B. TECH. PROJECT REPORT

On

Data Augmentation by Generating 3D Adversarial Point Cloud

By
Kolla Krishna Teja

DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE
May 2022

Data Augmentation by Generating 3D Adversarial Point Cloud

A PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the degrees

of
BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

Submitted by:
Krishna Teja Kolla

Guided by:
Dr. Surya Prakash
(Associate Professor, Computer Science and Engineering)

DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE
May 2022

CANDIDATE'S DECLARATION

We hereby declare that the project entitled **"Data Augmentation by Generating 3D Adversarial Point Cloud"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in 'Computer Science and Engineering' completed under the supervision of **Dr. Surya Prakash, Associate Professor, Computer Science and Engineering, IIT Indore** is an authentic work.

Further, I declare that I have not submitted this work for the award of any other degree elsewhere.

K.Krishna Teja Kolla Krishna Teja
26-05-2022

Signature and name of the student with date

CERTIFICATE by BTP Guide(s)

It is certified that the above statement made by the students is correct to the best of my knowledge.

Signature of BTP Guide with dates and their designation

27-May-2022
(Dr. Surya Prakash)
Associate Professor

Preface

This report on **"Data Augmentation by Generating 3D Adversarial Point Cloud"** is submitted in partial fulfillment for the award of the degree of Bachelor of Technology in 'Computer Science and Engineering' completed under the supervision of **Dr. Surya Prakash, Associate Professor, Computer Science and Engineering, IIT Indore.**

Through this report, I have tried to provide a detailed description of the approach, design, and implementation of the Data Augmentation Technique by Generating 3D Adversarial Point Clouds for University of Notre Dame 3D face Dataset.

Kolla Krishna Teja
B.Tech. IV Year
Discipline of Computer Science and Engineering
IIT Indore

Acknowledgements

I wish to thank Dr.Surya Prakash for his kind support, expertise and valuable guidance. He provided a perfect environment for critical thinking and research acumen and was always available for discussions, doubt clearance and guidance at every part of the project. He has constantly motivated me to take the project to its very culmination.

I would also like to acknowledge Akhilesh Mohan Srivastatva (Phd scholar, IIT Indore) for his support and sincere cooperation. He was always available for discussion and doubt clearance.

Kolla Krishna Teja
B.Tech. IV Year
Discipline of Computer Science and Engineering
IIT Indore

Data Augmentation by Generating 3D Adversarial Point Cloud

Abstract

The use of 3D data in many applications has gained traction in recent years due to the availability of inexpensive 3D sensors such as Kinect and LIDAR. Object identification has gained popularity, and it is now done with 3D data to overcome the challenges posed by position, expression, lighting fluctuations, and occlusions in 2D data. Though 3D object recognition is more accurate, the availability of 3D data is limited, causing overfitting in Deep Neural Networks and making processing more difficult, requiring more space and time. To circumvent this issue, the variability of 3D data must be raised by using data augmentation to expand the quantity of available data. Adversarial examples are carefully constructed instances that drive Deep Neural Network models to make incorrect predictions. In this report, we present a way of doing data augmentation by creating 3D adversarial examples. We use random and stratified techniques to generate augmented data for each 3D sample of a subject. The augmented data is trained using PointNet augmented with the siamese model, with the generated augmented data from a sample labeled as a single class. We use the trained model to implement the Adversarial Point Perturbation technique and generate the perturbed data. We then compare the Iterative Closest Point Registration Error between a pair of perturbed samples belonging to the same class and their respective parent augmented samples pair, and between a pair of perturbed samples belonging to different classes and their respective parent augmented samples pair to ensure that the perturbed data created carries the same information as the parent augmented data.

Keywords: Data Augmentation, Adversarial Point Clouds, PointNet, Siamese Network, Adversarial Point Perturbation, ICP

Contents

1 Introduction **6**

2 Literature Review **9**

3 Methodology and Data generation **11**

3.1 Prepossessing 11

3.2 Data Augmentation 12

3.3 Proposed Model 14

3.3.1 Feature extraction with PointNet 14

3.3.2 Training using the Siamese network 16

3.3.3 Adversarial Point Perturbation 17

3.3.4 Verification Using ICP 20

3.4 Novelty in proposed technique 21

4 Results and Discussion **22**

4.1 Database used 23

4.2 Augmentation of databases 23

4.3 Experiment 1 24

4.4 Experiment 2 - UND 3D face Dataset with PointNet 25

4.5 Experiment 3 - ModelNet40 Dataset with PointNet 26

4.5.1 Varying Number of points chosen 27
4.5.2 Varying Both Learning Rate and Number of points chosen . . 27
4.5.3 Varying Number of classes 27
4.5.4 Varying Both Number of classes and Number of points chosen 28
4.6 Performance on UND 3D database 29

5 Conclusion **35**

List of Figures

3.1 Preprocessing Step 12

3.2 Data Augmentation Flow Chart 13

3.3 PointNet Feature Extraction Flow Chart 15

3.4 Siamese Network Training Flow Chart 17

3.5 Adversarial Point Perturbation Flow Chart 20

4.1 A few sample subjects from UND database used in the experimental evaluation of the proposed model 24

4.2 Confusion Matrix of Siamese Network for few subjects from UND database 30

4.3 ROC curve of Siamese Network for few subjects from UND database 30

4.4 Training and Validation Accuracy versus epoch of Siamese Network for few subjects from UND database 31

4.5 Training and Validation loss versus epoch of Siamese Network for few subjects from UND database 31

List of Tables

4.1 Details of 3D objects of UND database used in the experimental evaluation of the proposed model. 23

4.2 Experiment 1 Results 24

4.3 Training and Testing Accuracy values of Experiment 2 for few subjects from UND 3D face Database 26

4.4 Training and Testing Accuracy values Obtained by Varying Number of Points in Experiment 3 for few subjects from UND 3D face Database 27

4.5 Training and Testing Accuracy values Obtained by Varying Both Learning Rate and Number of points chosen in Experiment 3 for few subjects from UND 3D face Database 28

4.6 Training and Testing Accuracy values Obtained by Number of Classes in Experiment 3 for few subjects from UND 3D face Database 28

4.7 Training and Testing Accuracy values Obtained by Varying Both Number of classes and Number of points chosen in Experiment 3 for few subjects from UND 3D face Database 29

4.8 Number of Perturbed samples generated for the number of samples in Dataset for a subject 32

4.9 Intra-Class ICP value of each class for few subjects. NA in above table means Not Applicable. 33

4.10 Average Intra-Class ICP value and Average Inter-Class ICP value for few subjects 33

Chapter 1

Introduction

Many object recognition systems now employ 3D data instead of 2D data, and 3D object recognition systems are becoming increasingly commonly used in areas such as biometrics. In the case of object recognition, while the usage of 2D objects has delivered good results in normal settings, pose, expression, lighting fluctuations, and occlusions impede performance. Because 3D data can capture the object's whole geometric information, the limitations posed by 2D objects can be solved. Similar 3D data applications can be found in a variety of other domains, such as robotic sensing and manipulation.

Although 3D data outperforms 2D data in domains like object identification and biometrics, there are challenges with 3D data training. Because obtaining 3D data from objects takes a long time, there is very little data available for 3D objects. Due to the restricted availability of 3D data, the model learns the features as well as the noise of these limited samples very effectively, resulting in overfitting. To minimize overfitting, the variability of 3D data needs to be improved by extending the size of available data through data augmentation.

Data augmentation is a technique for adding slightly changed copies of current data or newly created synthetic data from current data to expand the amount of data available. When training a machine learning model, it functions as a regularizer and helps reduce overfitting[1].

3D data can be represented in a variety of ways. 3D voxel, 3D

mesh, and point cloud are all typical approaches to represent an object in 3D. A 3D object is represented in 3D voxel representation by discretizing its volume, with a voxel being the unit cubic volume. The pixel, the smallest raster unit, is the 2D equivalent of a voxel. The three-dimensional voxel representation is a highly regularised representation. The bulky character of this representation, on the other hand, renders it computationally and spatially costly. Furthermore, when fine structures must be captured, a high voxel resolution is required, which consumes a lot of memory.A 3D object is represented as a polygon mesh in 3D mesh representation, which is made up of a collection of vertices and polygons that describe the geometry of an object in 3D [2]. This form is also quite huge and demands a lot of memory. Point clouds, on the other hand, are the unprocessed 3D data from most 3D sensors, such as depth cameras and Lidars. The envelope of an object is represented by a point cloud, which is an unordered set of data points in a three-dimensional coordinate system. Rather than being translated into normal 3D representations like 3D voxels or 3D mesh, point clouds can be used directly as an input to any deep neural network. Simple representation, small storage requirements, and invariance to transformations like translation and rotation are all advantages of point clouds.

The PointNet[3] is a deep learning CNN framework that directly inputs point clouds. The PointNet architecture is special in that it can preserve the rotational and translational invariant features of point clouds. By comparing a test sample with a reference sample and predicting whether they belong to the same or distinct subject classes, the Siamese Network[4][5] generates a similarity score.

We directly consume point clouds as 3D object input representations, overcoming the limitations of 3D voxel and 3D mesh. PointNet is chosen as the victim model in [6]. However, because PointNet is ineffective at categorising substantially similar objects, we adopt the victim model provided by [7], in which the PointNet architecture is supplemented with the Siamese network.

[6] provides numerous innovative attack methods for two forms of adversarial attacks on point clouds: adversarial point perturbation

and adversarial point creation, both of which are unnoticed by humans or concealed in the human psyche. For adversarial point perturbation, the existing points are shifted negligibly and the perturbation vector is optimized under the generally used Lp norm constraint. In order to increase the amount of available samples, we apply the adversarial point perturbation technique. We demonstrate that the samples generated carry similar information to the original samples using the Iterative Closest Point(ICP) [8][9] technique.

The rest of the report is organized as follows. Chapter 2 presents Literature Review. Chapter 3 describes our proposed technique. In Chapter 4, the results of the proposed technique's various experiments are reported. Finally, in the last Chapter, the report is concluded.

Chapter 2

Literature Review

Multiple representations of 3D data lead to varied learning methodologies. Deep learning approaches on 3D data frequently employ volumetric CNNs. Voxelized forms [10][11][12] are one type of input to 3D CNNs. Volumetric representation, on the other hand, is constrained by its empty data spaces, resolution due to data sparsity, and 3D convolution computational complexity. A high level of voxel resolution is required to record great facial shapes, which consumes a lot of memory. Multiview CNNs [11][13] attempted to transform 3D shapes into 2D images and then categorise them using 2D convolution nets. With well-engineered image CNNs, this line of approaches has reached overwhelming performance on shape classification and retrieval tasks [14]. However, extending them to additional 3D tasks like point categorization and shape completion is not straightforward.

The majority of the current features in point clouds are handmade for specific tasks. Certain statistical properties of points are often encoded in Point features. Point features are designed to be invariant to certain transformations, which are typically classified as intrinsic [15][16][17] or extrinsic [18][19] . Point features are also classified as local and global. Finding the best feature combination for a certain task is not simple.Point clouds consist of unordered points with varying cardinality. This makes it difficult for neural networks to consume Point Clouds. Qi et al. [3] proposed a novel network called PointNet, which is commonly used for deep point cloud processing, to address the

problem of point clouds. PointNet and its derivatives [20][21] simplify learning by reducing the unordered and variable cardinality input to a fixed-length global feature vector using a symmetric function termed max pooling.

Face recognition has also been done with Deep Siamese Neural Networks [5]. Hayale et al.[4] present an analogous technique with a supervised loss function that favours inter-class variations while limiting intra-class variations by increasing the distance between the features for distinct classes. Joshi et al. [22] employed the Siamese Network to compare scanned facial photos to digital images.

Szegedy et al. [23] were the first to bring out that neural networks and other machine learning models were sensitive to skillfully crafted adversarial perturbation. An adversarial example that appears to be identical to the original data can easily trick neural networks. Several works have been proposed to increase adversarial attack performance [24][25] and search for suitable adversarial attack defences [26][27]. The state-of-the-art attack algorithm, optimization based attack [28], employs optimization to discover a near-optimal adversarial solution for 2D data by defining an objective loss function that quantifies both attack effectiveness and perturbation magnitude.

Image transformation methods such as translation, rotation, scaling, and mirroring are commonly utilised in data augmentation strategies to tackle the problem of a shortage of large datasets [29][30][31]. Training pictures for color-based algorithms have been augmented by a process of modifying the hue-channel for coloured data in [30]. Ge et al. [32] presented a 3D transformation for data augmentation in depth-based approaches. To synthesise the 3D data, the transformation requires randomly rotating and stretching the 3D point cloud. Hinterstoisser et al. [33] used training samples produced from 3D models to generate enhanced data. Because the synthetic data does not have a distribution similar to that of real data, the method is limited by over-fitting and so requires a well constructed training process.

Chapter 3

Methodology and Data generation

The proposed approach uses a general solution for creating 3D object augmented data. As an input, the approach leverages the point cloud representation of 3D objects. Each 3D object's point cloud contains a large number of points. We employ numerous subsets of each point cloud to feed into the model in order to make it time and space-efficient. This expands the capacity of our database, which was previously restricted to only one sample per class, preventing overfitting.

The technique initially employs two networks: the PointNet architecture for feature extraction and the Siamese Network for training. We extract features from the second-last dense layer of the architecture after the PointNet model has been trained on a subject's training samples. The Siamese network is then trained using the extracted features. After the training is completed, we use the trained models in the adversarial point perturbation step. In Adversarial Point Perturbation, we shift existing points of a sample negligibly and create a new sample.

3.1 Prepossessing

3D noise can have an impact on the object classification model's performance. To reduce variations in poses and 3D noise, 3D images must

be preprocessed. To do this, frontalization and denoising techniques are widely utilised. In addition, the 3D objects are frequently aligned to a base reference image to ensure that the database is consistent with some ground truth. In contrast, the PointNet design is invariant to geometric transformations such as rotation and translation[3], eliminating the need to frontalize the objects.

The point clouds of 3D objects may become contaminated with spikes as a result of sensor noise, affecting the feature extraction process. As a result, before employing object 3D data for feature extraction, this must be addressed. We use a standard spike removal technique with a moving averaging filter to denoise the images. This method involves moving a sliding window across the object and determining the offset along the Z-coordinates. The centre of the window is additionally translated to the mean offset if the resulting value is higher than a threshold. The 3D scan of the UND dataset is denoised with this approach, which limits the spikes to the chosen threshold. The Preprocessing Step is depicted in Figure 3.1.

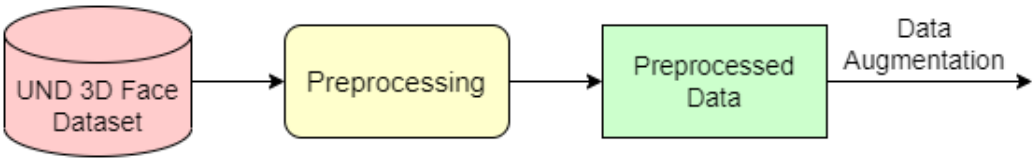


Figure 3.1: Preprocessing Step

3.2 Data Augmentation

Because obtaining 3D data from objects takes time, there is frequently only a limited amount of data available for 3D objects. There are very few examples per subject in the University of Notre Dame (UND) database that we used to train our model. The number of samples per subject varies between 2 to 9. During the training phase, each sample is given a unique label. As a result, we will only have one sample per class for both training and testing. With such a minimal amount of data per class, the model will not be successfully trained or

extract features. When only a few samples are provided, the chosen model learns the noise of these few samples so well that it hinders the model's assessment of new data. To minimize overfitting, we employ data augmentation to increase the variability of the 3D data. As a result, augmentation is required to provide enough examples to train the model. The augmented database is created using the Random Point Cloud Augmentation (Type II) technique [7], and the subsamples are generated using both Random and Stratified Sampling techniques [34].

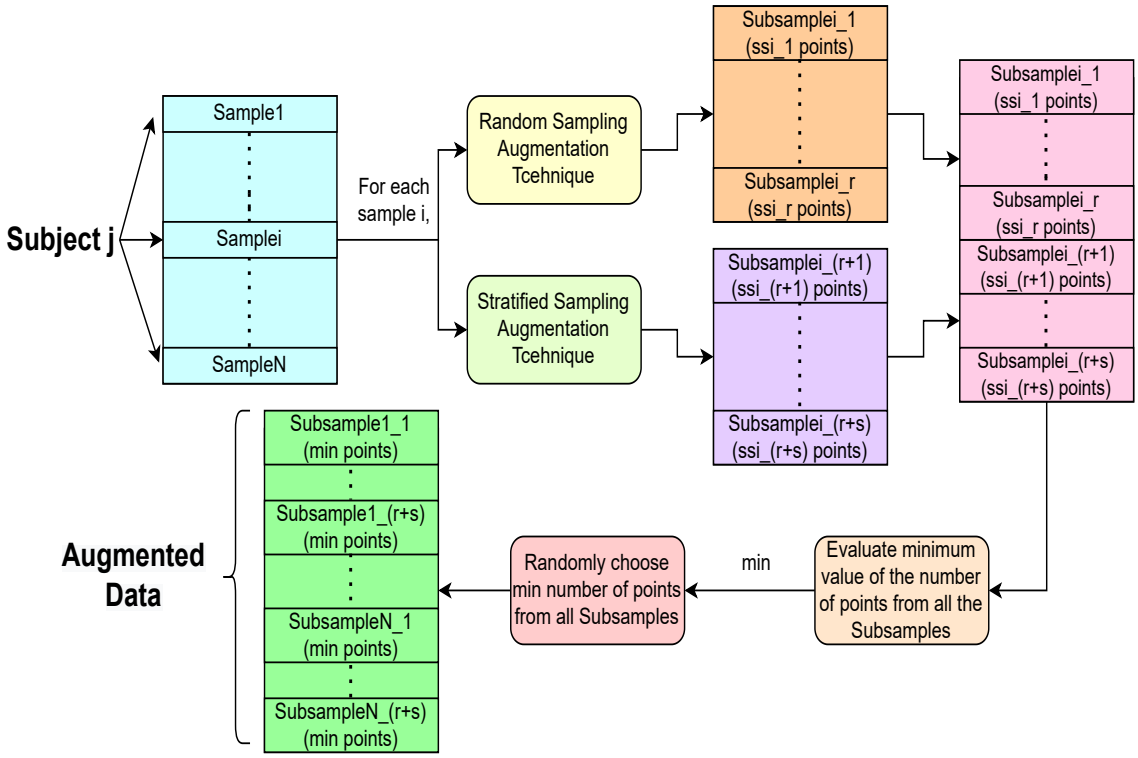


Figure 3.2: Data Augmentation Flow Chart

We produce a certain number of subsamples for each sample using random sampling and stratified sampling techniques, with each subsample having a varied number of points. We make all the subsamples generated from a single sample have the same number of points by first evaluating the value of the minimum number of points among the subsamples and then selecting the evaluated number of points at random from each of the subsamples. For the following phase of the procedure, we'll use these subsamples in place of the original sample. The Data Augmentation Flow Chart is shown in Figure 3.2.

3.3 Proposed Model

We begin by dividing the complete data into train and test sets. We also make the labels for the train and test sets. We train PointNet with the train set, and train labels and weights are saved in a file. By passing train and test sets as input, the Trained PointNet model extracts train features and test features from the intermediate layer. Then, using train labels, genuine and imposter pairs are produced from the train features. The generated pairs are then used for Siamese network training and the weights are saved in a file.

We initially restore the trained PointNet model and Siamese network from their respective weight files in the perturbation step. In predicting the label of the input, the restored PointNet model and Siamese network are useful. The method starts by determining how much perturbation is needed to minimize the optimization equation. To create the new perturbed sample, the amount of perturbation is simply added to the initial sample.

3.3.1 Feature extraction with PointNet

Point clouds have an unordered representation, unlike 2D pixel arrays in images or 3D voxel arrays in 3D objects. Many researchers convert a point cloud to various forms by taking multi-view projections onto 2D space or quantizing it to 3D voxels in order to exploit current approaches based around (2D and 3D) convolutions. Multi-view projections, on the other hand, generate massive volumes of data, and quantizing the 3D structure can introduce variance due to natural artifacts.

Raw point cloud data is fed into PointNet. The input points are first transformed using the input transform, and then each of the n input points is mapped from 3 dimensions to 64 dimensions using a shared multi-layer perceptron. The 64-dimensional data is then transformed using a feature transform. Each of the n points is mapped from 64 to 1024 dimensions in the next layer. We now use max-pooling to

generate a global feature vector in $\mathbb{R}^{1024}$. Finally, the global feature vector is mapped to k output classification scores using a three-layer fully connected network.

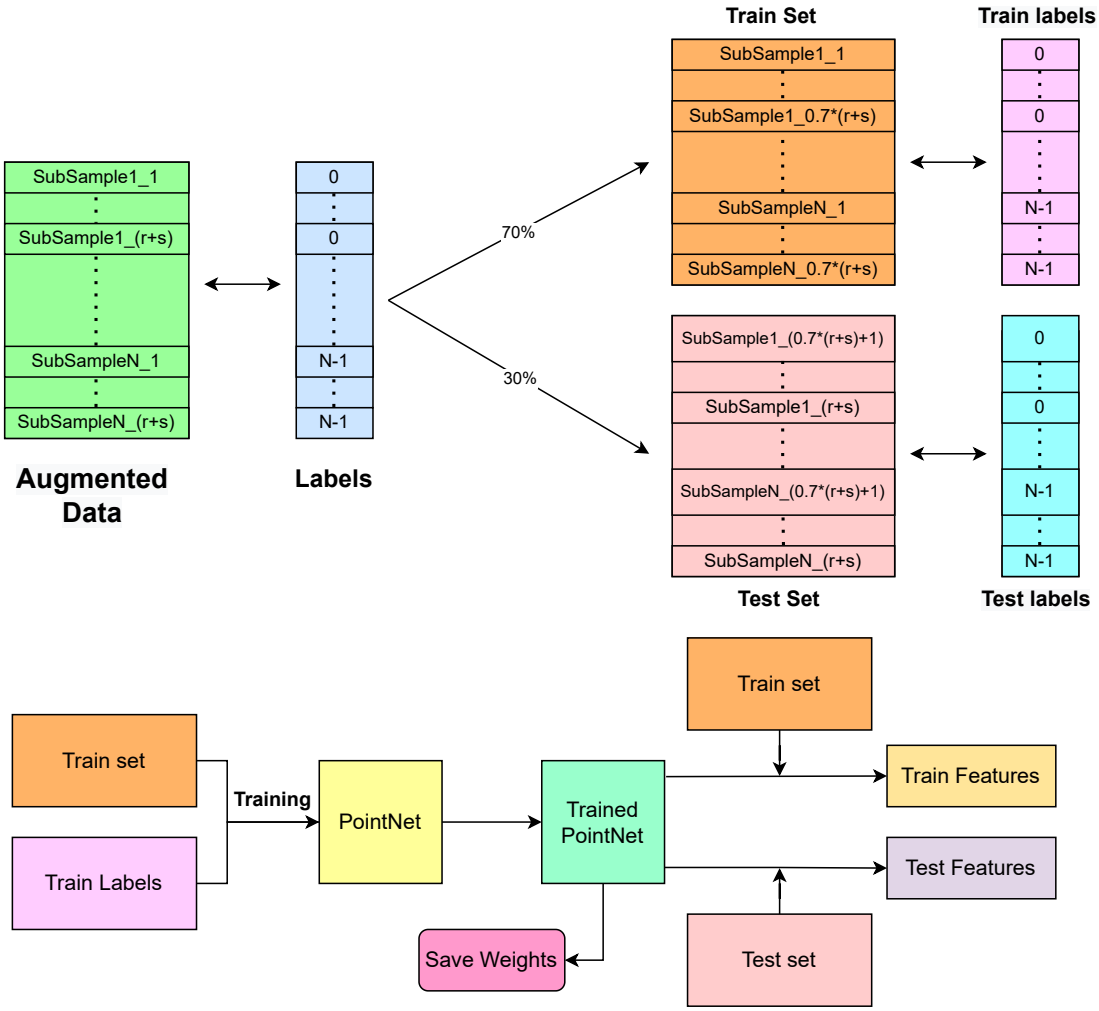


Figure 3.3: PointNet Feature Extraction Flow Chart

In our technique, we begin by creating labels for the data received during the data augmentation step. The same label is applied to all subsamples created from a single sample. If a subject has N samples, for example, we will have N different classes during the training phase, namely $0, 1, 2, ..., (N-1)$ and all subsamples created from a sample i are given the label $(i-1)$. The data was then divided into two sets: the train set and the test set. The PointNet model is trained using subsamples from the train set and their appropriate labels from train labels. The trained model's weights are preserved and used to restore

the model in the perturbation step.

In general, PointNet outputs the input sample's class as the last layer's output. According to [7], a shortcoming of the PointNet architecture is the poor categorization of samples belonging to the same object class, such as faces. Rather than solving a 1:n classification problem for objects in the same class, we use the PointNet architecture to create feature vectors for the input point clouds. The feature vectors for the input data are output by the architecture's second-last dense layer, which has 256 neurons. The feature vectors for the train and test sets are extracted and named train set features and test set features, respectively. The PointNet Feature Extraction Flow Chart is shown in Figure 3.3. The Siamese network is then trained using the train set features and train labels. Finally, we use the test set to put our trained Siamese network to the test and generate a classification report.

3.3.2 Training using the Siamese network

A neural network, in traditional terms, collects data and learns to predict several classes. Deep neural networks require a large quantity of data to train. Siamese networks, on the other hand, use a similarity score to decide whether the reference and test inputs belong to the same or different classes based on a threshold value. It uses the similarity score to train itself. The similarity score ranges from 0 and 1, with 0 indicating no similarity and 1 indicating total similarity. The following are the key benefits of Siamese network: more resistant to the problem of class imbalance - With One-shot learning, a few samples per class are enough for Siamese networks to recognize those samples. Learning from Semantic Similarity - Siamese focuses on learning embeddings (features) that group together similar classes.

The Siamese network determines the relationship between the pairs of features using four functions: absolute difference, addition, square of the absolute difference, and multiplication between the pair of features. These four operations' outputs are combined and transmitted

to the convolutional layers to be trained. We require two sets of pairs from the features acquired from PointNet to train the Siamese network: impostor pairs that contain features from subsamples from a different class and genuine pairs that contain features from subsamples from the same class. These feature pairs are used to train the network to determine if they are genuine or impostor pairs. To make the training more robust, we build all feasible genuine and imposter pairs by combining all features within a class for all classes and combinations of features in each class with every other class. A similarity score or probability is the output of the Siamese network. We employ a scoring threshold to determine whether the pair is genuine or not.

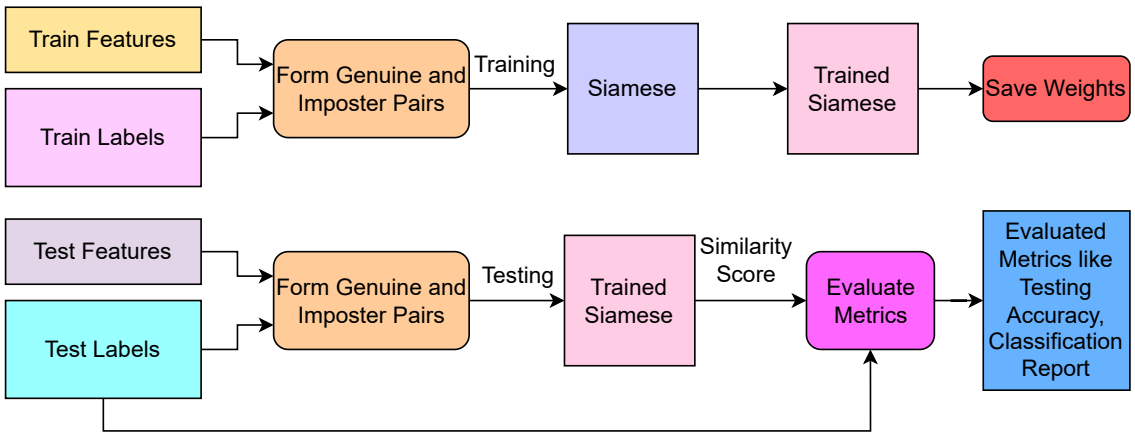


Figure 3.4: Siamese Network Training Flow Chart

In our technique, Initially, train features are utilised to train the Siamese Network by using train labels to build Genuine and Imposter pairs. The trained model's weights are preserved and used to restore the model in the perturbation step. Then, using test labels, we put our model to the test by creating Imposter and Genuine pairs of test features. The Siamese Network Training Flow Chart is depicted in Figure 3.4.

3.3.3 Adversarial Point Perturbation

For creating perturbed data, we leverage targeted attacks on 3D point cloud classification algorithms in our method. Targeted attacks attempt to deceive a 3D deep model into classifying an adversarial

example as a specific target class. As mentioned in [6], Formally, for a classification model $G : A \rightarrow B$, which maps an input $a \in A \subset \mathbb{R}^{n\times 3}$ to its respective class label $b \in B \subset \mathbb{Z}$ (i.e., $G(a) = b$), $t \in B$ is a t is a fraudulent target class used by an adversary. The purpose of the attack is to discover a legitimate input $a' \in \mathbb{R}^{n'\times 3}$ using a perturbation metric $D : \mathbb{R}^{n\times 3} \times \mathbb{R}^{n'\times 3} \rightarrow \mathbb{R}$ which:

$$\text{minimize } D(a, a'), \text{ such that } G(a') = t$$

This problem can be solved by expressing it as an appropriate optimization instance that can be solved using known optimization methods. Formulating the optimization instance as an objective function is one such method. However, The objective function formulation is difficult to solve because $G(a') = t$ is highly non-linear (the classification model is not a straightforward linear function). So directly solving this problem is difficult. According to [28], we can express the constraint in a different form, as an objective function 'f' such that when $G(a') = t$ is satisfied $f(a') \leq 0$ is also satisfied. Conceptually, the objective function f tells us, how close we are getting to being classified as t. In [28], the Authors evaluated 7 different objective functions (all of them are loss functions) 'f' and selects the best one among them that is given by:

$$f(a') = (\max_{i\neq t}(Z(a')_i) - (Z(a')_t))^+ \tag{3.1}$$

Where the adversarial loss function, $f(a')$, assesses the likelihood of a successful attack, where $Z(a) = z$ is the output of all layers except the softmax (Input to softmax layer, so z are the logits), and G is a full neural network including the softmax function, $G(a) = \text{softmax}(Z(a))$ $= b$ and $(r)^+$ represents $\max(r, 0)$ Therefore, the problem is reformulated as a gradient-based optimization technique:

$$\text{minimize } f(x') + \lambda * D(a, a')$$

where the adversarial loss function, $f(a')$, assesses the likelihood of a successful attack and taken from equation 3.1. By optimizing over

this new formulation, we aim to search for adversarial examples with the least 3D perturbation.

We change existing points by moving their XYZ positions with adversarial jitters in adversarial point perturbation [6], such that a point $a_i \in \mathbb{R}^3$ in the point cloud a becomes $a_i^{'} = a_i + \delta_i$, for $i = 1, ..., n$ where $\delta_i \in \mathbb{R}^3$ is the perturbation to the i^{th} point. We optimize the perturbation vector or amount of adversarial jitter under the commonly used L_p norm constraint.

The Adversarial Point Perturbation technique is used in this work for each subject by setting the target class as one of the classes within the same subject. As a result, the targeted adversarial attack aims to change a subsample so that it is classified as the target class within the same subject by the model. To get the value of λ, we use a binary search method. We perform specific operations in order for numerous iterations for each step of binary search. The operations are:

1. By optimising the objective function, modify the value of the perturbation vector or the amount of adversarial jitters.
2. The perturbed point cloud is created by adding the updated perturbation vector to the input point cloud.
3. The saved weights are used to reconstruct the PointNet model. One randomly picked point cloud from the train set for each class, as well as the perturbed point cloud, are provided as input to the restored PointNet model.
4. The Siamese network is restored using the weights saved. The perturbed point cloud features are paired with the features from each class's point cloud and supplied into the restored Siamese network. The pair with the highest similarity score belongs to the same class, i.e. the perturbed point cloud belongs to the same class as the point cloud with the highest similarity score.
5. The predicted class value is used for step 1 execution.

Despite the fact that the points in the subsample are being updated, the created subsample after modification serves as a fresh subsample

of the same subject because the target class is one of the classes within the same subject. Figure 3.5 depicts the Adversarial Point Perturbation Flow Chart.

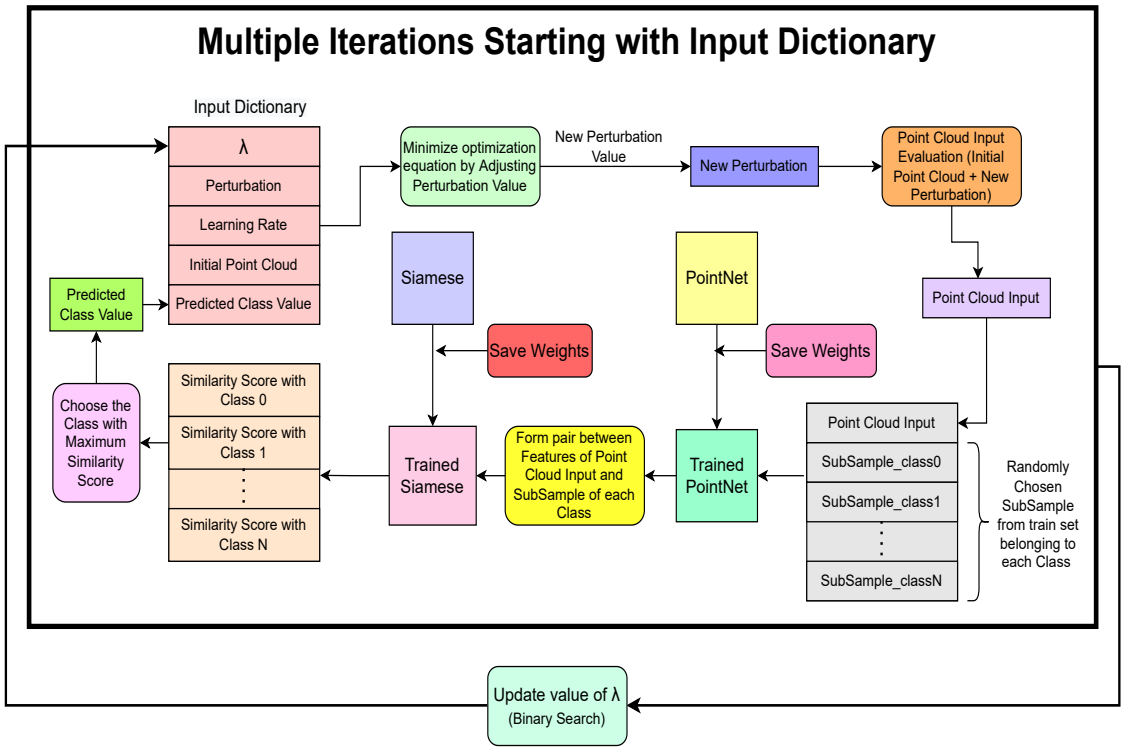


Figure 3.5: Adversarial Point Perturbation Flow Chart

3.3.4 Verification Using ICP

The algorithm iterative closest point (ICP) [8][9] is used to reduce the difference between two point clouds. In the Iterative Closest Point algorithm, one point cloud, the reference, is kept unchanged while the other, the source, is altered to best match the reference. The Iterative Closest Point algorithm iteratively revises the transformation, which is a combination of rotation and translation operations required to minimise an error metric, such as the sum of squared differences between the matched pairs' coordinates.

1. **Intra-Class**

To begin, Choose two subsamples from perturbed data and evaluate the ICP registration between these two perturbed subsamples for each class of data obtained by perturbation. Similarly, evaluate the ICP registration error between the two parent augmented subsamples from which the two perturbed subsamples are created. The ICP registration error values of the parent augmented and perturbed subsamples are then compared.

2. **Inter-Class**

   To begin, select one subsample per class from the perturbed data to create a set of subsamples. Then, for all possible permutations of pairs in the set, evaluate the ICP registration error values. Simultaneously, evaluate the ICP registration error values of the corresponding parent augmented subsample pairs for each feasible permutation of pairs from the set. Compare the ICP registration error values from the perturbed and parent augmented data pairs.

3.4 Novelty in proposed technique

Because obtaining 3D data from objects takes a long time, there is very little data available for 3D objects. Due to the limited availability of 3D data, the model effectively learns the Features as well as the noise of these few samples, resulting in overfitting. Data augmentation is the only known solution to the problem of overfitting caused by small sample sizes per person. We have suggested a novel Data Augmentation technique based on Adversarial Point Perturbation.

Chapter 4

Results and Discussion

We now use UND 3D face data to demonstrate the effectiveness of the proposed approach. We use data augmentation to improve feature extraction and avoid overfitting in the UND 3D face database. The larger database after data augmentation is split into two sets at random: 70% for training and 30% for testing. The model for the training phase was first chosen as PointNet. However, following a series of experiments, it was observed that when the input samples are very similar, the PointNet model classification accuracy is quite low. Then we use the PointNet model augmented with the Siamese network as the model to classify highly similar objects. We perform the PointNet training using the training data and the weights of the trained model are saved, and the training data and testing data are supplied as inputs to the trained model to extract the features. The Siamese network is then trained using the extracted features, and the weights of the trained model are saved. We assess classification performance using the Receiver Operating Characteristics (ROC) curve and verification accuracy. The perturbed data is then generated by performing adversarial point perturbation on the database. Iterative Closest Point Registration Error is used to assess the quality of the perturbed data.

4.1 Database used

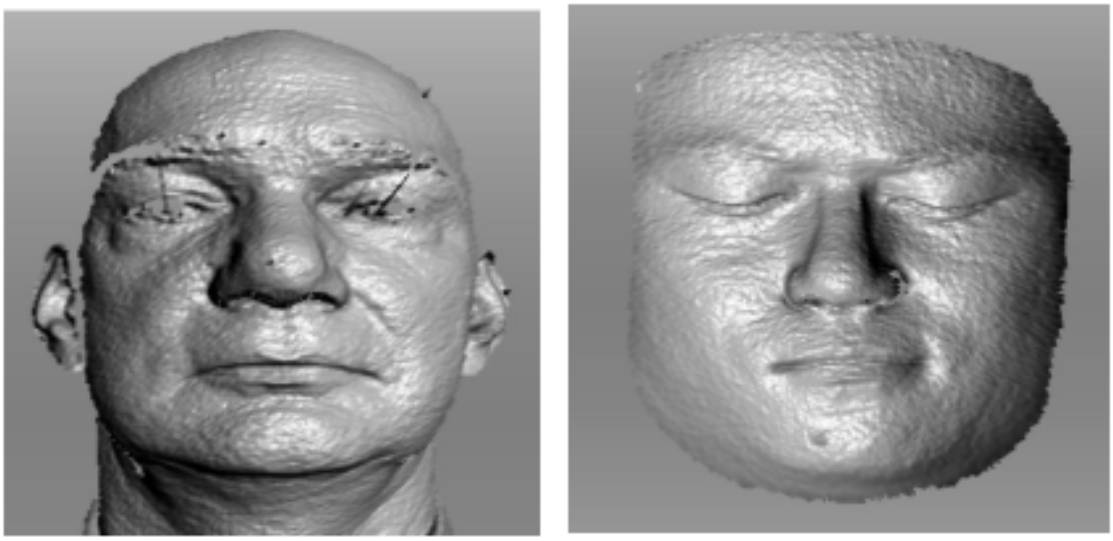
We use 3D face data from the University of Notre Dame (UND) database to evaluate our model. It should be noted that the 3D face recognition challenge was handled solely using the database's spatial coordinate information. The UND database contains 277 subjects with 953 aligned 3D facial images (ND-collection D). A Minolta Vivid 900 3D range scanner was used to take these photos. These facial scans have a lot of noise in the form of spikes. As a result, we're denoising the 3D scans with spike removal. Before being used in the experiment, the 3D face scans are pre-processed as described in Section 3.1. Figure 4.1 shows a few example scans from the UND database after prepocessing. The summary of UND 3D database is provided in Table 4.1.

Dataset	# of Subjects	# of 3D object samples
University of Notre Dame	277	953

Table 4.1: Details of 3D objects of UND database used in the experimental evaluation of the proposed model.

4.2 Augmentation of databases

We augment the databases to enhance the sample size because the number of samples per subject is rather low in the available dataset. We use the Random point cloud augmentation (Type II) technique to create the augmented database, and both Random and Stratified Sampling approaches to generate the subsamples, as explained in Section 3.2. We create 30 subsamples from each sample using Random Sampling and 30 subsamples from Stratified Sampling approaches. The number of points in each of the 60 subsamples is different. By first assessing the value of the minimum number of points among the subsamples and then randomly selecting the evaluated minimum number



(a) (b)

Figure 4.1: A few sample subjects from UND database used in the experimental evaluation of the proposed model

of points from each of the subsamples, we make the number of points the same in all the 60 Subsamples generated from a single sample.

4.3 Experiment 1

To verify the promised outcomes in [3], an initial experiment was conducted on the ModelNet40 dataset. Table 4.2 displays the acquired results.

Mean Training Loss	Training Accuracy	Testing Accuracy
0.1818662	93.59356%	86.7289%

Table 4.2: Experiment 1 Results

4.4 Experiment 2 - UND 3D face Dataset with PointNet

This Experiment is performed on UND 3D face Dataset. The training, testing happens individually for each subject in the UND 3D face dataset. For each subject, we generate augmented data as mentioned in 4.2. All the generated 60 Augmented samples from a single sample are given same label during the Process. The Number of labels during the process(in a single subject) is the number of samples available for the respective subject in UND dataset. For a subject with N samples, during the process, value of the number of Classes is N and the number of Augmented samples available per Class are 60. The Number of points chosen is the minimum number of points available in all the Augmented Samples of a particular Subject. We split the data(after data augmentation) 70 : 30 ratio to create train and test sets i.e, Among the 60 Augmented Samples available per class, 42 Augmented Samples are part of Training data and 18 Augmented Samples are part of Testing data. The Batch size is chosen as the Greatest common divisor of the Number of training, testing samples So that no samples are leftover during training and testing.

The UND 3D face Dataset is used in this experiment. Each subject in the UND 3D face dataset is separately trained and tested. We generate augmented data for each participant as described in 4.2. During the Process, all 60 Augmented samples generated from a single sample are given the same label. The number of labels used during the process (in a single subject) is equal to the number of samples in the UND dataset for that subject. For a subject with N samples, during the process, value of the number of Classes is N and the number of Augmented samples available per Class are 60. The chosen number of points is the smallest number of points accessible across all Augmented Samples for a given Subject. We divided the data (after data augmentation) into train and test sets in a 70 : 30 ratio, i.e., 42 Augmented Samples are part of Training data and 18 Augmented Samples are part of Testing data, out of 60 Augmented Samples available per class. So that no samples are left over during training and testing,

the Batch size is determined as the greatest common divisor of the number of training and testing samples.

Subject	# of Training samples	# of Testing samples	# of Points Chosen	Batch Size	Learning Rate	Training Accuracy	Testing Accuracy
1	210	90	18500	30	0.01	37.1429%	20%
2	126	54	32601	18	0.01	38.0952%	55.56%
					0.001	66.67%	53.7037%
3	168	72	17104	24	0.01	33.9286%	25%
					0.001	72.0238%	25%

Table 4.3: Training and Testing Accuracy values of Experiment 2 for few subjects from UND 3D face Database

We can deduce from table 4.3 that, while the Training and Testing Accuracy values are both quite low, changing the learning rate from 0.01 to 0.001 greatly increased the Training Accuracy. The number of training and testing samples has decreased as the number of points chosen has increased, while the value of training accuracy has increased (Increasing points has increased the number of points available for training).

4.5 Experiment 3 - ModelNet40 Dataset with PointNet

This experiment is performed on ModelNet40 Dataset under similar conditions as Experiment 2. Each object in the ModelNet40 dataset is trained and tested individually. The number of points chosen and the number of classes are both variables that we may control. The number of samples is determined by the number of points picked. We choose samples with more points than the number of points chosen from all the samples available for the specific object based on the value of Number of points chosen.

4.5.1 Varying Number of points chosen

The number of classes and the learning rate have been set to 4 and 0.01, respectively. We may deduce from table 4.4 that, despite the low Training and Testing Accuracy values, increasing the Number of Points chosen reduced the number of Training and Testing Samples while increasing the Training Accuracy Value (Increasing points has increased the number of points available for training).

Subject	# of Points Chosen	# of Training samples	# of Testing samples	Training Accuracy	Testing Accuracy
Car	1024	196	96	32.7778%	25%
	2048	192	96	32.8125%	25%
Vase	1024	364	72	30.2198%	34.72%
	2048	260	52	32.7778%	25%
Guitar	1024	148	92	35.4167%	26.0870%
	2048	144	84	35.8108%	25%

Table 4.4: Training and Testing Accuracy values Obtained by Varying Number of Points in Experiment 3 for few subjects from UND 3D face Database

4.5.2 Varying Both Learning Rate and Number of points chosen

The number of classes is fixed to four. Although the Training and Testing Accuracy values are relatively low, we may conclude from table 4.5 that changing the learning rate from 0.01 to 0.001 while fixing the Number of Points chosen has increased the value of Training Accuracy.

4.5.3 Varying Number of classes

The number of points chosen has been set to 1024. We may deduce from table 4.6 that, despite the fact that the Training and Testing Accuracy values are both quite low, increasing the number of classes

Subject	# of Points Chosen	# of Training samples	# of Testing samples	Learning Rate	Training Accuracy	Testing Accuracy
Cup	1024	40	12	0.01	40%	25%
				0.001	45%	25%
	2048	28	8	0.01	42.8571%	25%
				0.001	42.8571%	25%

Table 4.5: Training and Testing Accuracy values Obtained by Varying Both Learning Rate and Number of points chosen in Experiment 3 for few subjects from UND 3D face Database

while keeping the number of points fixed has drastically reduced the Training Accuracy value. Because the number of classes increased, the amount of data available for training and testing each class decreased dramatically.

Subject	# of Classes	# of Training samples	# of Testing samples	Training Accuracy	Testing Accuracy
Guitar	3	150	93	43.2432%	34.7826%
	4	148	92	35.8108%	26.087%
	8	144	88	22.2222%	18.18%

Table 4.6: Training and Testing Accuracy values Obtained by Number of Classes in Experiment 3 for few subjects from UND 3D face Database

4.5.4 Varying Both Number of classes and Number of points chosen

The Batch Size has been set at 5. We may deduce from table 4.7 that, despite the fact that the Training and Testing Accuracy values are both quite low, increasing the number of classes while keeping the number of points fixed has drastically reduced the Training Accuracy value. Because the number of classes increased, the amount of data available for training and testing each class decreased dramatically. Increasing the value of the Number of Points chosen has also increased the Training Accuracy for a fixed number of classes.

Subject	# of Classes	# of Points Chosen	# of Training samples	# of Testing samples	Training Accuracy	Testing Accuracy
Tv Stand	3	1024	138	42	42.9630%	45%
		2048	93	30	50%	40%
	6	1024	138	42	25.1852%	25%
		2048	90	30	33.33%	23.33%

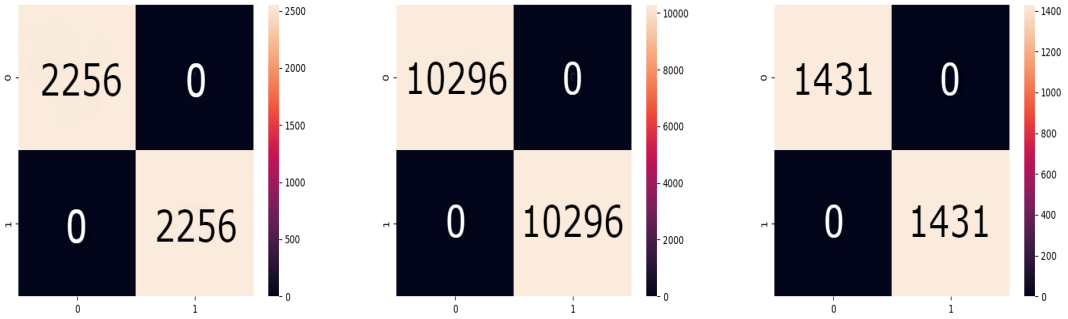
Table 4.7: Training and Testing Accuracy values Obtained by Varying Both Number of classes and Number of points chosen in Experiment 3 for few subjects from UND 3D face Database

4.6 Performance on UND 3D database

we create the labels corresponding to the data obtained from the data augmentation step by assigning the same label for the 60 subsamples created from the same sample and different labels for the subsamples created from different samples. We split the extended data (after data augmentation) and labels into a 70 : 30 ratio for all the samples of a subject to create train, test sets, and train, test labels, i.e., for each sample, the 60 Subsamples generated from the data augmentation step are split into a 70 : 30 ratio, i.e., the samples from 1 to 42 will be part of the train set, and the samples from 43 to 60 will be part of test set. Similarly, we split the labels into train and test labels in a 70 : 30 ratio.

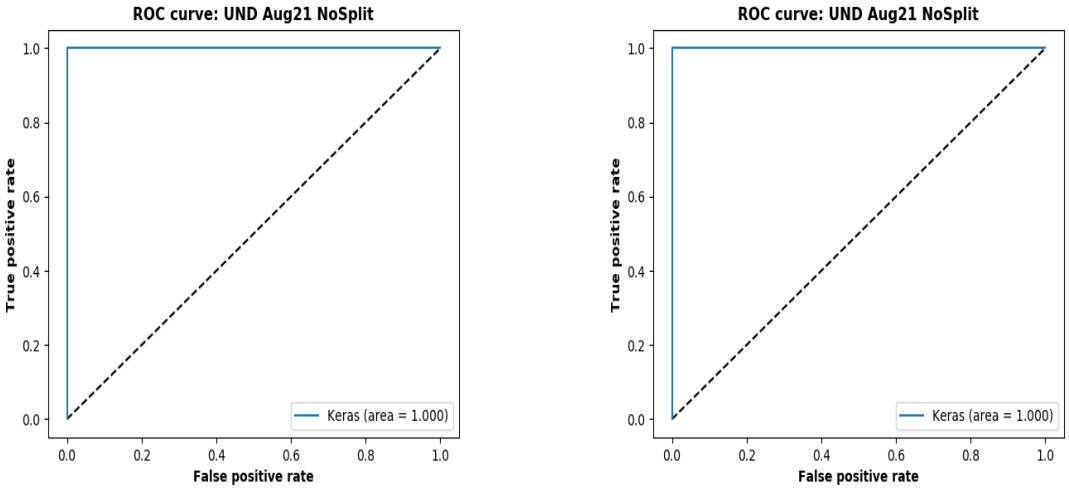
Train set and train labels are used to train the PointNet model. The trained PointNet model's weights are saved in order to restore the model in subsequent steps. To test the model, the test set is fed as input into PointNet. Features are extracted from the trained PointNet model by passing the train set and test set as input. With the use of train labels, the extracted features from the train set are utilised to train the Siamese network by establishing genuine and imposter pairs among train features. The weights of the trained Siamese network are preserved in case the model needs to be restored in the future. With the use of test labels, the features retrieved from the test set are utilised to evaluate the Siamese network by constructing genuine and imposter pairs among test features. To test the model's capacity to distinguish genuine and impostor pairs from an unknown collection

of samples, the pairs are built from the test set's features (test-test pairs).



(a) (b) (c)

Figure 4.2: Confusion Matrix of Siamese Network for few subjects from UND database

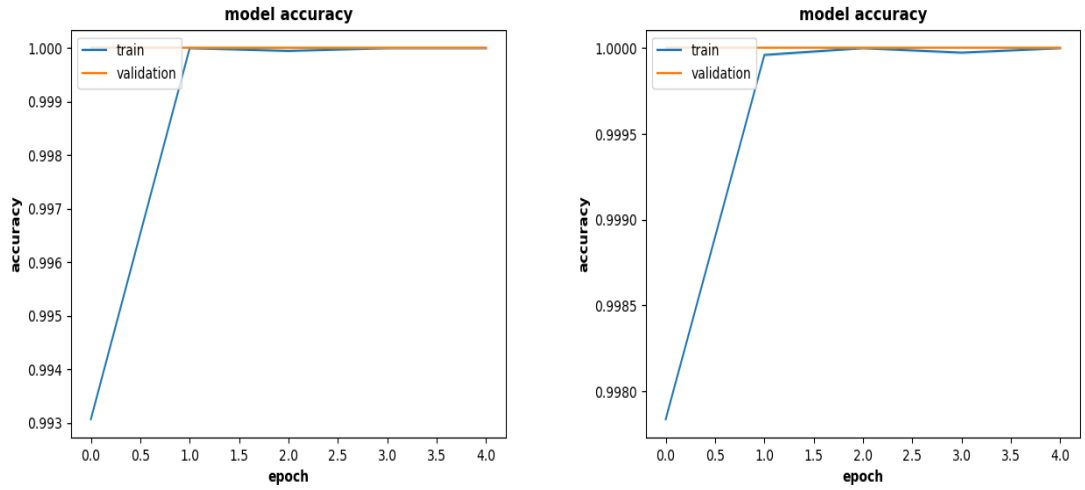


(a) (b)

Figure 4.3: ROC curve of Siamese Network for few subjects from UND database

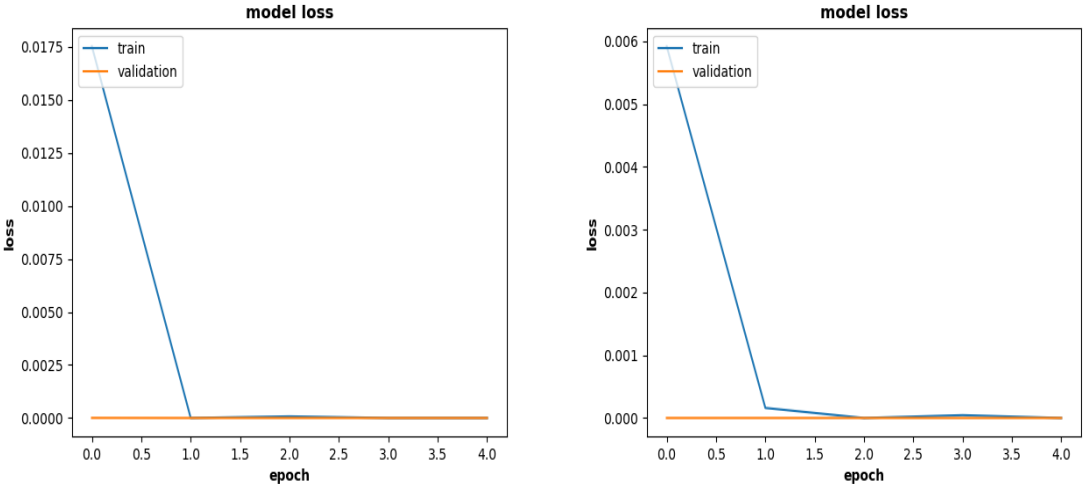
The Confusion matrix, Classification Report, and Receiver Operating Characteristics (ROC) curve are used to assess the Siamese network. For a few subjects, the confusion matrix, Receiver Operating Characteristics (ROC) curve, training, validation accuracy vs epoch, and training, validation loss vs epoch are presented in Figures 4.2, 4.3, 4.4, 4.5 respectively.

The database is then subjected to Adversarial Point Perturbation.



(a) (b)

Figure 4.4: Training and Validation Accuracy versus epoch of Siamese Network for few subjects from UND database



(a) (b)

Figure 4.5: Training and Validation loss versus epoch of Siamese Network for few subjects from UND database

We use the saved weights to restore the trained PointNet and Siamese models. The label of an input sample is required during the Adversarial Point Perturbation. For each class, we randomly select one sample from the train set to evaluate the label. To obtain features, the randomly picked samples and the input sample are fed into a restored PointNet. Each of the features obtained from chosen samples is paired with the input sample features. Features of chosen samples are already seen by the Siamese model during training. As a result, we're creating pairs with one unknown data (Input sample features) and one known data (chosen sample features) to feed into the restored Siamese model. The input sample's label is determined by the pair with the highest similarity score. To generate the perturbed data, we use the procedure described in section 3.3.3.

For a subject with N samples, we generate $60 \times N$ Augmented subsamples using Random and Stratified Augmentation techniques. The $60 \times N$ Augmented subsamples become the input subsamples (original / parent subsamples) for the proposed model. After applying the adversarial point perturbation, perturbed data is generated only for $60 \times (N-1)$ original samples i.e, for all the original subsamples not belonging to the target class, and no perturbed subsamples are generated for the original subsamples belonging to target class. Table 4.8 shows the number of Perturbed subsamples generated for the number of Augmented subsamples.

# of samples in Dataset for a subject	# of Augmented subsamples	# of perturbed subsamples generated
3	180	120
4	240	180
5	300	240
6	360	300
7	420	360
8	480	420
9	540	480

Table 4.8: Number of Perturbed samples generated for the number of samples in Dataset for a subject

The perturbed data is evaluated with both inter-class ICP registration error and intra-class ICP registration error as mentioned in section 3.3.4.

# of Samples	Average ICP of						
	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
8	0.1604164	0.1876751	0.1568064	0.1928608	0.348255	0.045206	0.4275612
3	0.07482	0.0740165	NA	NA	NA	NA	NA
4	0.1645941	0.2171263	0.1126933	NA	NA	NA	NA

Table 4.9: Intra-Class ICP value of each class for few subjects. NA in above table means Not Applicable.

In Intra Class ICP registration error, under each class, for each perturbed sample i, we randomly choose one perturbed sample among $i+1, i+2, ..., 60$ and form the pair to evaluate the ICP registration error. The respective parent subsample pair of the perturbed subsample pair is chosen to evaluate the ICP registration error. Then we compare the corresponding ICP registration error values of perturbed subsample pair and original subsample pair. So, for each class, we have 59 ICP registration error values since last subsample cant form a pair. Figure 4.9 depicts the Intra-Class ICP value of each class for few subjects.

Subject	# of Samples	Average of Intra-Class ICP value	Average of Inter-Class ICP value
1	8	0.3583523286	0.7524851
2	9	0.2381803875	0.3986797
3	4	0.2154470333	0.9308781
4	5	0.101544	0.8268065
5	6	0.1640386352	0.7432993

Table 4.10: Average Intra-Class ICP value and Average Inter-Class ICP value for few subjects

In Inter Class ICP registration error, for each class, we randomly choose one perturbed sample. Among the randomly chosen perturbed samples from all classes, we form all possible combinations of pairs and

evaluate the ICP registration error. The respective parent sample pair of the perturbed sample pair is chosen to evaluate the ICP registration error. Then we compare the corresponding ICP registration error values of perturbed sample pair and original sample pair. We repeat this process multiple times. In experiments, For an N class Subject, we form a total of $60 \times (N-1)$ pairs. Table 4.10 depicts the Average Intra-Class ICP value and Average Inter-Class ICP value for few subjects

Chapter 5

Conclusion

When only a few examples of 3D objects are provided, the model learns the features as well as the noise of these few samples very effectively, resulting in overfitting. Data augmentation, or enhancing the variety of available samples, can help to minimise overfitting. We employed Adversarial Point Perturbation as a data augmentation technique. The Victim Model was initially chosen as PointNet. However, Experiments 4.2 and 4.3 indicated that PointNet fails to categorise data that is highly similar. As a result, we employed PointNet with Siamese as the Victim Model. We were able to generate massive volumes of data. After utilising ICP to test the technique, it is clear that the perturbed data is highly comparable to their parent' data and may be employed in a variety of applications where 3D data is scarce.

In the future, this perturbation technique will need to be evaluated with partial 3D data to see if the perturbation technique can be implemented. Due to time constraints, the technique must be evaluated on different data sets to ensure its viability.

Bibliography

[1] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, p. 60, Jul 2019.

[2] B. Furht, ed., *Mesh, 3D*, pp. 406–407. Boston, MA: Springer US, 2006.

[3] C. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 77–85, 2017.

[4] W. Hayale, P. Negi, and M. Mahoor, "Facial expression recognition using deep siamese neural networks with a supervised loss function," in *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)*, (Los Alamitos, CA, USA), pp. 1–7, IEEE Computer Society, may 2019.

[5] H. Wu, Z. Xu, J. Zhang, W. Yan, and X. Ma, "Face recognition based on convolution siamese networks," *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pp. 1–5, 2017.

[6] C. Xiang, C. Qi, and B. Li, "Generating 3d adversarial point clouds," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 9136–9144, 2019.

[7] A. M. Srivastava, A. Jain, P. Rotte, S. Prakash, and U. Jayaraman, "A technique to match highly similar 3d objects with an application to biomedical security," *Multim. Tools Appl.*, vol. 81, pp. 13159–13178, 2022.

[8] J. Procházková and D. Martišek, "Notes on iterative closest point algorithm," in *Proc. in 17th Conference on Applied Mathematics*, pp. 876–884, 2018.

[9] D. Chetverikov, D. Stepanov, and P. Krsek, "Robust euclidean alignment of 3d point sets: the trimmed iterative closest point algorithm," *Image and Vision Computing*, vol. 23, pp. 299–309, Mar 2005.

[10] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 922–928, 2015.

[11] C. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view cnns for object classification on 3d data," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5648–5656, 2016.

[12] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1912–1920, 2015.

[13] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 945–953, 2015.

[14] M. Savva, F. Yu, H. Su, M. Aono, B. Chen, D. Cohen-Or, W. Deng, H. Su, S. Bai, X. Bai, N. Fish, J. Han, E. Kalogerakis, E. Learned-Miller, Y. Li, M. Liao, S. Maji, A. Tatsuma, Y. Wang, and Z. Zhou, "Shrec'16 track large-scale 3d shape retrieval from shapenet core55," 01 2016.

[15] M. Aubry, U. Schlickewei, and D. Cremers, "The wave kernel signature: A quantum mechanical approach to shape analysis," *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pp. 1626–1633, 2011.

[16] J. Sun, M. Ovsjanikov, and L. Guibas, "A concise and provably informative multi-scale signature based on heat diffusion," *Computer Graphics Forum*, vol. 28, no. 5, pp. 1383–1392, 2009.

[17] M. M. Bronstein and I. Kokkinos, "Scale-invariant heat kernel signatures for non-rigid shape recognition," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1704–1711, 2010.

[18] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," *2009 IEEE International Conference on Robotics and Automation*, pp. 3212–3217, 2009.

[19] R. B. Rusu, N. Blodow, Z.-C. Márton, and M. Beetz, "Aligning point cloud views using persistent feature histograms," *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3384–3391, 2008.

[20] C. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *NIPS*, 2017.

[21] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Transactions on Graphics (TOG)*, vol. 38, pp. 1 – 12, 2019.

[22] J. C. Joshi, A. Gupta, K. Tiwari, and K. K. Gupta, "Scanned to digital face images matching with siamese network," in *2018 3rd International Conference and Workshops on Recent Advances and Innovations in Engineering (ICRAIE)*, pp. 1–6, 2018.

[23] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *CoRR*, vol. abs/1312.6199, 2014.

[24] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv 1412.6572*, 12 2014.

[25] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial

settings," in *2016 IEEE European Symposium on Security and Privacy (EuroS P)*, pp. 372–387, 2016.

[26] N. Carlini and D. A. Wagner, "Defensive distillation is not robust to adversarial examples," *ArXiv*, vol. abs/1607.04311, 2016.

[27] D. Meng and H. Chen, "Magnet: A two-pronged defense against adversarial examples," *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017.

[28] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, 2017.

[29] F. Xiong, B. Zhang, Y. Xiao, Z. Cao, T. Yu, J. T. Zhou, and J. Yuan, "A2j: Anchor-to-joint regression network for 3d articulated pose estimation from a single depth image," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 793–802, 2019.

[30] L. Yang, S. Li, D. Lee, and A. Yao, "Aligning latent spaces for 3d hand pose estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2335–2343, 2019.

[31] M. Oberweger and V. Lepetit, "Deepprior++: Improving fast and accurate 3d hand pose estimation," *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pp. 585–594, 2017.

[32] L. Ge, Y. Cai, J. Weng, and J. Yuan, "Hand pointnet: 3d hand pose estimation using point sets," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8417–8426, 2018.

[33] S. Hinterstoisser, V. Lepetit, P. Wohlhart, and K. Konolige, "On pre-trained image features and synthetic images for deep learning," 10 2017.

[34] A. M. Srivastava, P. A. Rotte, A. Jain, and S. Prakash, "Handling data scarcity through data augmentation in training of deep neural networks for 3d data processing," *International Journal on*

Semantic Web and Information Systems (IJSWIS), vol. 18, no. 1, pp. 1–16, 2022.