any object in any pose. This contribution is also a benefit of using Graph Convolutional Neural Networks (Figure 1.3).



Figure 1.3: Example scans of all 10 subjects showing the range of ages and body shapes. (Figure Source: [3])

1.4 The Outline of the Thesis

This thesis consists of four chapters, excluding for this Introduction chapter. They are:

Chapter 2; In this chapter the backbone of our proposed method, which is Graph Convolutional Neural Network and its assign algorithm Convolutional Neural Networks are presented. Other mesh segmentation methods are also examined in detail.

Chapter 3; The segmentation process is explained in this section. This section starts with the pre-processing stage. Following this, we show the architecture of our algorithm and functions used. Lastly, the training stage is clarified.

Chapter 4; we begin this chapter with the dataset information used in our experiments. The experimental details and hyperparameters follow the dataset part. The evaluation and the results concerning other popular methods are then shown.

Chapter 5; this section outlines the conclusion of the thesis and also the future work in the field.

CHAPTER 2

BACKGROUND AND RELATED WORK

Before presenting the details of our process, it is important to remind the basics of two important topics; Neural Networks and Mesh Segmentation. Both of them are vast areas. We are going to examine them both in this chapter.

2.1 Artificial Neural Networks

The Neural Network is a methodology that is used in the computer engineering area inspired by the human brain, imitating its working processes. The concept relies on the artificial intelligence area. As the concept creates an imitation of a brain, the process is very similar to the human body. The process of thinking includes taking inputs using our senses, processing with the current state of our brain, creating the output signals to our body. Neural Networks do the same. We generate the input for them; they process with the current state and create an output signal.

Nevertheless, the matter is how accurate their output signals are. The methodology of Neural Networks is developmental. The output signals depend on how the neural networks are constructed and how they are fed with the data. The construction of networks is a subject specific topic, and there is no best structure that can fit all. The sequence and the elements used in the structure change the whole topology and affect the learning process altogether. The best structure for application is an active research area, and the structure we use in this work depends on a specific neural network, which is Graph Convolutional Neural Network. In this chapter, we will present the necessary information about Graph Convolutional Neural Networks, but before that we should relay some basic knowledge on Neural Networks.

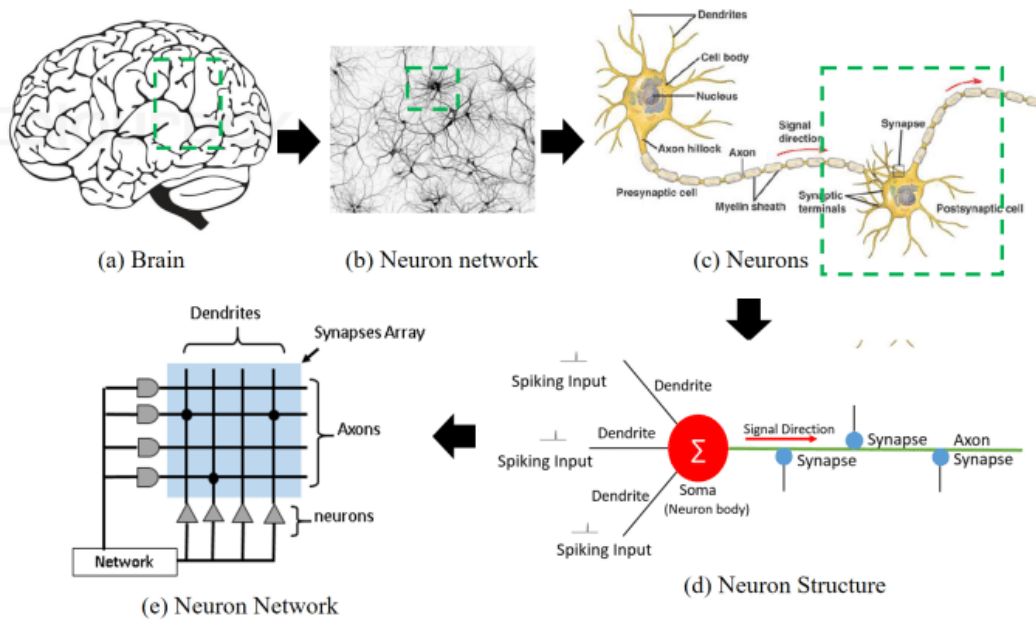


Figure 2.1: Neuron networks: (a) brain; (b) neuron network (c) neuron connecting structure (d) neuron structure (e) neuron network architecture (Figure Source: [4])

After the construction of a Neural Network, we can use this for decision making or generation of new data. However, do we know all the facts when we are born? The answer is surely no. The development of our brain continues as we are exposed to new things. These experiences include memories, education, talking, even walking is a learned process. However, how does our brain do the learning? What is changing in our brains? These are interesting questions, and in our opinion, the answer will never be known “exactly”.

Nevertheless, the scientists have approached and tried to mimic this learning progress. This progress is data-driven and depends on results. As we feed the Neural Networks with new data, we also manipulate them as we want them to be directed. As it all driven by the mathematical equations, the manipulation is also a mathematical operation, which is changing some variables in our constructed network. We summarized the learning approach in a very comprehensible way. Now we should focus on the area we utilised and give the details about it.

We use the Graph Convolutional Neural Networks. The customized version of Convolutional Neural Networks. Before going into details of Graph Convolutional Neural Networks, we begin with defining defining what the Convolutional Neural Networks

are. What sets them apart from the Neural Networks? Convolutional neural networks (CNN) utilize layers with convolving filters that are applied to local features [16]. Convolutional Neural Networks are a class of Deep Neural Networks that are primarily created for types of input data that cannot be represented straightforwardly. Image data, video data, speech data are typical examples of is. The prior feature of the Convolutional Neural Networks enables us to accomplish, in which they create their features by using convolutional filters. That is also very important in Graph Convolutional Neural Networks because it is not an easy work to explain features in graph than its structured nature.

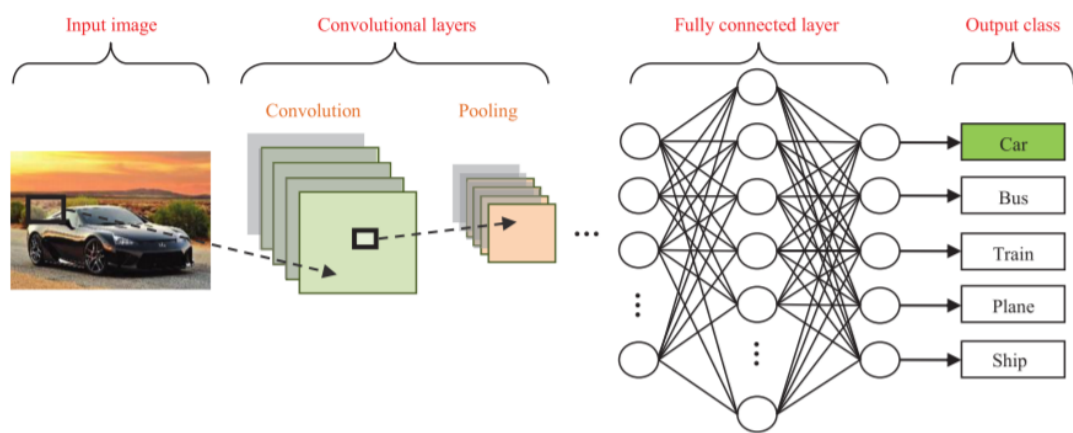


Figure 2.2: CNN image classification pipeline. (Figure Source: [5])

There are special classes of Convolutional Neural Networks to work with graph-structured data. In the study of Li et al. [17] and Duvenaud et al. [18] they have developed solutions but for the specific cases. To build a generic model, we need to transfer convolutional filters on Convolutional Neural Networks to cover graph data. That is possible with two methods. They are Spatial Construction and Spectral Construction methods.

The method used in The Henaff et al. [19], uses spectral construction for the filters. The Spectral Construction is a different presentation of the convolutional filters as a Laplacian operator. The deficiencies of this construction are high-cost in computation and filter localization. These drawbacks lead to a new solution which is proposed by Deferrard et al. [15] They introduced smooth filters by using Chebyshev polynomials in the spectral domain. After that, Kipf and Welling [20] came up with a better per-

forming Graph Convolutional Network by simplifying convolutional filtering. They achieve simplification by the generalized and differentiable version of the Weisfeiler-Lehman algorithm. In our model, we use their approach, which is the state-of-art in the area.

2.2 Mesh Segmentation

The mesh segmentation is always a trendy and active research area in computer graphics. This popularity not surprising since it has a critical role to play in shape analysis. The shapes of the objects tell us many things about them, such as composition of moving parts or their balance. So, to analyze a shape, we should analyze the mesh that constructs it. There are various alternatives proposed to create a method for segmentation. We will examine them under four general headlines, which are categorized in Rodrigues et al. [6]. They are Surface-Based Segmentation, Skeleton-Based Segmentation, Volume-Based Segmentation, and Multiple Shape Segmentation.

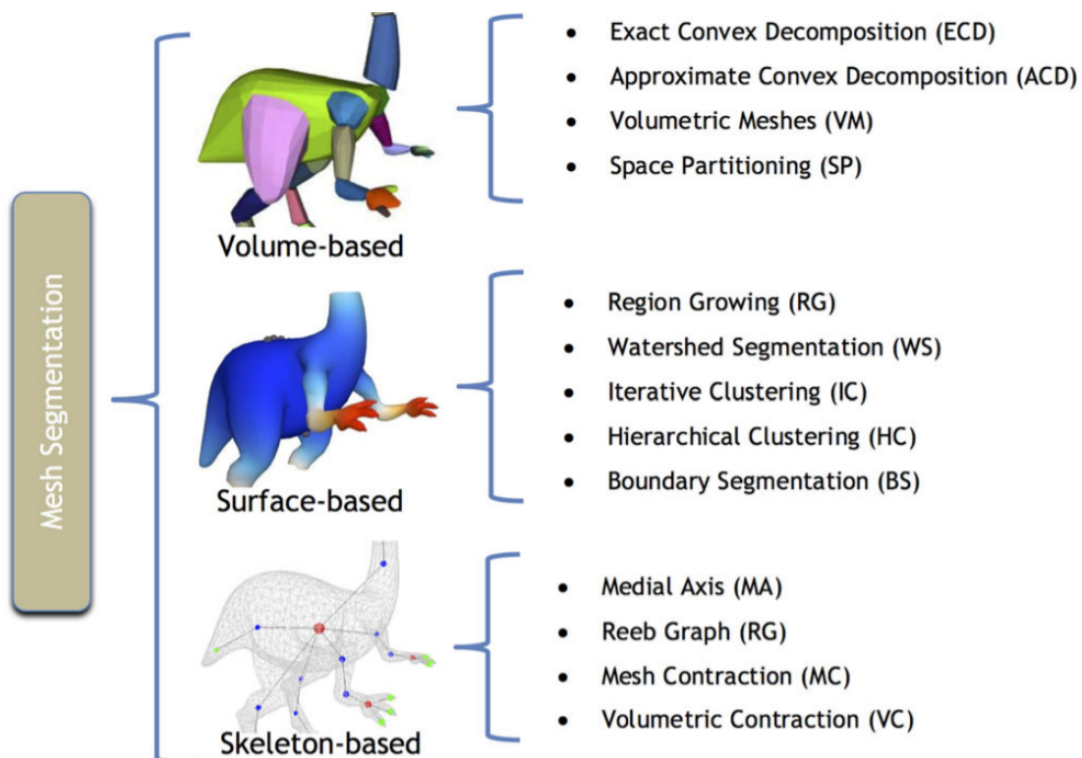


Figure 2.3: Taxonomy of mesh segmentation algorithms. (Figure Source: [6])

The Surface-Based Segmentation type of algorithms use their 3D surfaces to create 2D output regions as a result. This type of algorithms uses natural physical facts such as curves and angles in their surfaces. Examples of this type of algorithms are Adams et al. [21], which uses iterative clustering, Shapira et al. [22] and Lai et al. [23] use hierarchical clustering, Lavoue and Wolf [24] use Region Growing, Mangan and Whitaker [9] use watershed segmentation and Lin et al. [25] uses boundary segmentation.

The Skeleton-Based Segmentation type of algorithms use their 1D skeleton structure to segment objects. The skeleton has a vital role in the animation area since it defines the movement throughout. This type of algorithm has to extract skeleton information before segmentation. The examples of these types of algorithms are Aleotti and Caselli [26] which use reeb graph method, Mortera et al. [27] uses medial axis method, and Li et al. [28] use Geometric contraction.

The Volume-Based Segmentation type of algorithms use object volume as input and generate 3D volumes as output. The examples of these types of algorithms are Attene et al. [29] and Xian et al. [30] which use volumetric meshes. Chzelle et al. [31] use exact convex decomposition method, Liu et al. [32] and Kreavoy et al. [33] used approximate convex decomposition and Simatri et al. [34] used space partitioning.

The Multiple Shape Segmentation type of algorithms uses shape collections to segment objects. These collections may include fully labeled objects, partially labeled objects, or non labeled objects. These collections may contain the same type of objects or differentiating objects. Only the collection used is method dependent and decides the category that we should consider the method. The categories are Unsupervised, Semi-supervised, and Supervised as the labels indicate. Since we use Semi-Supervised Multiple Shape Segmentation, we will cover it in-depth in this section.

The logic used in the Unsupervised Multiple Shape Segmentation is that the same classes share the same segmentation models. This logic gives the method flexibility in terms of pose and object types. Sidi et al. [35] created a spectral clustering and performed analysis in object descriptor space.

The Semi-Supervised Multiple Shape Segmentation uses the only labeled data on the object to use segmentation. This can be done in several variations. Lv et al. [36] adds an energy term to the unlabeled random fields. As another variation, Shu et al. [7] introduced scribble based segmentation with weakly labeled objects. In our thesis, we also use this type of algorithm with sparsely labeled objects, but we are using the benefit of Graph Convolutional Neural Networks.

The Supervised Multiple Shape Segmentation uses the labeled objects to generate method constraints and apply the method to object that is subject to being segmentation. This methodology shows similarity with machine learning methods, and it is popular nowadays. Kalogerakis et al. [37] formulated the state-of-art in this area. However, the method used in his work needs too many features that make the method costly.

CHAPTER 3

SEGMENTATION MODEL

We will explain the steps taken in our method in this chapter. We will begin with the pre-processing stage, explain how we created inputs for our method. Then, we will give details of the architecture and structures used in the model architecture topic. Lastly, we will give details of the training and results of our method.

3.1 Pre-Processing Stage

This stage begins with the input models, which are objects consisting of meshes. To use the Convolutional Neural Networks, we know from the traditional 2D methods that we should have a uniform type of inputs for better results. Applying the rule for 3D object meshes, we need to use uniform graph structures for all meshes in our dataset. So before we generate any other information, we should downsample them and then use them as graphs. In addition to that, to use Graph Convolutional Neural Networks, we need to create a particular type of data. This particular type of data includes identification of faces to be labeled and features to use for each face. Also, the adjacent faces should be extracted from the objects.

3.1.1 Generating Labeled Nodes

This part has a crucial role in our thesis. Observation over the changing number of labeled nodes has never been done before, and it is exciting to see if the graph information can manage the segmentation with a meager number of labeled nodes. We will be discussing how the results of this part are affecting our final segmentation

in the evaluation section. We will randomly select the labeled nodes for each object in the training set separately. The only rule is not selecting neighbouring faces as labeled. This rule will let us use more neighborhood information in the model. The critical point in this chapter is that we may use label information of the object, which is intended to segment, but the test objects may not require any labels.

3.1.2 Generating Features

As we mentioned in the related work section, most of the learning-based segmentation models use a large number of features around 600 to 1000. However, our model uses only six features for each face. These feature contain the position of a face and its surface normal. For each vector, we need to store three variables, which are X, Y, and Z coordinates. These vectors are calculated by simple graph geometry techniques in the preprocessing.

3.2 Model Architecture

3.2.1 Input and Output

Our segmentation algorithm takes all types of 3D objects that can be represented as meshes. In this work, we only stick with the same types of objects, for example, humans. For our case, we need different styles and poses of human meshes that are only required to have the same number of segments. This is a requisite for training the neural network. After pre-processing steps we mentioned, the network model takes four inputs for each object in our dataset, which are the graph structure generated from mesh, the adjacencies of meshes, the features for meshes, and the some of the ground truth labels which we give as seed points.

Adjacencies of meshes consist of id couples of faces that are neighbours. The face ids should be represented as integers.

Mesh features consist of six decimal numbers for each face. They should be set in the order of center and surface normal vector. Each should be given in the order of X, Y,

Z coordinate triplets.

The ground truth labels consist of an integer for each face that starts from zero.

The output of the model is the predicted label of each face. The predictions also start from zero.

3.2.2 Loss Functions

To explain our loss function, we should begin with the training stage. As we mentioned before, we feed some inputs to our network. The details of this process will be explained in the training stage, but we should go over it briefly in this part. Our method works on an iterative basis. For each iteration, the model takes inputs and starts to predict within the basis of the current model. After this prediction, the model checks the results with the existing seed points and can check some other things with the prediction. This check decides how accurate the results are. To use the results of this assessment, we utilise loss functions to digitize this information and revise our model. This revision determines the performance at the next iteration, and so on. As a result, the accuracy and the loss functions are strongly connected. In this perspective, the loss functions are also the degree of accuracy for the network to be used. So for our problem, we should ask: How we can give a degree of accuracy when we do not know the complete ground truth information?

To this end we can use our seed points. For each iteration, we can check if we successfully labeled the ground truth labels and revise our model. We also know the graph structure, and using the structure with the seed point information should improve the results. This part of equation is given below (31).

$$L_{seed} = \left(- \sum_{p_i}^N \log(p_i) \right) / N, \quad (31)$$

where L_{seed} is the loss for given seed points, N is the number of sparse seed points.

This expression is necessary but not sufficient in our case. The reality is that we can use the graph structure more effectively if we adopt the neighborhood information

in the loss function. Except for the intersection, the network should consider the likelihood of a neighbouring faces belonging to the same segment is too high. So, we created a loss function that gets higher when too many neighbouring faces have different labels. This part of equation is given below (32).

$$L_{smooth} = \left(\sum_{i=1}^N \sum_{a_j}^3 \sum_{k=1}^L |p_{i_k} - p_{j_k}| \right) / N, \quad (32)$$

where L_{smooth} is the smoothness loss, N is the total number of faces, L is the number of adjacent faces for given face.

The final loss function is given below (33).

$$L_{total} = L_{seed} + \alpha L_{smooth}, \quad (33)$$

where α is smoothness factor. We will try varying smoothness factor values in the experiments part of the thesis.

3.2.3 Graph Convolutional Layers

The common aim of the neural network models is to create a function of features on a graph $g = (V, E)$ which takes as input:

- Feature description X_i for every node i ; summarized in a $N \times D$ feature matrix X (D: number of features, N: number of nodes)
- Adjacency matrix A (or another representation of a graph structure)

It then produces an output for each node Z ($N \times F$ feature matrix in which F is the different output features for each node). A neural network layer can be represented

as a non-linear function

$$\begin{aligned}
H^{(l+1)} &= f(H^{(l)}, A), \\
H^{(0)} &= X, \\
H^{(L)} &= Z,
\end{aligned}
\tag{34}$$

where L is the number of layers, for each layer l .

We see that all specific models differ only how $f(., .)$ is chosen and represented.

In our model, we will be using the model introduced in Kipf and Weiling[20], which is widely employed. The model is represented as;

$$\begin{aligned}
f(H^{(l)}, A) &= \theta(\hat{D}^{-1/2} \hat{A} \hat{D}^{-1/2} H^{(l)} W^{(l)}), \\
&\text{where;} \\
A &= \text{Adjacency matrix,} \\
\theta &= \text{non-linear activation function,} \\
\hat{D} &= \text{diagonal node degree matrix of, } \hat{A} \\
\hat{A} &= A + I, \\
I &= \text{identity matrix,} \\
W^{(l)} &= \text{weight matrix for layer } l.
\end{aligned}
\tag{35}$$

We used this propagation rule for each convolutional layer. However, our model consists of multiple layers. Some of these layers are graph convolutional, and some of them are linear layers. As shown in equation, for each graph convolutional layer one feature passing operation is applied.

3.3 Sparsely Labeled Training Stage

We showed a propagation model on graphs in the last chapter. Now we will introduce the problem of semi-supervised face classification in objects.

After building our model structure, we have done several experiments in order to find the optimal model. These experiments include two to ten layers of graph convolutional layer. In this chapter, we will express our learning equations with a two-layered simple model in order to keep expressions manageable. The best resultant architecture identified experimentally will be further explained in the evaluation part.

Using Softmax and Relu as a non-linear function, we can show a forward model as:

$$Z = f(X, A) = \text{softmax}(\hat{A}\text{ReLU}(\hat{A}XW^{(0)})W^{(1)}). \quad (36)$$

Here $W(0) \in R^{C \times H}$ is an input-to-hidden sized layer weight matrix with H feature maps. $W(1) \in R^{H \times F}$ is a hidden-to-output sized layer weight matrix.

At the last stage of forwarding operation, the softmax activation function is applied row-wise to calculate scores.

When we get the scores, we can evaluate the error with the loss function we have shown (37).

$$L_{total} = \left(- \sum_{p_i}^M \log(p_i) \right) / M + \alpha \left(\sum_{i=1}^N \sum_{a_j}^3 \sum_{k=1}^L |p_{i_k} - p_{j_k}| \right) / N, \quad (37)$$

where M is the number of given seed points, N is the total number of faces, α is the smoothness factor.

After calculating the error, we should update the weight matrices, $W(0)$ and $W(1)$ in this case. To update weight matrices, we have used a batch gradient descent method. These operations are applied in each iteration until convergence is achieved.

The pseudo-code of our method is given in algorithm (1).

Algorithm 1: Training Segmentation Network

Input : Sparsely Labeled & Pre-Processed Training Dataset, T ; Labeled & Pre-Processed Validation Dataset, V ; Epochs, e ; Batch Size, b ; Learning Rate, l ; Smoothness factor, α ;

Output: Weights, W

```
1 initialize  $W$ 
2 for  $e$  epochs do
3   BatchGroup  $\leftarrow$  subsets of  $T$  with size  $B$ 
4   foreach  $batch$  in BatchGroup do
5      $L \leftarrow$  calculate overall loss (Eqn. 37) for  $batch$ 
6      $grads \leftarrow$  Calculate  $L$  Backward
7     Update  $W$  with gradient  $grads$  and learning rate  $l$  by using
      Optimization Algorithm
8   end
9    $L_{val} \leftarrow$  calculate overall loss
10  if  $L_{val}$  is not converging then
11    Update  $l$ 
12  end
13 end
14 return  $W$ 
```

CHAPTER 4

EXPERIMENTS AND RESULTS

In this chapter of our thesis, we will first show and examine the datasets we used. Then we will show our different experiments performed on data. In these experiments, we will be changing the network topology, smoothness factor on loss function, labeled face ratio, dropout value, and activation functions.

Before going into the details of the experiments and results, we will cover the environment and the frameworks used. In the beginning, we have a 3D object. To visualize this object and manual labeling, we use Blender and Blender Python API. Feature and adjacency extraction are also achieved by using the same API. After applying aforementioned pre-processing operations, we should continue with the training phase. For the training and matrice operations, we use PyTorch [38] and DGL [39]. PyTorch is a commonly used open-source machine learning library that uses Torch. DGL is also an open-source machine learning library, but it supports graph operations more fluently and smoothly.

4.1 Dataset

We have used the two datasets. The first dataset is the FA [40]. The FAUST dataset has ten different subjects and for each subject ten different poses. We have used this dataset because it has correspondence information and this information lets us label models easily. This dataset is used more often in the experimental part. For the evaluation, we have the second dataset, which is used in the Princeton segmentation benchmark. The dataset consists of polygonal models supplied by Daniela Giorgi [41]. In this dataset, there are 20 different objects, each of which in 20 different

positions. That makes 400 total objects for that dataset. In our work, we only used the object “Human.” The dataset we have used does not support correspondence between the object positions. That makes the sparse labeling process harder and difficult to compare except for random labeling. The ground truth labels are also provided from the Princeton segmentation benchmark. The main reason for using this dataset was its prevalence in the area. Most of the methods use this benchmark to compare their results with the others. However, we should mention that our method does not require any necessity for the data. Having a mesh objects in different poses or styles and some of the ground truth labels are the only requirements for our model. Thus, one can apply our model to any object that is represented by meshes.

4.2 Experiments

In this part of the thesis, we will show and examine the results of the experiments we have carried out with our model.

4.2.1 Labeled Face Ratio

Using sparse semi-supervised learning can effectively achieve a good balance between classifier performance and the number of unlabeled examples retained [42]. The key point of this thesis is having sparsely labeled objects and training them with the semi supervised method. The question is; how much sparsity can our model handle?

We tried eight different levels from %0.2 to %50. The architectural details of the network (4.1), epoch/loss (4.1) and accuracy/loss (4.2) figures are given.

On the top, layers row, the layer sizes are represented. For the graph convolutional layers, the size is followed by designation "g". The “Epoch” column points out the training loop count. “Learning Rate” column points out the learning rate used in the network. The scheduler also used to adjust the learning rate. “Activation Function” indicates the activation function used in the network. “Batch Size” points out the batch size used while training the network. “Smoothness Factor” points out the

smoothness coefficient in the loss function. “Dropout” points out the dropout ratio in training.

Table 4.1: Architectural details of the changing labeled face ratio experiment.

Layers					
$6(g) \times 36(g) \times 72(g) \times 144(g) \times 288(g) \times 144(g) \times 72(g) \times 36(g) \times 4$					
Epoch	Learning Rate	Activation Func.	Batch Size	Smoothness Factor	Dropout
100	0.001	ReLU	10	0.5	0

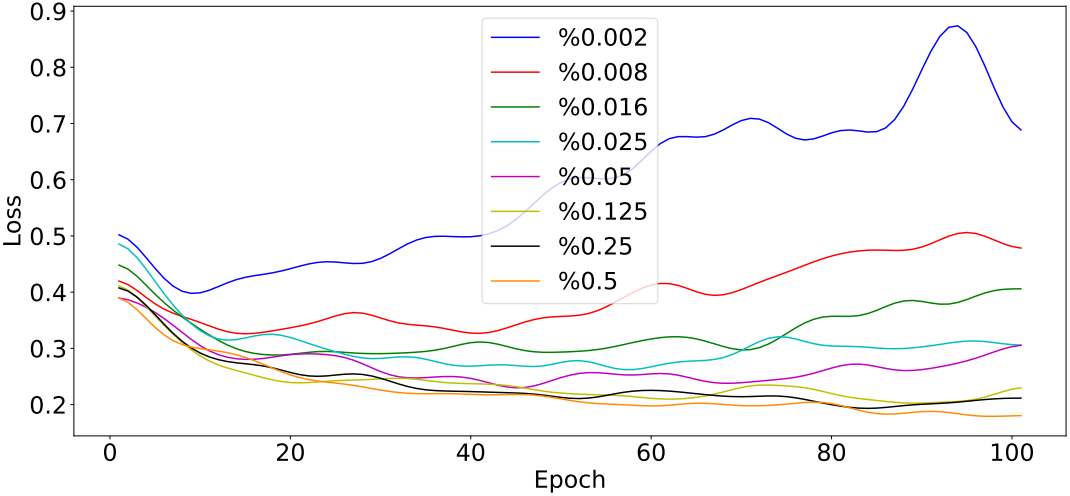


Figure 4.1: Loss plot of different label ratios in the network.

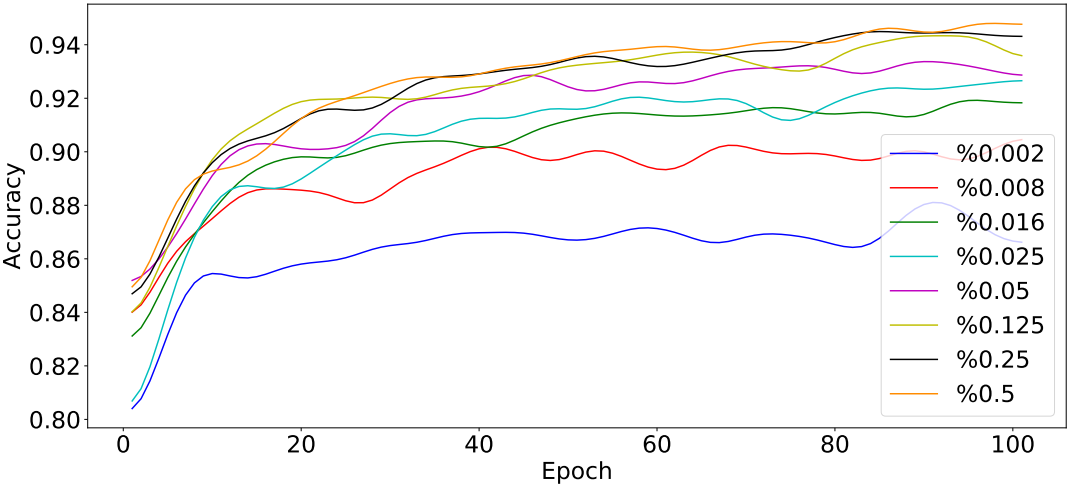


Figure 4.2: Accuracy plot of different label ratios in the network.

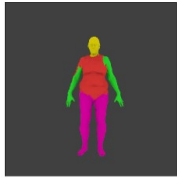



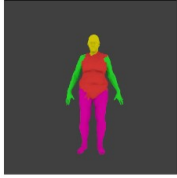
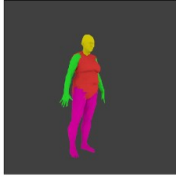






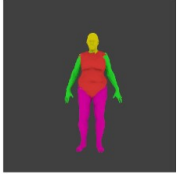
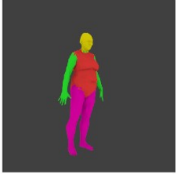




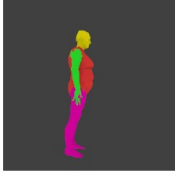

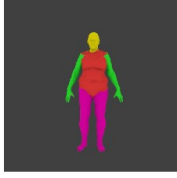



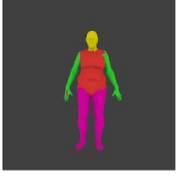



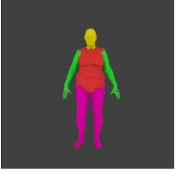



Table 4.2: Best accuracy values for different label ratios.

Sparsity	%0.2	%0.8	%1.6	%2.5	%5	%12.5	%25	%50
Accuracy	0.887	0.916	0.926	0.933	0.943	0.951	0.951	0.953

Based on the results (4.2), the accuracy of the algorithm depends on the labeled face ratio; however, the effect is exponentially decreasing.

Another interpretation is that having even a %0.2 percentage of the labeled faces; the model still manages to converge with %88.7 accuracy.

Table 4.3: Visual results of Labeled Face Ratio experiment.

%0.2				
%0.8				
%1.6				
%2.5				
%5				
%12.5				
%25				
%50				

4.2.2 Network Topology

Neural networks have been known to produce wide variations in their predictive properties for small changes in network design [43]. In our case, these changes are much more observable. This clarity of observation is because we use Graph Convolutional Neural Networks and make a layer Graph Convolutional Layer or Linear changes the whole behaviour of the layer.

We created ten different network topologies to compare with each other (4.4). While creating different topologies, we tried to make them comparable with each other. The architectural details of the network (4.5), epoch/loss (4.3) and accuracy/loss (4.4) figures are given.

Table 4.4: Different network topologies for the experiment.

NET-1	$6(g) \times 36(g) \times 72(g) \times 144(g) \times 288(g) \times 144(g) \times 72(g) \times 36(g) \times 4$
NET-2	$6(g) \times 36(g) \times 128(g) \times 256(g) \times 128(g) \times 64(g) \times 4$
NET-3	$6(g) \times 36(g) \times 128(g) \times 256(g) \times 128(g) \times 4$
NET-4	$6(g) \times 64(g) \times 128(g) \times 256(g) \times 512(g) \times 4$
NET-5	$6(g) \times 512(g) \times 1024(g) \times 512(g) \times 64 \times 4$
NET-6	$6(g) \times 64(g) \times 256(g) \times 64 \times 16 \times 4$
NET-7	$6(g) \times 64(g) \times 256(g) \times 512 \times 64 \times 4$
NET-8	$6(g) \times 64(g) \times 256(g) \times 128 \times 64 \times 32 \times 4$
NET-9	$6(g) \times 36(g) \times 72(g) \times 144(g) \times 288(g) \times 576(g) \times 288(g) \times 144(g) \times 72(g) \times 36(g) \times 4$
NET-10	$6(g) \times 36(g) \times 72(g) \times 144(g) \times 288(g) \times 576(g) \times 288(g) \times 144(g) \times 72(g) \times 36 \times 4$

Table 4.5: Parameters for network topology experiment.

Epoch	Labeled Face Ratio	Learning Rate	Activation Func.	Batch Size	Smoothness Factor	Dropout
100	%25	0.001	ReLU	10	0.5	0

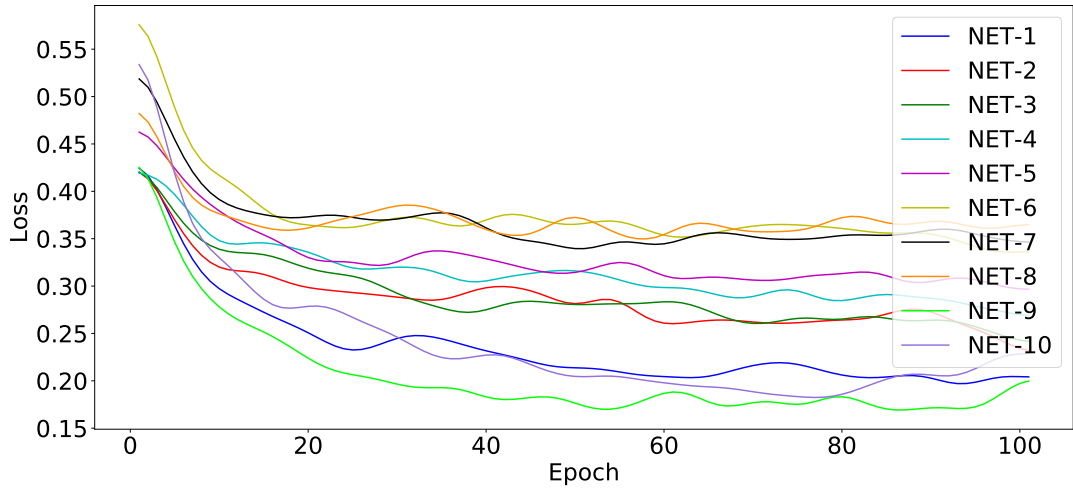


Figure 4.3: Loss plot of different network topologies.

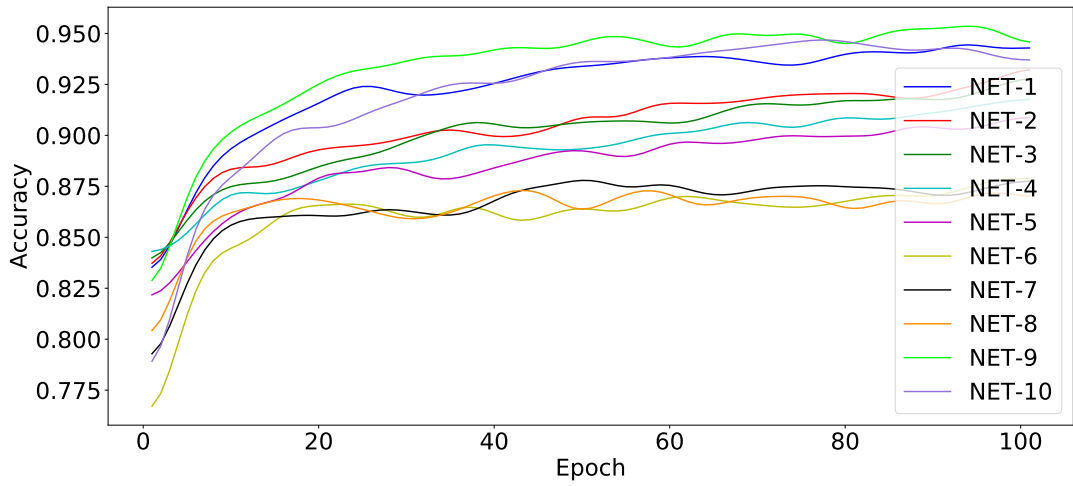


Figure 4.4: Accuracy plot of different network topologies.

Table 4.6: Best accuracy values for different network topologies.

NET-1	NET-2	NET-3	NET-4	NET-5
0.952	0.940	0.932	0.925	0.921
NET-6	NET-7	NET-8	NET-9	NET-10
0.886	0.890	0.886	0.960	0.951

Based on the results (4.6), NET-9 performed best with the highest accuracy. However,

we can make additional observations by comparing other topologies with each other.

- Increasing Graph Convolutional Layer count in topology affects the performance positively (NET-1:NET-9,NET-2:NET-3).
- Increasing linear layer sizes in topology affects the performance positively (NET-6:NET-7).
- Increasing Graph Convolutional Layer sizes does not always affect positively (NET-3:NET-4). This result may be related to the drop in the last layer in NET-4.
- Increasing the linear layer count in topology does not affect the performance (NET-6:NET-8).

We can see that adding a new Graph Convolutional Layer affects the performance positively as expected but having an additional graph convolutional layer makes the learning process much more longer. So, we will not be using NET-9 for the rest of the experiment but we will be using NET-1 instead since the difference is small.

Table 4.7: Visual results of Network Topology experiment (1).

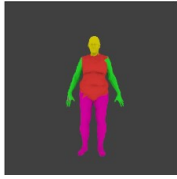
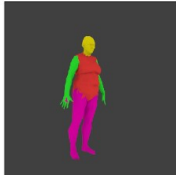






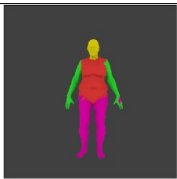
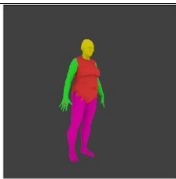
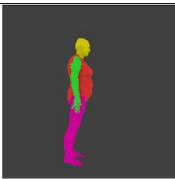
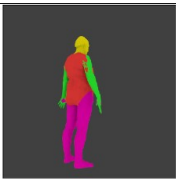




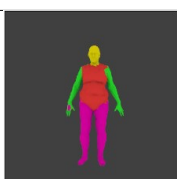



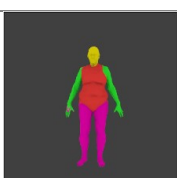


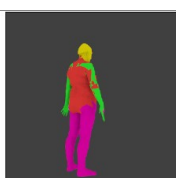
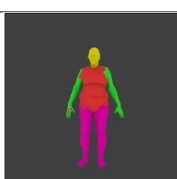



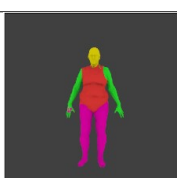



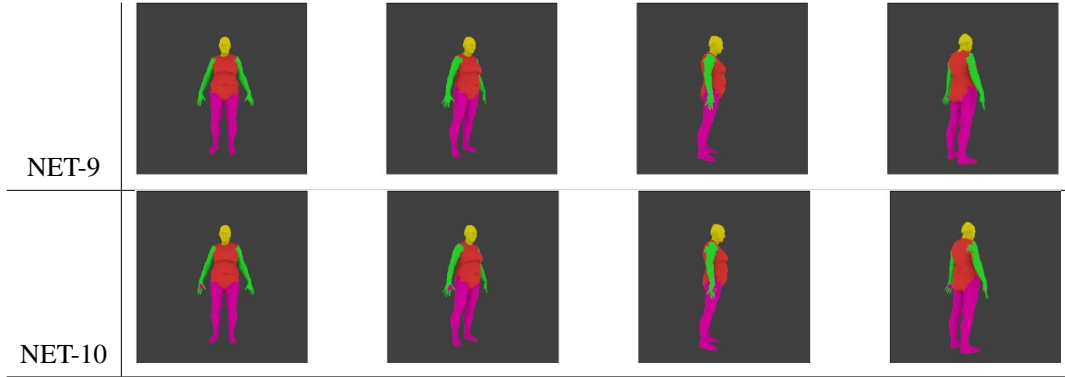
NET-1				
NET-2				
NET-3				
NET-4				
NET-5				
NET-6				
NET-7				
NET-8				

Table 4.8: Visual results of Network Topology experiment(2).



4.2.3 Adjacency Impact on the Loss function

We have used a problem specific loss function (41) and mentioned it on the model section.

$$L_{total} = L_{seed} + \alpha L_{smooth}. \quad (41)$$

However, this specific loss function added a new hyperparameter to our learning process, and in order to analyze the effects of this parameter, we should run experiments

We selected ten different alpha values to compare with each other. The architectural details of the network (4.9), epoch/loss (4.5) and accuracy/loss (4.6) figures are given.

Table 4.9: Architectural details of the changing adjacency impact on the loss function experiment.

Layers					
$6(g) \times 36(g) \times 72(g) \times 144(g) \times 288(g) \times 144(g) \times 72(g) \times 36(g) \times 4$					
Epoch	Learning Rate	Activation Func.	Batch Size	Labeled Face Ratio	Dropout
100	0.001	ReLU	10	%25	0

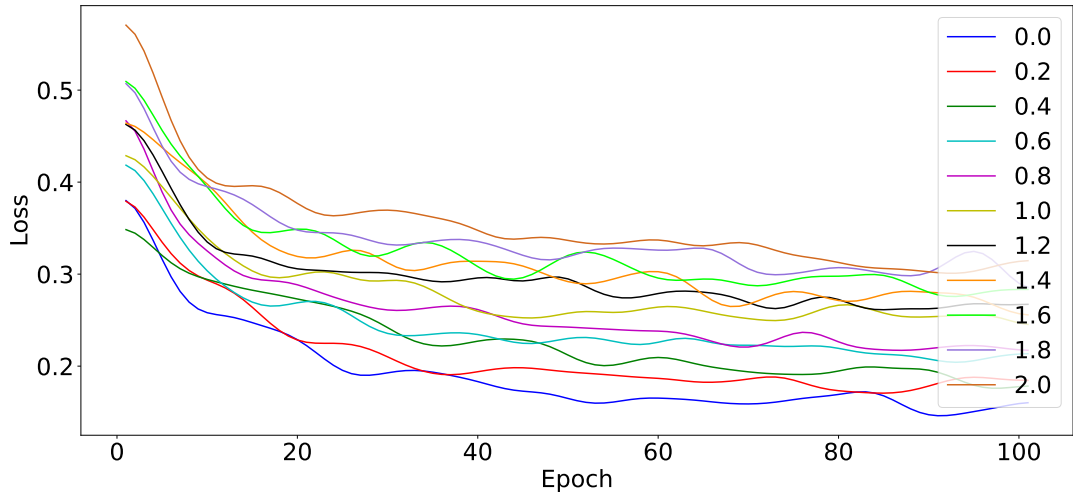


Figure 4.5: Loss plot of different adjacency impact on the loss function in the network.

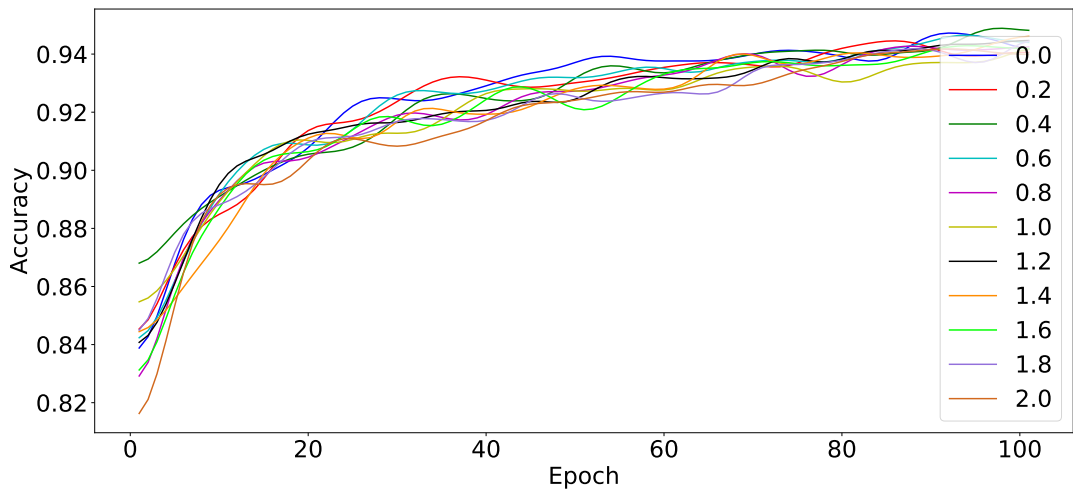


Figure 4.6: Accuracy plot of different adjacency impact on the loss function in the network.

Table 4.10: Best accuracy values for different adjacency impacts (1).

α	0	0.2	0.4	0.6	0.8	1.0
Accuracy	0.903	0.949	0.955	0.954	0.949	0.952

Table 4.11: Best accuracy values for different adjacency impacts (2).

α	1.2	1.4	1.6	1.8	2.0
Accuracy	0.950	0.949	0.951	0.952	0.950

It is easy to interpret that we should expect fewer stains with the lower rates of alpha. The results are supporting the expectation which is a positive for our model. However, the effect of neighbouring faces seem to cause disruptions on the joining points. For this reason, accuracies are increasing at the beginning and then starts to decrease.

Based on the results (4.10, 4.11), α value of 0.4 performed best. So, for the rest of the experiments, we are going to set the alpha value as 0.4.

Table 4.12: Visual results of Adjacency Impact on the Loss function experiment (1).









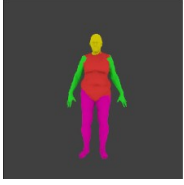

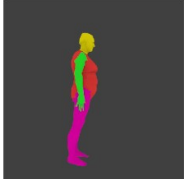

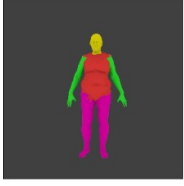

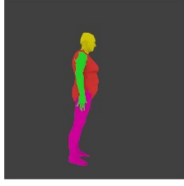

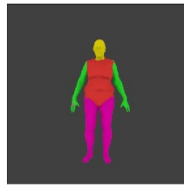





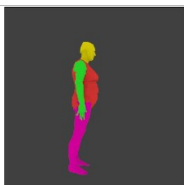





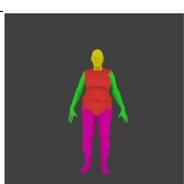


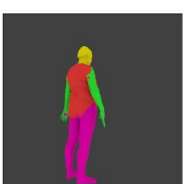
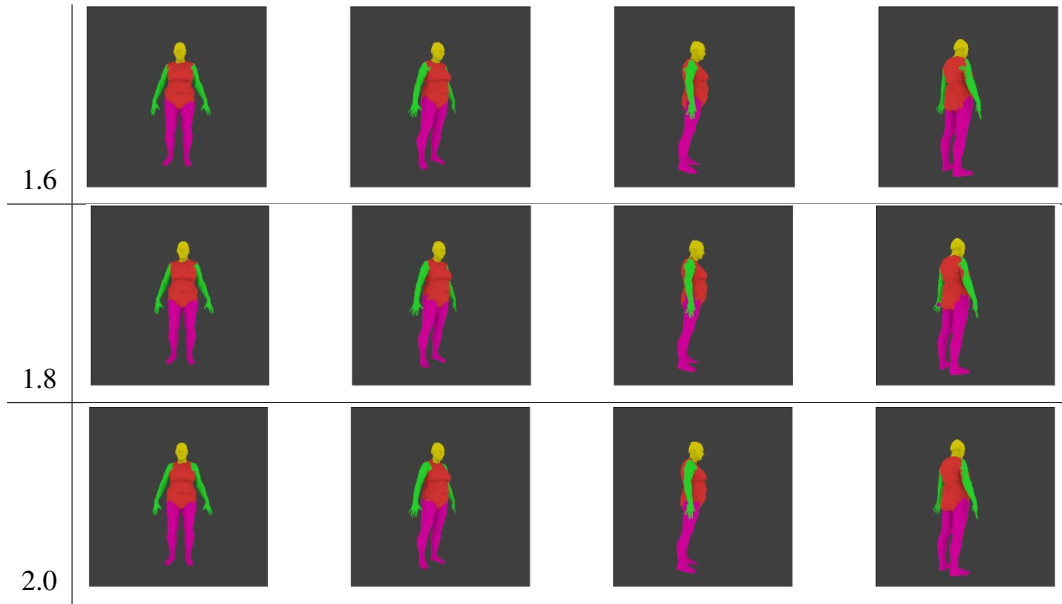
0				
0.2				
0.4				
0.6				
0.8				
1.0				
1.2				
1.4				

Table 4.13: Visual results of Adjacency Impact on the Loss function experiment (2).



4.2.4 Activation Function

The activation functions are attached to each neuron in the network and decide whether or not to block the activity of it or not. Having them in every layer makes their role important. The choice of activation functions in deep networks has a significant effect on the training dynamics and task performance [44]. Choice of activation functions are problem dependent.

We selected four different activation functions for comparison. They are ReLU, Leaky-ReLU, Tanh and Sigmoid. They are all non-linear activation functions, and they all have prominent features. The architectural details of the network (4.14), epoch/loss (4.7) and accuracy/loss (4.8) figures are given.

Table 4.14: Architectural details of the different activation functions experiment.

Layers					
$6(g) \times 36(g) \times 72(g) \times 144(g) \times 288(g) \times 144(g) \times 72(g) \times 36(g) \times 4$					
Epoch	Learning Rate	Labeled Face Ratio	Batch Size	Smoothness Factor	Dropout
100	0.001	%25	10	0.5	0

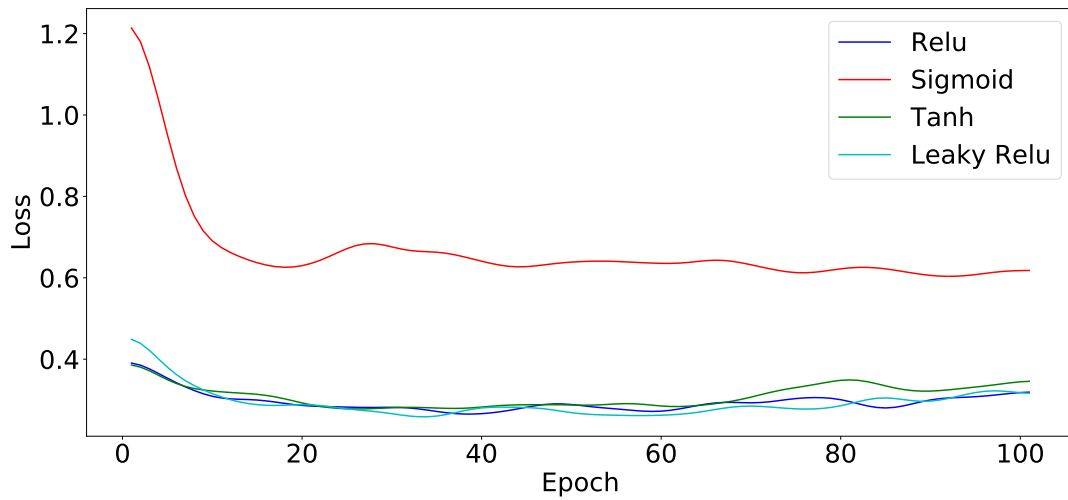


Figure 4.7: Loss plot of different activation functions in the network.

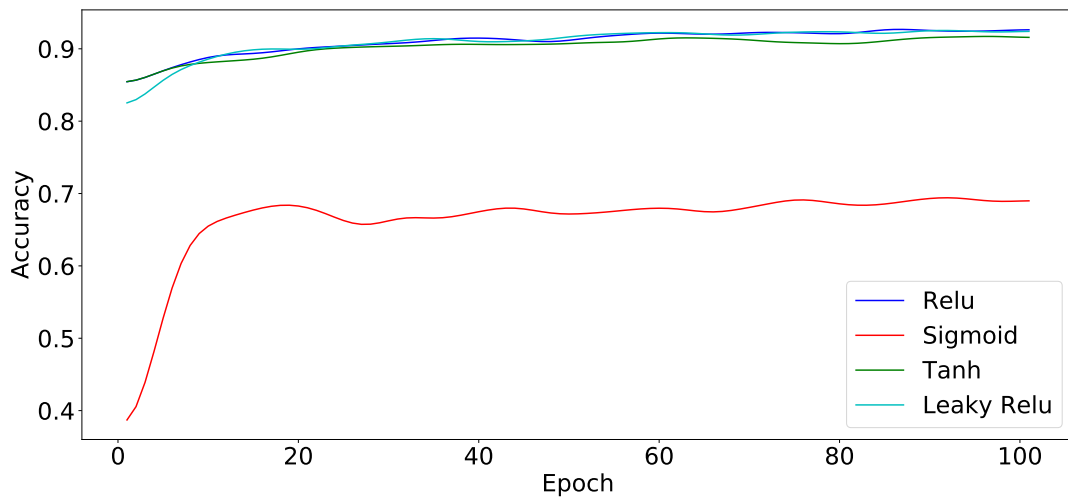


Figure 4.8: Accuracy plot of different activation functions in the network.













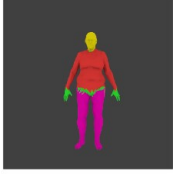



Table 4.15: Best accuracy values for different activation functions.

Activation func.	ReLU	L-ReLU	Tanh	Sigmoid
Accuracy	0.934	0.938	0.924	0.717

Based on the results (4.15), L-ReLU performed best. However, ReLU and Tanh also performed similarly. The reason why Sigmoid performed poorly should relate to the problem of vanishing gradients. Sigmoid function saturates and kills the gradients by

its nature. So, for the rest of the experiments, we avoided using Sigmoid, and mostly, we used ReLU.

Table 4.16: Visual results of Activation Function experiment.

ReLU				
L-ReLU				
Tanh				
Sigmoid				

4.2.5 Dropout

Dropout is a technique for improving neural networks by reducing overfitting [45]. So, it is a regularization technique and widely used in current state-of-art models.

We selected four different dropout ratios for comparison. They are 0, 0.1, 0.3 and 0.5. The architectural details of the network (4.17), epoch/loss (4.9) and accuracy/loss (4.10) figures are given.

Table 4.17: Architectural details of the changing dropout ratio experiment.

Layers					
$6(g) \times 36(g) \times 72(g) \times 144(g) \times 288(g) \times 144(g) \times 72(g) \times 36(g) \times 4$					
Epoch	Learning Rate	Activation Func.	Batch Size	Smoothness Factor	Labeled Face Ratio
100	0.001	ReLU	10	0.5	%2.5

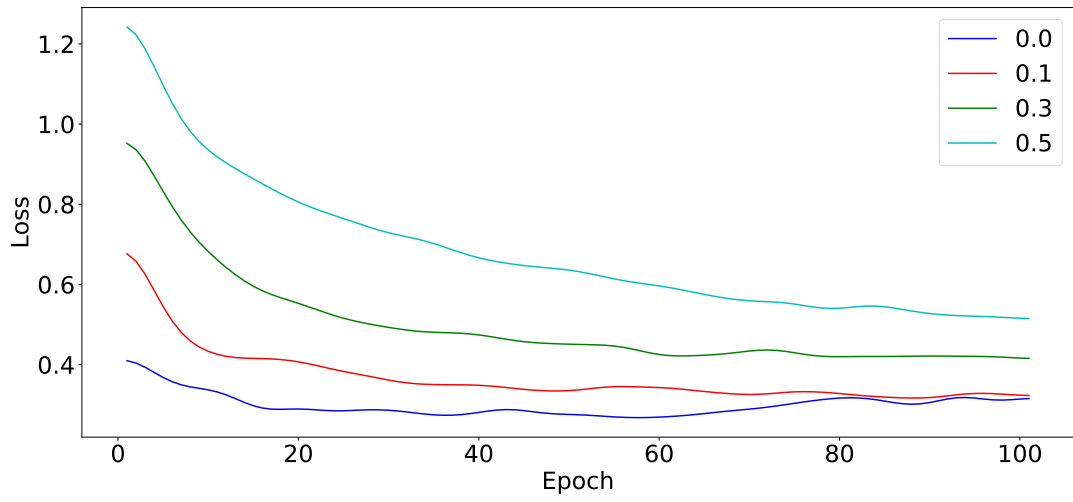


Figure 4.9: Loss plot of different dropout values in the network.

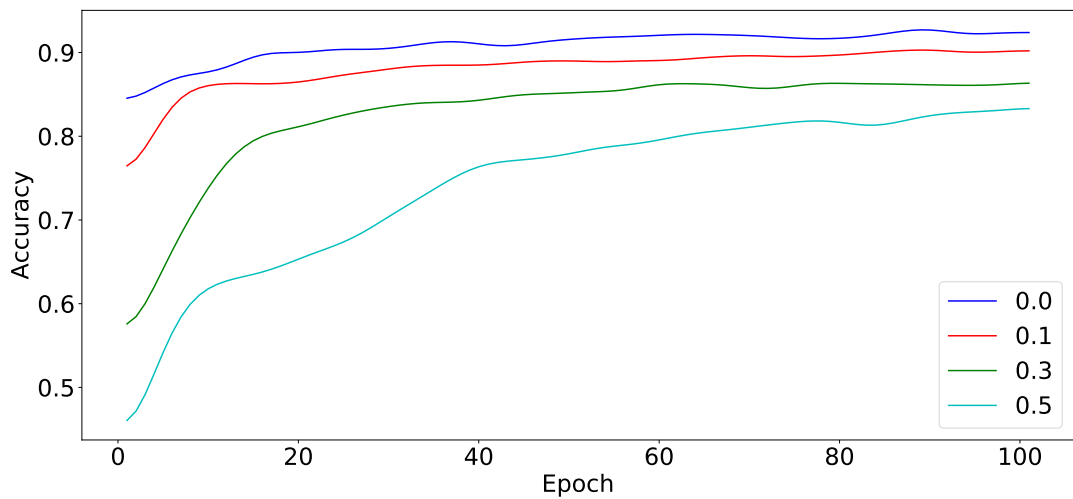


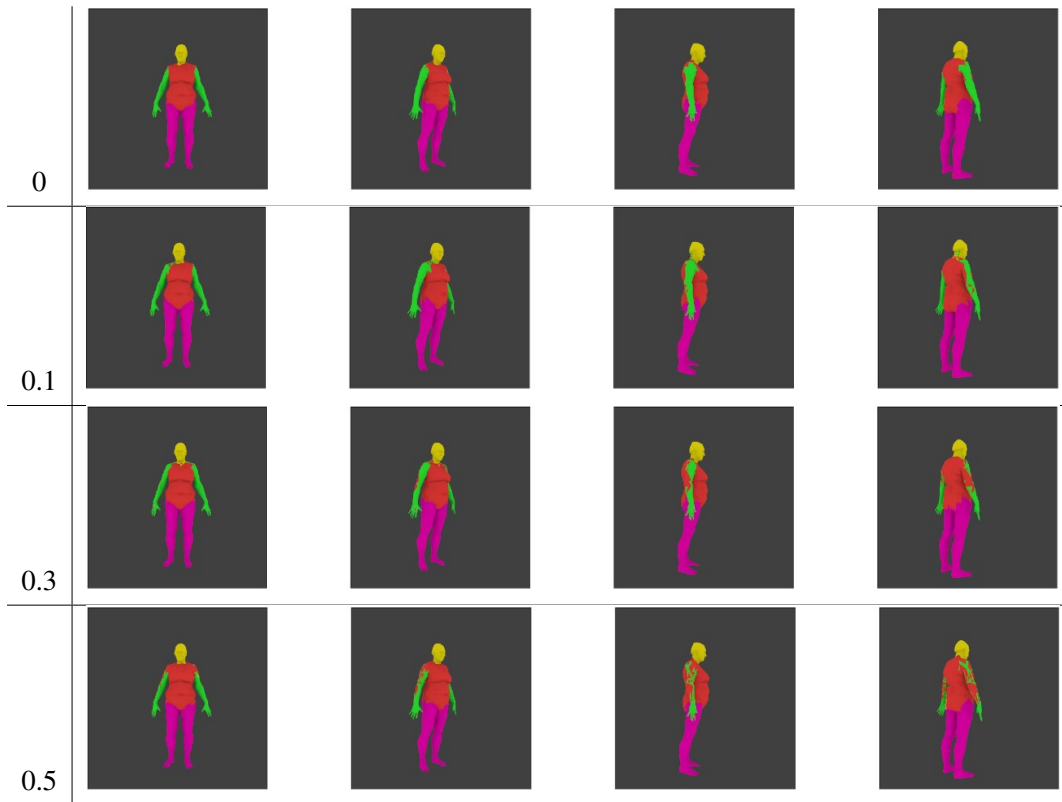
Figure 4.10: Accuracy plot of different dropout values in the network.

Table 4.18: Best accuracy values for different dropout values.

Dropout	0	0.1	0.3	0.5
Accuracy	0.937	0.912	0.876	0.840

Based on the results (4.18), zero dropout performed best. This result shows that we have no overfitting in our training model. This may be related to having different graphs for each of our models. So, for the rest of our experiments, we will not use the dropout technique.

Table 4.19: Visual results of Dropout experiment.



4.2.6 Batch Size

More extensive networks and large datasets result in longer training times which impede progress of research and development [46]. The stochastic gradient descent backpropagation method allows handling training using batches and is also an effi-

cient way of parallelization. Nevertheless, it has been observed that when using large batch sizes, there is a persistent degradation in generalization performance - known as the "generalization gap" phenomenon [47].

We selected five different batch sizes to compare with each other. The architectural details of the network (4.20), epoch/loss (4.11) and accuracy/loss (4.12) figures are given.

Table 4.20: Architectural details of the changing batch size experiment.

Layers					
$6(g) \times 36(g) \times 72(g) \times 144(g) \times 288(g) \times 144(g) \times 72(g) \times 36(g) \times 4$					
Epoch	Learning Rate	Activation Func.	Labeled Face Ratio	Smoothness Factor	Dropout
100	0.001	ReLU	%2.5	0.5	0

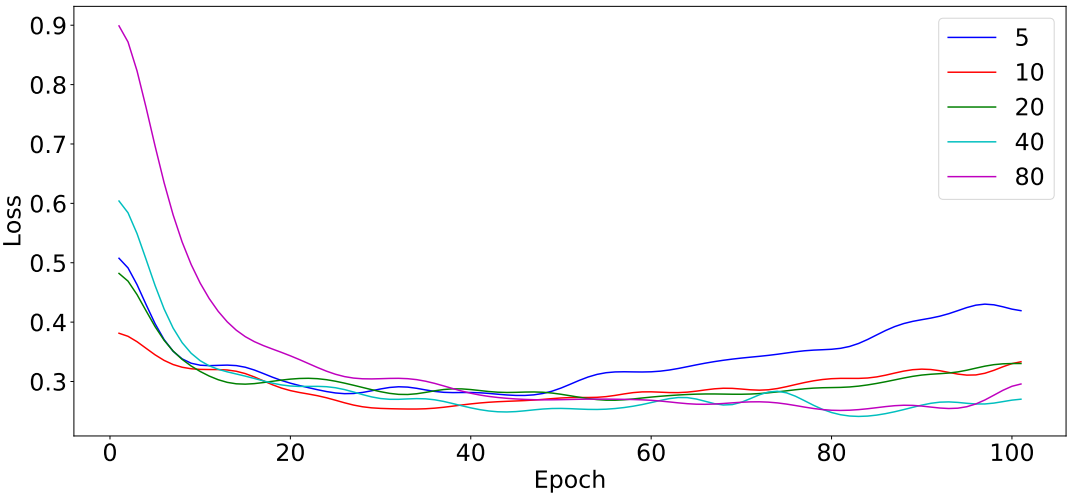


Figure 4.11: Loss plot of different batch sizes in the network.

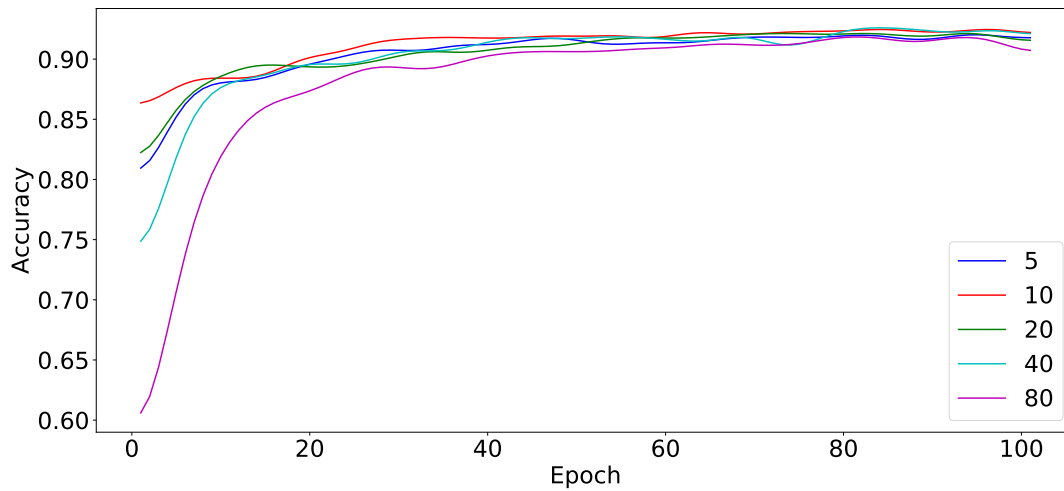


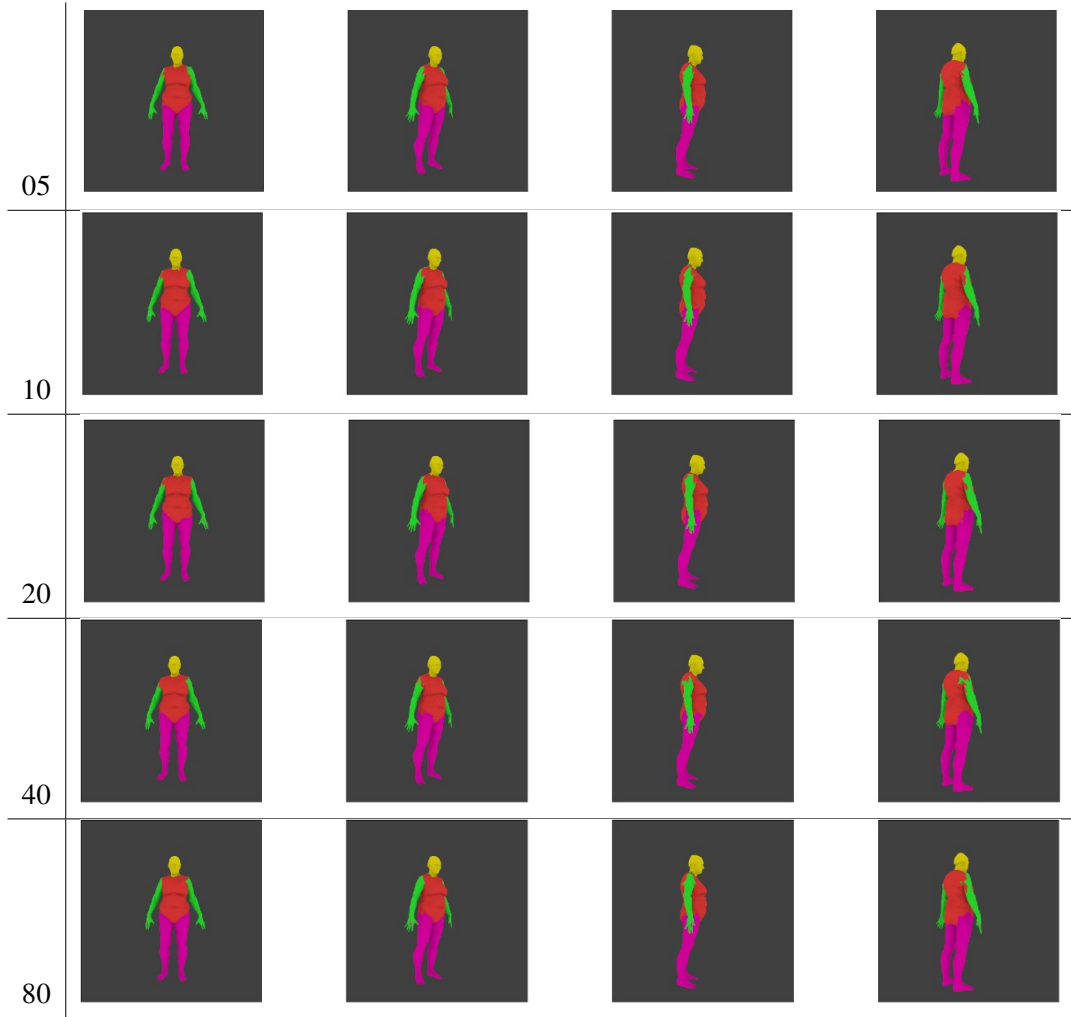
Figure 4.12: Accuracy plot of different batch sizes in the network.

Table 4.21: Best accuracy values for different batch sizes.

Batch Size	5	10	20	40	80
Accuracy	0.926	0.935	0.934	0.934	0.928

Based on the results (4.21), batch size ten performed best. However, batch size 40 performed very similar with respect to 10. In order to take advantage of parallelization, 40 is four times better than 10. So, for the rest of our experiments, we will use the batch size 40.

Table 4.22: Visual results of Batch Size experiment.



4.3 Evaluation

We evaluated our method on the Princeton Segmentation Benchmark dataset, as we mentioned in the dataset section. The benchmark comprises a data set with 4,300 manually generated segmentations for 380 surface meshes of 19 different object categories, and it includes software for analyzing 11 geometric properties of segmentations [48]. We will only be using the human object category in this benchmark.

While evaluating our method with the 20 different objects, we have used the leave-one-out cross-validation technique. This technique requires using 19 objects for the training except for the one which we used in testing.

We have presented several observations in the experiments section; We have also shown that the labeled face ratio affects our score in an exponentially decreasing manner. Thus, before having benchmark scores, we have tried our method with several labeled ratios (4.23).

Table 4.23: Accuracy comparison of our method for human category of 3D shapes in Princeton Segmentation Benchmark with different labeled face ratios.

Sparsity	%0.2	%0.8	%1.6	%2.5	%5	%12.5	%25	%50
Accuracy	0.659	0.797	0.838	0.847	0.855	0.870	0.874	0.877

After testing our method with the different labeled face ratios, we decided to get the benchmark score for three different labeled face ratios. By doing this, we will be showing how sparsity can be handled by using the method we have used.

The Rand Index scores of segmentation for the human category of 3D shapes in Princeton Segmentation Benchmark with different methods is given below (4.24). The lower Rand Index scores stand for better performances.

Table 4.24: The Rand Index scores of segmentation for the human category of 3D shapes in Princeton Segmentation Benchmark with different methods.

Method	Ours %50	Ours %12.5	Ours %1.6	PMC[49]
Rand Index	0.112	0.116	0.124	0.059
Method	Scribble[7]	WcSeg[50]	RandCuts[51]	NormCuts[51]
Rand Index	0.064	0.084	0.117	0.126
Method	Kmeans[52]	SDF[22]	FitPrim[53]	CoreExtra[54]
Rand Index	0.136	0.133	0.139	0.199

CHAPTER 5

CONCLUSIONS AND DISCUSSION

In this thesis, we have produced and analyzed a mesh segmentation method for the sparsely labeled 3D objects. We have adopted the mesh structure into a graph model in order to take advantage of Graph Convolutional Neural Networks. We tried to manage sparsely labeled training data by using the graph structures and also editing the loss function in the way which uses the adjacency information more effectively.

As we mentioned before, using the Graph Convolutional Neural Networks for the sparsely labeled 3D mesh segmentation problem is novel to our work. Analyzing the behaviors of the network for the different setups has been shown experimentally in the experiments section. We have shown the power of Graph Convolutional Neural Networks on the sparsely labeled mesh segmentation problem by reaching similar results with the labeled face percentages of %1.6 and %50. Using the semi-supervised learning method for sparsely labeled data was one of the primary motivation that we have mentioned and the results were sufficient to prove the power of the method we have used in terms of handling the sparsity.

In the evaluation part of this thesis we have shown that using the proposed method, we got similar results to most of the state of the art methods by only using 1.6 percent of face label information and by only having six features for each face. Still, our method does not rank away the best compared with some of the leading methods. However, training with more data than only 19 models and further improving the method as described in the future work, will improve its performance significantly.

5.1 Future Work

Most of our erroneous labels are around the joints of segments as all our visual results clearly show. We have come up with a number of ideas that would increase the performance for the faces around joints. Firstly, we may turn this model into a GAN network which consists of two networks. While one network remains the same and trying to segment our model, the another network is pre-trained to check if the segmentation is done correctly. By doing this, we could come up with a solution that is closer to the segments that we used while training the pre-trained network. Another idea is adding new and more powerful features that include more information about the face, such as curvature. Apart from all these suggestions, we could add a post-process for cleaning the missing stains or bad junction points.

REFERENCES

- [1] E. Kalogerakis, M. Averkiou, S. Maji, and S. Chaudhuri, “3D shape segmentation with projective convolutional networks,” in *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [2] X. Zhu and A. B. Goldberg, “Introduction to semi-supervised learning,” *Synthesis lectures on artificial intelligence and machine learning*, vol. 3, no. 1, pp. 1–130, 2009.
- [3] F. Bogo, J. Romero, M. Loper, and M. J. Black, “FAUST: Dataset and evaluation for 3D mesh registration,” in *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, (Piscataway, NJ, USA), IEEE, June 2014.
- [4] H. An, “Opportunities and challenges on nanoscale 3d neuromorphic computing system,” 10 2017.
- [5] W. Rawat and Z. Wang, “Deep convolutional neural networks for image classification: A comprehensive review,” *Neural computation*, vol. 29, no. 9, pp. 2352–2449, 2017.
- [6] R. S. Rodrigues, J. F. Morgado, and A. J. Gomes, “Part-based mesh segmentation: A survey,” in *Computer Graphics Forum*, vol. 37, pp. 235–274, Wiley Online Library, 2018.
- [7] Z. Shu, X. Shen, S. Xin, Q. Chang, J. Feng, L. Kavan, and L. Liu, “Scribble based 3d shape segmentation via weakly-supervised learning,” *IEEE transactions on visualization and computer graphics*, 2019.
- [8] E. Kalogerakis, A. Hertzmann, and K. Singh, “Learning 3d mesh segmentation and labeling,” in *ACM Transactions on Graphics (TOG)*, vol. 29, p. 102, ACM, 2010.
- [9] A. P. Mangan and R. T. Whitaker, “Partitioning 3d surface meshes using water-

- shed segmentation,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 4, pp. 308–321, 1999.
- [10] X. J. Zhu, “Semi-supervised learning literature survey,” tech. rep., University of Wisconsin-Madison Department of Computer Sciences, 2005.
- [11] O. Chapelle, B. Scholkopf, and A. Zien, “Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews],” *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 542–542, 2009.
- [12] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, “Semi-supervised learning with deep generative models,” in *Advances in neural information processing systems*, pp. 3581–3589, 2014.
- [13] J. Weston, F. Ratle, H. Mobahi, and R. Collobert, “Deep learning via semi-supervised embedding,” in *Neural Networks: Tricks of the Trade*, pp. 639–655, Springer, 2012.
- [14] J. Turian, L. Ratinov, and Y. Bengio, “Word representations: a simple and general method for semi-supervised learning,” in *Proceedings of the 48th annual meeting of the association for computational linguistics*, pp. 384–394, Association for Computational Linguistics, 2010.
- [15] M. Defferrard, X. Bresson, and P. Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering,” in *Advances in neural information processing systems*, pp. 3844–3852, 2016.
- [16] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *et al.*, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [17] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, “Gated graph sequence neural networks,” *arXiv preprint arXiv:1511.05493*, 2015.
- [18] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, “Convolutional networks on graphs for learning molecular fingerprints,” in *Advances in neural information processing systems*, pp. 2224–2232, 2015.

- [19] M. Henaff, J. Bruna, and Y. LeCun, “Deep convolutional networks on graph-structured data,” *arXiv preprint arXiv:1506.05163*, 2015.
- [20] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” *arXiv preprint arXiv:1609.02907*, 2016.
- [21] B. Adams, L. Guibas, Q. Huang, and M. Wicke, “Shape decomposition using modal analysis,” 2009.
- [22] L. Shapira, A. Shamir, and D. Cohen-Or, “Consistent mesh partitioning and skeletonisation using the shape diameter function,” *The Visual Computer*, vol. 24, no. 4, p. 249, 2008.
- [23] Y.-K. Lai, S.-M. Hu, R. R. Martin, and P. L. Rosin, “Fast mesh segmentation using random walks,” in *Proceedings of the 2008 ACM symposium on Solid and physical modeling*, pp. 183–191, ACM, 2008.
- [24] G. Lavoué and C. Wolf, “Markov random fields for improving 3d mesh analysis and segmentation.,” in *3DOR*, pp. 25–32, 2008.
- [25] H.-Y. S. Lin, H.-Y. M. Liao, and J.-C. Lin, “Visual salience-guided mesh decomposition,” *IEEE Transactions on Multimedia*, vol. 9, no. 1, pp. 46–57, 2006.
- [26] J. Aleotti and S. Caselli, “A 3d shape segmentation approach for robot grasping by parts,” *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 358–366, 2012.
- [27] M. Mortara, G. Patané, and M. Spagnuolo, “From geometric to semantic human body models,” *Computers & Graphics*, vol. 30, no. 2, pp. 185–196, 2006.
- [28] X. Li, T. W. Woon, T. S. Tan, and Z. Huang, “Decomposing polygon meshes for interactive applications,” in *Proceedings of the 2001 symposium on Interactive 3D graphics*, pp. 35–42, ACM, 2001.
- [29] M. Attene, M. Mortara, M. Spagnuolo, and B. Falcidieno, “Hierarchical convex approximation of 3d shapes for fast region selection,” in *Computer graphics forum*, vol. 27, pp. 1323–1332, Wiley Online Library, 2008.
- [30] C. Xian, S. Gao, and T. Zhang, “An approach to automated decomposition of volumetric mesh,” *Computers & Graphics*, vol. 35, no. 3, pp. 461–470, 2011.

- [31] B. Chazelle, “Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm,” *SIAM Journal on Computing*, vol. 13, no. 3, pp. 488–507, 1984.
- [32] H. Liu, W. Liu, and L. J. Latecki, “Convex shape decomposition,” in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 97–104, IEEE, 2010.
- [33] V. K. D. J. A. Sheffer, “Shuffler: Modeling with interchangeable parts,” *Visual Computer journal*, 2007.
- [34] P. Simari, D. Nowrouzezahrai, E. Kalogerakis, and K. Singh, “Multi-objective shape segmentation and labeling,” in *Computer Graphics Forum*, vol. 28, pp. 1415–1425, Wiley Online Library, 2009.
- [35] O. Sidi, O. van Kaick, Y. Kleiman, H. Zhang, and D. Cohen-Or, *Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering*, vol. 30. ACM, 2011.
- [36] J. Lv, X. Chen, J. Huang, and H. Bao, “Semi-supervised mesh segmentation and labeling,” in *Computer Graphics Forum*, vol. 31, pp. 2241–2248, Wiley Online Library, 2012.
- [37] E. Kalogerakis, M. Averkiou, S. Maji, and S. Chaudhuri, “3d shape segmentation with projective convolutional networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3779–3788, 2017.
- [38] M. Fey and J. E. Lenssen, “Fast graph representation learning with pytorch geometric,” *arXiv preprint arXiv:1903.02428*, 2019.
- [39] M. Wang, L. Yu, D. Zheng, Q. Gan, Y. Gai, Z. Ye, M. Li, J. Zhou, Q. Huang, C. Ma, *et al.*, “Deep graph library: Towards efficient and scalable deep learning on graphs,” *arXiv preprint arXiv:1909.01315*, 2019.
- [40] F. Bogo, J. Romero, M. Loper, and M. J. Black, “Faust: Dataset and evaluation for 3d mesh registration,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3794–3801, 2014.

- [41] D. Giorgi, S. Biasotti, and L. Paraboschi, “Shape retrieval contest 2007: Water-tight models track,” *SHREC competition*, vol. 8, no. 7, 2007.
- [42] S. Sun and J. Shawe-Taylor, “Sparse semi-supervised learning using conjugate functions,” *Journal of Machine Learning Research*, vol. 11, no. Sep, pp. 2423–2455, 2010.
- [43] S. Owusu-Ababio, “Effect of neural network topology on flexible pavement cracking prediction,” *Computer-Aided Civil and Infrastructure Engineering*, vol. 13, no. 5, pp. 349–355, 1998.
- [44] P. Ramachandran, B. Zoph, and Q. V. Le, “Searching for activation functions,” *arXiv preprint arXiv:1710.05941*, 2017.
- [45] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.
- [46] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, “Accurate, large minibatch sgd: Training imagenet in 1 hour,” *arXiv preprint arXiv:1706.02677*, 2017.
- [47] E. Hoffer, I. Hubara, and D. Soudry, “Train longer, generalize better: closing the generalization gap in large batch training of neural networks,” in *Advances in Neural Information Processing Systems*, pp. 1731–1741, 2017.
- [48] X. Chen, A. Golovinskiy, and T. Funkhouser, “A benchmark for 3d mesh segmentation,” in *Acm transactions on graphics (tog)*, vol. 28, p. 73, ACM, 2009.
- [49] L. Fan, L. Lic, and K. Liu, “Paint mesh cutting,” in *Computer graphics forum*, vol. 30, pp. 603–612, Wiley Online Library, 2011.
- [50] O. V. Kaick, N. Fish, Y. Kleiman, S. Asafi, and D. Cohen-Or, “Shape segmentation by approximate convexity analysis,” *ACM Transactions on Graphics (TOG)*, vol. 34, no. 1, p. 4, 2014.
- [51] A. Golovinskiy and T. Funkhouser, “Randomized cuts for 3d mesh analysis,” in *ACM transactions on graphics (TOG)*, vol. 27, p. 145, ACM, 2008.

- [52] S. Shlafman, A. Tal, and S. Katz, “Metamorphosis of polyhedral surfaces using decomposition,” in *Computer graphics forum*, vol. 21, pp. 219–228, Wiley Online Library, 2002.
- [53] M. Attene, B. Falcidieno, and M. Spagnuolo, “Hierarchical mesh segmentation based on fitting primitives,” *The Visual Computer*, vol. 22, no. 3, pp. 181–193, 2006.
- [54] S. Katz, G. Leifman, and A. Tal, “Mesh segmentation using feature point and core extraction,” *The Visual Computer*, vol. 21, no. 8-10, pp. 649–658, 2005.