UKRAINIAN CATHOLIC UNIVERSITY

MASTER THESIS

Point cloud human pose estimation using capsule networks

Author:
Oleksandr ONBYSH

Supervisor:
Andrii BABII

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2021

Declaration of Authorship

I, Oleksandr ONBYSH, declare that this thesis titled, "Point cloud human pose estimation using capsule networks" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

UKRAINIAN CATHOLIC UNIVERSITY

Abstract

Faculty of Applied Sciences

Master of Science

Point cloud human pose estimation using capsule networks

by Oleksandr ONBYSH

Human pose estimation based on points cloud is an emerging field that develops with 3D scanning devices' popularity. Build-in LiDAR technology in mobile phones and a growing VR market creates a demand for lightweight and accurate models for 3D point cloud. Widely advanced deep learning tools are mainly used for structured data, and they face new challenges in unstructured 3D space. Recent research on capsule networks proves that this type of model outperforms classical CNN architectures in tasks that require viewpoint invariance from the model. Thus capsule networks challenge multiple issues of classic CNNs like preserving the orientation and spatial relationship of extracted features, which could significantly improve the 3D points cloud classification task's performance.

The project's objective is to experimentally assess the applicability of capsule neural network architecture to the task of point cloud human pose estimation and measure performance on non-synthetic data. Additionally, measure noise sustainability of capsule networks for 3D data compared to regular models. Compare models' performance with restricted amount of training data.

Contents

List of Figures

List of Tables

List of Abbreviations

CapsNet	Capsule network
AP	Average precision
mAP	Mean average precision
ITOP	Invariant Top View dataset
SOTA	State Of The Art

List of Symbols

P	set of points in D space (point cloud)
J	set of key joints positions
N	resolution of point cloud (number of points in one cloud)
d_{CD}	Chamfer distance

Chapter 1

Introduction

1.1 Problem

Human pose estimation is a task based on a human's image or 3D points, and the model should locate the main joints in the human's body (head, neck, left and right arms, spin, etc.) as shown in the Figure 1.1. Each joint is represented as a point in 2D or 3D space based on the task's objective.

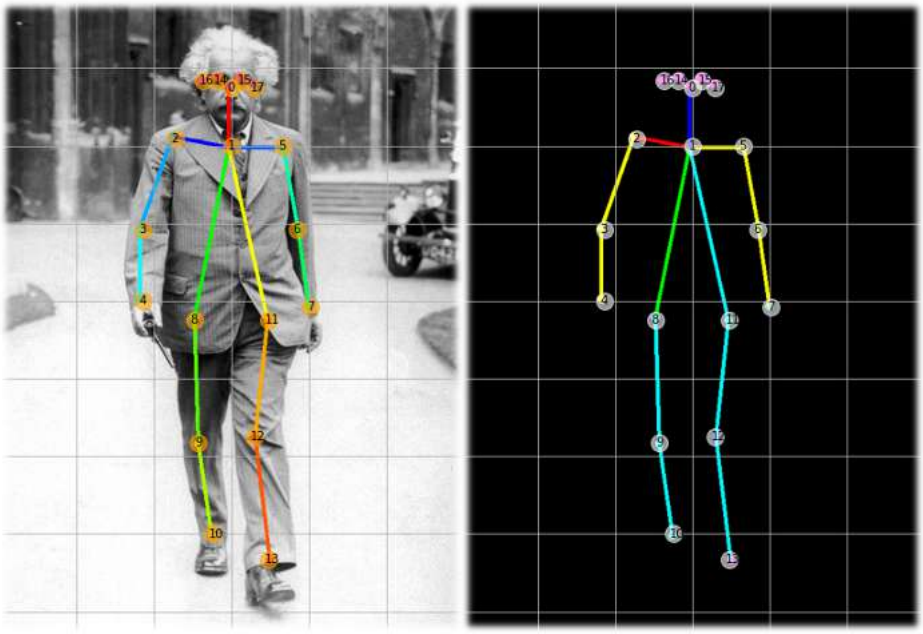

FIGURE 1.1: Example of human pose estimation key points Rovai, 2020

A 3D point cloud is simply a set of points with three positional coordinates and represents points in 3D space. The points represent the shape of the object in 3D space. 3D point clouds are usually gathered by 3D scanners or dual-lens cameras.The output of scanners is point cloud where each point corresponds to some point on scanning surface with predefined precision. With the rapid growth of LIDAR and VR fields, the importance of accurate and fast 3D point cloud human pose estimation algorithms is clear.

1.2 Challenges

The obvious challenge of human pose estimation is the potential space of different human postures. The small change in the body part position changing the target pose. The task gets more complicated with different obstacles like clothes. Using recent ML algorithms such as deep learning on 3D point cloud results in many challenges. Some of the common issues are:

- The high dimensionality of the input space. Compared to pose estimation based on images, 3D point cloud has higher-dimensional space.
- Noisy inputs from 3D point cloud scanners. The sparsity and accuracy of the point cloud greatly influence the model's performance. The accuracy and granularity of points are significantly dependent on the scanning device. Compact LIDAR scanning devices the most popular and less accurate.
- Geometric-viewpoint relation. The human body has a strict geometric relation between body parts, which is invariant to the viewpoint. Most of the deep learning algorithms are not viewpoint agnostic resulting in additional challenges for human pose estimation.
- Lack of data. The amount of data for 3D point cloud human pose estimation is significantly less than regular image datasets. This fact is due to the high complexity of collecting 3D point cloud data, e.g., need special multicamera or LIDAR equipment, need diverse human positions, need different human constitutions.

1.3 Motivation

Human pose estimation is an important task that is used in different fields. A lot of research work was done to solve the task of human pose estimation using point clouds. These works made significant progress in the regression task and showed good performance on public datasets. However, most methods use the end-to-end approach of learning, paying less attention to the inner structure of point cloud objects. The mutual relationship between an object's parts (in our case human body) plays a crucial role during the regression. The recent works (Cheraghian and Petersson, 2018; Wu et al., 2020) which use capsule-based methods for point clouds, show that these types of models could consider object's part relationships. Moreover, auto-encoder capsule-based networks could learn objects' properties in unsupervised manner (Wu et al., 2020). These properties of capsule networks look promising for tasks where viewpoint invariance is a substantial challenge. Moreover, some experiments (Wang et al., 2020; Gritsevskiy and Korablyov, 2018) stated that capsule networks need less data for convergence compared to non-capsule models. Besides, it is more noise agnostic than regular convolutional networks due to latent space inside the capsule network.

1.4 Research Gap

- There is no capsule-based model for 3D human pose estimation task.
- There is no comparison of the influence of noisy data on capsule-based and non-capsule-based models for 3D space.

- There is no comparison of how well capsule-based works with limited dataset compared to regular models.

1.5 Objective

The work's objective is to propose a model based on a capsule network for a 3D point cloud human pose estimation. Evaluate the model on public benchmark dataset for human pose estimation. Compare results with state of the art approaches for the task as mentioned earlier. Evaluate the influence of the noise in training dataset on capsule-based and non-capsule-based networks. Measure the performance of the models with different sizes of training dataset.

1.6 Paper structure

Section 2 covers the related work of human pose estimation based on both 2D images and 3D point clouds. This section described conventionally, and state of the art approaches for solving the issue. Reviews capsule networks for different 3D point cloud tasks like point classification, segmentation, and position estimation.

The rest of the paper is organized in the following manner:

- Section 3 presents the project's hypothesis and problems;
- section 4 describes the dataset which is used for training and evaluation along with evaluation metrics;
- section 5 describes the approach for solving the project's objectives;
- section 6 describes experiments on project's objections described in 5;
- section 7 sums up the paper's ideas and present brief conclusions and possible future work in this field.

Chapter 2

Related work

In this chapter, we overview common approaches to solve problems with point clouds. We review common techniques, models, and algorithms for point clouds.

In Section 2.1 and Section 2.2 we review common approaches on how to use point clouds in deep learning, along with preprocessing methods for 3D point data.

In Section 2.3 we review models and techniques for the task of human pose estimation, both for 2D and 3D spaces.

In Section 2.4 we review the initial work on the capsule network which were released for the task of image classification. Also, we review few works which adapt capsule networks for 3D point cloud data.

2.1 Deep learning approaches for point cloud

The different number of points and high dimensionality of the point cloud input makes it challenging to use regular 2D convolutions. The typical approach for such an issue is the conversation of the point cloud to a different format. Such approaches are projection-based methods, volumetric-based methods, and other geometric-based methods.

2.1.1 Projection-based methods

Projection-based methods take point cloud and project it into a different panel view. After the projection, each view provides a set of combined features for target classification, regression, or segmentation. The critical challenge for the projection-based algorithm is the multi-view feature aggregation into one global feature space.

MVCNN (Su et al., 2015) is the first CNN-based architecture which recognize rendered views of different shapes independently from each other. The model's performance shows that even from one view the 3D shape could be recognized with competitive accuracy. Adding more views increase the overall accuracy of the model. The processing pipeline if the MVCNN is shown in Figure 2.1.

MHBN (Yu, Meng, and Yuan, 2018) (Multi-view Harmonized Bilinear Network) is the continuation of MVCNN. The approach proposes to integrates local convolutional features by harmonized bilinear pooling to produce a compact global descriptor. To persist the information from different views, the View-GCN (Wei, Yu, and Sun, 2020) proposes constructing view-graph with multiple views as graph nodes, then designing a graph convolutional neural network over view-graph to learn discriminative shape descriptor hierarchically. All projection-based methods struggle from high memory consumption and high computational complexity since, for one feature extraction, the model should be run for the number of different views.

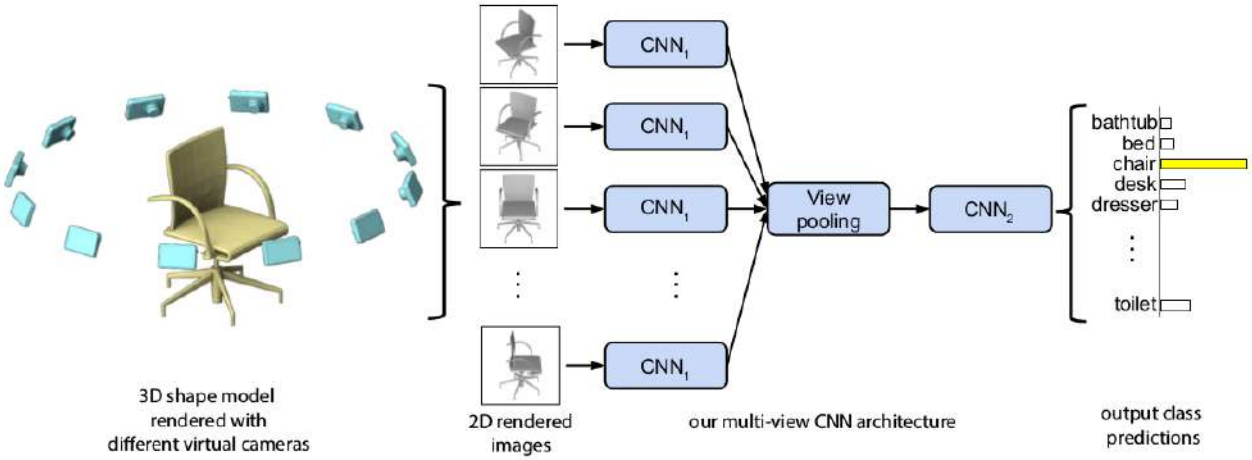

FIGURE 2.1: MVCNN processing pipeline (Su et al., 2015)

2.1.2 Volumetric-based methods

Volumetric-based methods map the point cloud into a 3D grid. Then conventional 3D convolutions are using for feature extraction. VoxNet (Maturana and Scherer, 2015) is the first method that exploits the volumetric representation of the point cloud. In this work, each cloud point is mapped to a discrete voxel point. The size of the target grid is 32 x 32 x 32 voxels. After the mapping, three convolutional layers are using to produce the target feature representations. Processing steps of VoxNet is shown in Figure 2.2

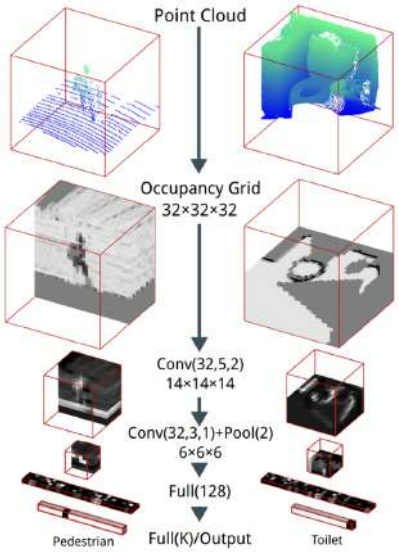

FIGURE 2.2: VoxNet processing pipeline (Maturana and Scherer, 2015)

The more advanced volumetric-based models use octrees data structure. OctNet (Riegler, Ulusoy, and Geiger, 2017) propose to represent the point cloud as several octrees along a regular grid (Figure 2.3), each octree is encoded as a bit string, and features are generated through naive arithmetic. This approach reduces the memory consumption of the model during the training and inference stages. The next iteration of octrees representation of point cloud is proposed in O-CNN (Wang et al.,

2017). The model uses 3D convolutions to extract features from octrees. This model also uses octree representation. The model takes the average normal vectors if 3D model from leafs of octants and runs 3D CNN on this to perform classification.

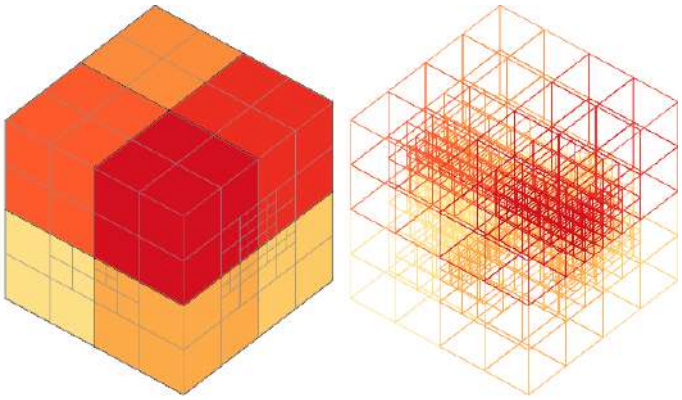

FIGURE 2.3: An example of octree grid (Riegler, Ulusoy, and Geiger, 2017)

2.2 Point-based Methods

Compared with projection-based methods and volumetric-based methods that aggregate points from a spatial neighborhood, point-based methods attempt to learn features from individual points. Most of the recent work focuses on this direction.

The first work which uses a point-based approach is PointNet (Qi et al., 2017a). PointNet learns pointwise features independently with several MLP layers and extracts global features with a max-pooling layer. The input (an $n \times 3$ 2D tensor) is first multiplied by an affine transformation matrix predicted by a mini-network (T-Net) to hold invariance under geometric transformations. The point set is then passed through a group of MLPs followed by another joint alignment network, and a max-pooling layer to obtain the final global feature. The model's architecture is depict in Figure 2.4

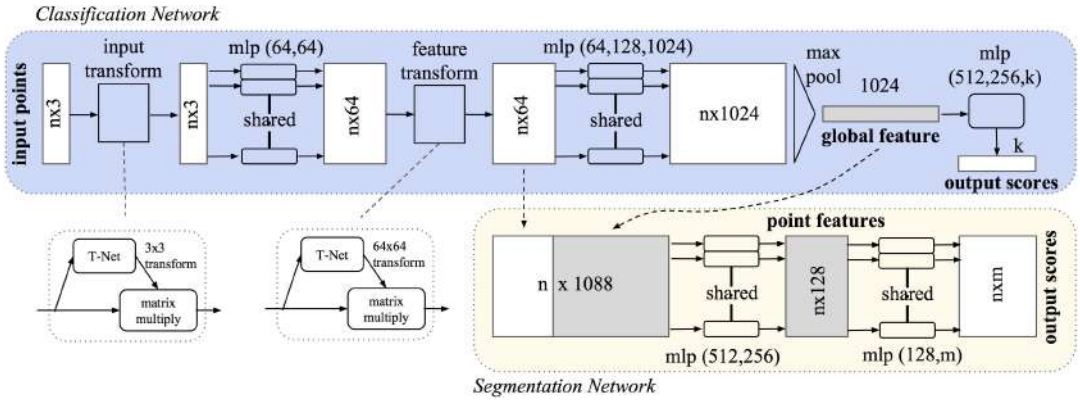

FIGURE 2.4: PointNet architecture (Qi et al., 2017a)

The second iteration of PointNet is PointNet++ (Qi et al., 2017b). PointNet++ introduces a hierarchical neural network that applies PointNet recursively on a nested partitioning of the input point set. Using metric space distance, the model could learn local features with increasing contextual scale. The state-of-the-art model for

point-based classification is Point Attention Transformers (Yang et al., 2019). The research for the first time proposes the mechanism of sampling which is end-to-end and task agnostic. The sampling is named "Gumbel Subset Sampling" - GSS. This sampling is used to select the most representative subset of the point cloud from the initial data. Using Gumble-Softmax, the model could provide a continuous point subset in the training phase, and a strict discrete subset in the test phase. Using such an approach the model is able to learn a more robust representation of the input data with less computational complexity.

2.3 Human pose estimation

The latest research approaches in the field of human pose estimation are based on deep learning. There are two main approaches to the task:

- pose estimation based on 2D images (mostly RGB);
- pose estimation based on the 3D point cloud.

The latter approach is more recent and promising. The 3D perspective gives more information for the models about body position in the space. Also, 3D point clouds mitigate the issue with occluded parts of the body. 2D image is a 2D projection of 3D space, and this transformation leads to the loss of information.

2.3.1 Image-based methods

The approaches for 2D image human pose estimation are divided into two types:

- top-down approach
- bottom-up approach

In the top-down approach, the first step is person detection and then pose regression. In the bottom-up approach, all body parts are detected first and then grouped according to the body's position.

OpenPose (Cao et al., 2019) is the most popular example of bottom-up approaches for multi-person pose estimation. The network first extracts features from the image using the VGG feature extractor. Then features are passed to two separate branches, the first branch predicts body parts key points, the second branch predicts the associativity between body parts. The result of human pose estimation from OpenPose is shown in Figure 2.5.

RMPE (AlphaPose) (Fang et al., 2018) is a top-down model. This approach proposes to use a four-stage pipeline. The first step is Symmetric Spatial Transformer Network (SSTN). In this step, the model extracts the human region using a bounding box. The second step is a Single Person Pose Estimator (SPPE). This step model uses extracted human regions to estimate key joints of the human body. The third step is spatial De-Transformer Network (SDTN). This step remap estimated human joint coordinates back to the original coordinate system. The last step is parametric pose Non-Maximum Suppression (NMS). This step handles the issue with redundant pose deductions.

FIGURE 2.5: An example OpenPose network result (Cao et al., 2019)

2.3.2 point-cloud-based methods

Point cloud-based estimation is a relatively young field due to the recent growth of popularity of point cloud scanning devices.

The first model for human pose estimation based on point cloud was presented by Diaz Barros, Garcia, and Sidibé, 2015. The paper presents an approach where based on a predefined human body skeleton the input point cloud is clustered using PCA and Expectation maximization algorithms.

The recent work in this field is presented by Zhou, Dong, and Saddik, 2020. The work takes point clouds as input data and model the surface of the object, in this research - human body, using deep human pose network. The pros of this approach is that it's an end-to-end model. It takes 2D depth image, transforms it to 3D point cloud, and then estimate key human joint coordinates. The model's architecture is shown in Figure 2.6

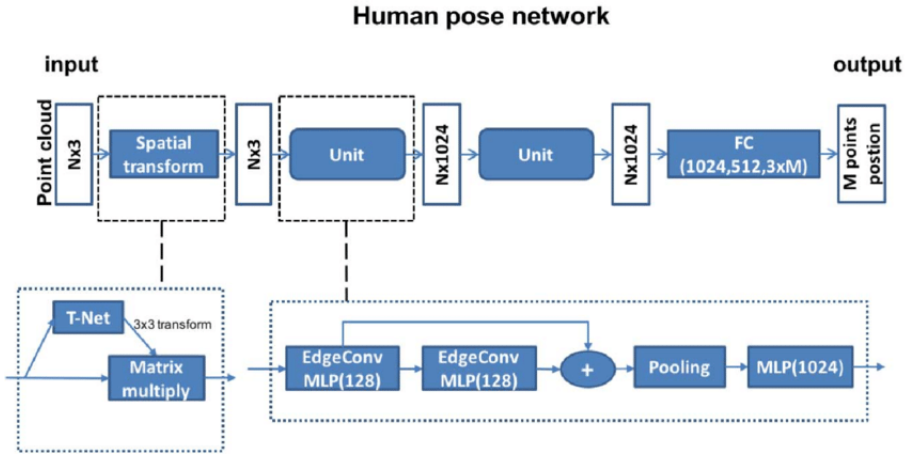

FIGURE 2.6: Architecture of DNN presented by (Zhou, Dong, and Saddik, 2020)

More point cloud human pose estimation methods will be covered in the next subsection. The next subsection covers models which are based on capsule architecture.

2.4 Capsule network

The concept of the capsule was first proposed by Hinton (Sabour, Frosst, and Hinton, 2017) and has been widely used in 2D and 3D deep learning (Kakillioglu et al., 2020; Qin et al., 2020; Duarte, Rawat, and Shah, 2018; LaLonde and Bagci, 2018).

Capsules are represented as a set of vectors. The length of the capsule's vector represents the probability of the object's presence. The direction of the vector describes the object's property e.g. position, viewpoint, size, shape, etc. For capsules' training Hinton proposes a new algorithm (Sabour, Frosst, and Hinton, 2017) called dynamic routing. The forward pass with dynamic routing propagates the input data from lower-level capsules to higher-level ones. Lower-level capsules pass learned and predicted data to the higher-level capsules. If multiple lower-level capsules agree (activated) then higher-level capsules activate accordingly. With each iteration of dynamic routing, each capsule gets more accurate.

2.4.1 Capsule networks for point cloud classification

The first work where capsule networks were applied to the problem of point cloud classification is 3DCapsNet (Cheraghian and Petersson, 2018). In this work, a new capsule-based layer is proposed - ComposeCaps. ComposeCaps learns spatially relevant feature mapping that can be exploited for 3D point cloud classification. The architecture of the network is shown in Figure 2.7

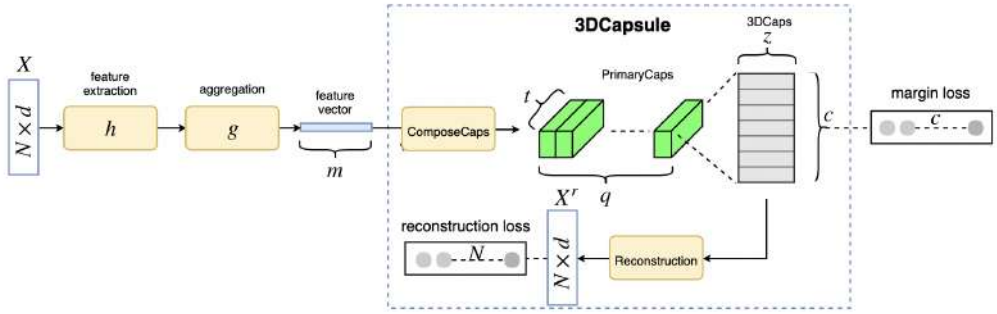

FIGURE 2.7: The architecture of 3DCapsNet (Cheraghian and Petersson, 2018)

The second iteration of capsule applicability to 3D classification is the 3D point capsule network (Zhao et al., 2019). 3D point capsule network is an auto-encoder designed based on capsule networks. In this work researchers propose new architecture with a capsule network encoder that encodes input point cloud to capsules' latent space, and a decoder that decodes latent capsules. The proposed architecture works for several common point cloud-related tasks, such as object classification, object reconstruction, and part segmentation.

2.4.2 Capsule networks for point cloud regression

The only work which is currently presented on the topic of point cloud regression is Capsule-HandsNet (Wu et al., 2020). This project is inspired by this research. Capsule-HandsNet uses model based on capsule auto-encoder proposed in Zhao et al., 2019. The model is an end-to-end, it takes hand point cloud and predicts key joints. The latent space of the model provides an ability to "memorize" the internal

structures of the object such as symmetry, junction, relative location of object's parts, etc. The restored point cloud is combined from local patches predicted by capsules inside of the decoder (Figure 2.8).

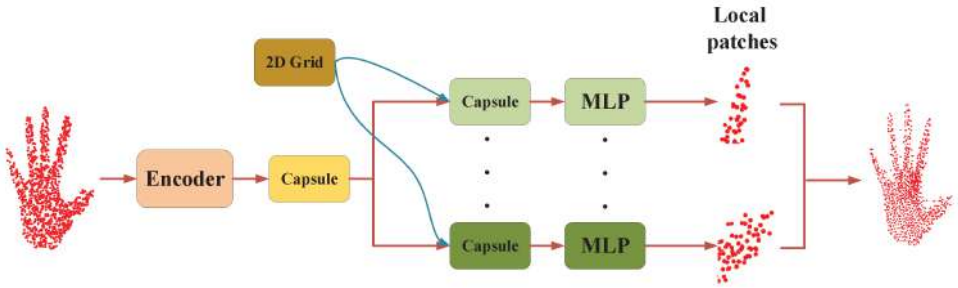

FIGURE 2.8: Reconstructed point cloud from capsule local patches (Wu et al., 2020)

Chapter 3

Research Hypothesis and Problem

3.1 Hypotheses

Model Comparison hypothesis: This project's main objective is to create a model for human pose estimation based on point cloud using a capsule-based neural network, which shows competitive performance on well-known benchmarks.

Noise resistance hypothesis: The impact of noise in the training dataset on the capsule-based model should be less compared to non-capsule-based models. A hypothesis is made based on 2D image recognition using capsule networks Sabour, Frosst, and Hinton, 2017.

Dataset size hypothesis: The dataset's size for compatible results should be smaller for capsule-based networks compared to non-capsule ones. The assumption is made based on experiments presented is Sabour, Frosst, and Hinton, 2017; Wang et al., 2020; Gritsevskiy and Korablyov, 2018 based on 2D image classification.

3.2 Problems

Noise problem. To achieve the project goal mentioned above, we need to generate realistic noise for point cloud data. We need to compare how noisy data influence capsule-based and not capsule-based models.

Models' retrain problem. To achieve the project's third goal, we need to retrain the reference SOTA [1] model with truncated training data. After this compare retrained model with the reference capsule-based model.

[1] state of the art

Chapter 4

Dataset and evaluation metrics

In the Section 4.1 we describe the dataset which is used for training and evaluation purpose for our task of the human pose estimation. In Section 4.2 we define the evaluation metric for the aforementioned task.

4.1 Dataset

The most common datasets for the task of human pose estimation from point clouds are ITOP (Haque et al., 2016) and EVAL (Liu et al., 2020). In this work we concentrate on ITOP dataset since it's more widely used (Shotton et al., 2011; Ho Yub Jung et al., 2015; Carreira et al., 2016; Chen et al., 2020; Moon, Chang, and Lee, 2018) thus has more related works for results' comparison.

4.1.1 ITOP

ITOP dataset was first released in the work of Haque et al., 2016. The dataset comprises depth images of people in different poses. An example of the depth map from the dataset is shown in Figure 4.1.

The snapshots of poses are taken from two different viewpoints: the side view, and the top view. In side view, the camera faces a person directly in front (the whole body is seen). In the top view, the camera is placed above the person's head and shots the view from the top (torso usually occluded by shoulders and arms).

The dataset is collected inside of the room thus some furniture items appear in the dataset. Due to excess obstacles in the field of view of the camera, we apply preprocessing pipelines to extract only human point clouds. The pipeline is described in Section 5.1. Figures 4.2 and 4.3 show examples of side and top views respectively.

The 3-dimensional (x, y, z) coordinates are relative to sensor's position. The distance is measured in meters. Thus, if x coordinate hasa value of 2, it means that the point is 2 meters right relatively to the sensor's $(0, 0, 0)$ reference point.

Ground truths consist of 15 joint key point coordinates - head, neck, shoulders, elbows, hands, torso, hips, knees, and feet. An example of named joints is shown in Figure 4.4. Table 4.1 covers general statistics on dataset size.

4.2 Evaluation metric

Since we are working on the regression task, we use mean Average Precision (mAP) as our main metric of the model's performance. The ground truths are coordinates of key human joints (denoted as J). The output of the regression model is also joints' coordinates of the same size as J (denoted as $\hat{J}$).

TABLE 4.1: General statistic for ITOP dataset

View	Split	Frames	People
side	train	39,795	16
side	test	10,501	4
top	train	39,795	16
top	test	10,501	4

FIGURE 4.1: Four examples of depth images from ITOP dataset. The darker the color the closer it's located to the camera

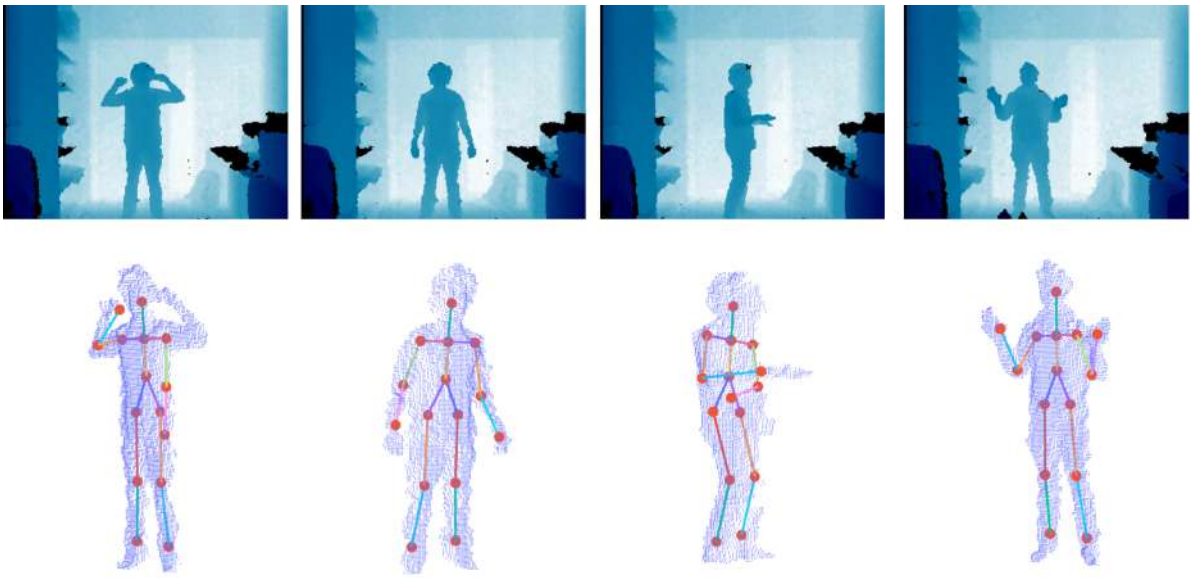

FIGURE 4.2: Four examples of front view subset of ITOP. Upper images are depth representation. Bottom images show human point clouds (blue) with key joints (red)

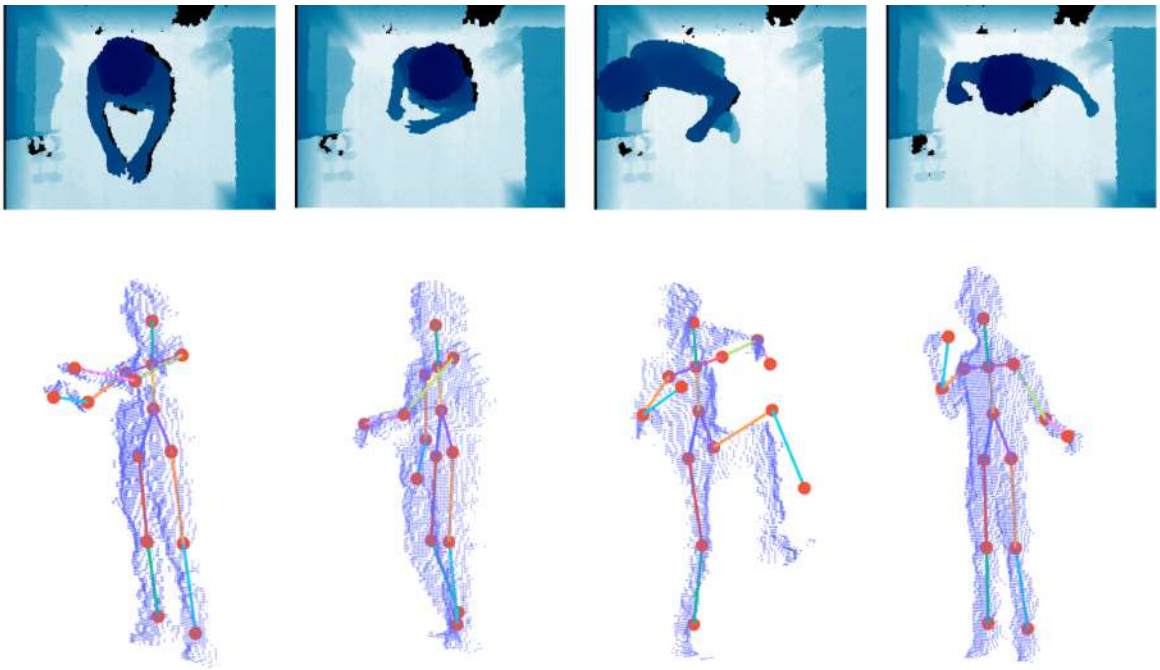

FIGURE 4.3: Four examples of top view subset of ITOP. Upper images are depth representation. Bottom images show human point clouds (blue) with key joints (red)

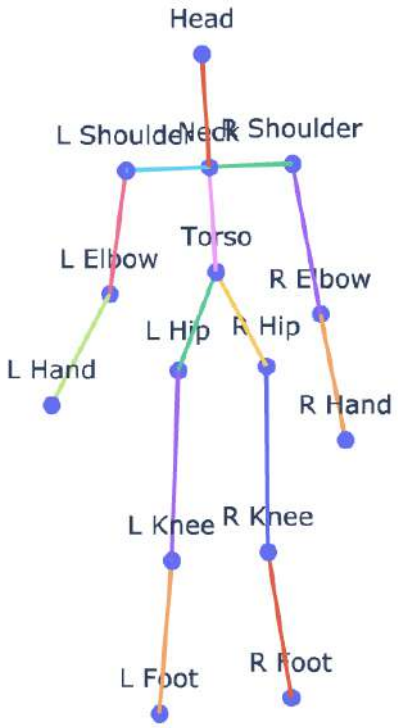

FIGURE 4.4: Example of key joints with names from ITOP dataset

To be able to compare our results with the work of others (Haque et al., 2016; Moon, Chang, and Lee, 2018; Guo et al., 2017) we use the same approach of calculation of AP using 10 cm distance threshold. By this rule (Formula 4.1) the predicted joint is considered correctly detected if the L_2 distance to ground truth is equal or less that 10 cm. The resulting mAP is calculated by averaging AP by the number of samples (Formula 4.2).

For further convenience, we will also apply the plot with mAP values for distances from 0 cm to 100 cm (Example in Figure 4.5).

$$AP(J, \hat{J}) = \begin{cases} 1, & \|J - \hat{J}\| < 10 \text{ cm} \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

$$mAP = \frac{\sum_{i=0}^{M} AP\left(J_i, \hat{J}_i\right)}{M} \tag{4.2}$$

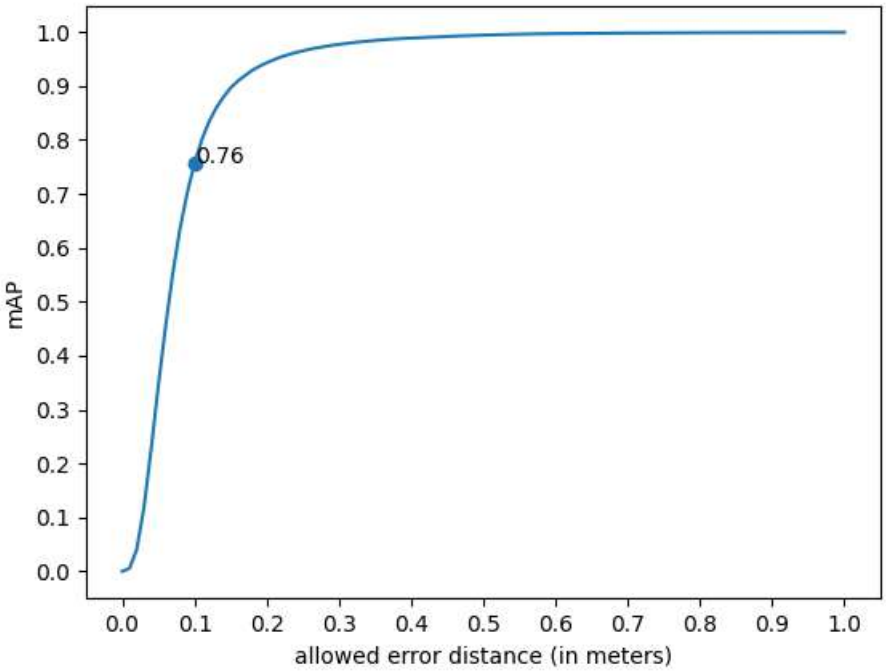

FIGURE 4.5: Example of the mAP plot. On the X axis - allowed distance between predicted and ground truth coordinates to be considered as a right detection. On the Y axis - mean Average Precision for given distance. The values for **distance** $= 10\ cm$ is highlighted on the plot.

Chapter 5

Methodology

In this section, we introduce the methodology for our proposed approaches for the task of human pose estimation using the point cloud. As we reviewed in Chapter 2, the key issue of any model which uses point clouds as input data, it's a huge spatial variation of the data. Point clouds much diverse compared to 2D images in terms of location and rotation. Capsule networks show promising results on 2D data (Sabour, Frosst, and Hinton, 2017), and such models handle better spatial invariance compared to regular NN models.

The chapter consists of two main sections. In Section 5.1 we cover the preprocessing for the input data and ground truths and methods of adding artificial noise to the input data.

In Section 5.2 we cover the network's architecture and algorithms for performing network training.

5.1 Data preparation

In this section, we cover the main preprocessing steps for datasets. Preprocessing consists of:

1. human extraction;
   (a) threshold filtering;
   (b) clusterization & human cluster selection;
2. point cloud normalization;
3. adding noise to data (optional).

In the first step, we remove the most obvious points which don't represent human posture. In the second step, we perform clustering and segmentation to extract "human" clusters. The third step is optional and is used in Section 6.4 where we measure the performance of different models with various levels of noise.

All steps of the preprocessing pipelines are shown in Figure 5.1: threshold filtering, clusterization, and segmentation.

5.1.1 Human extraction

To start working with point cloud for human pose estimation we need to extract human points out of the overall point cloud. As we can see from Figure 5.2, the raw point cloud contains not only a human point cloud but also other objects which are located in the room, such as walls, cupboards, and other objects. These obstacles will not only harm the model's performance but will also greatly slow the model's training and inference due to the excess number of points.

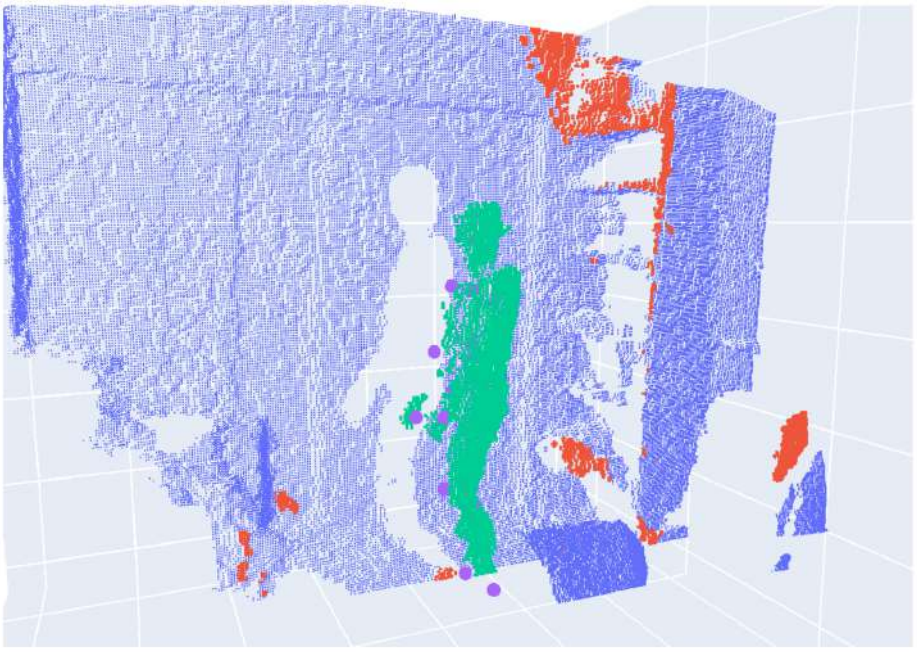

FIGURE 5.1: Example of preprocessing stages. **Blue** - points filtered by threshold. **Red** - points removed by human segmentation. **Green** - extracted human point cloud. **Purple** - key human joints

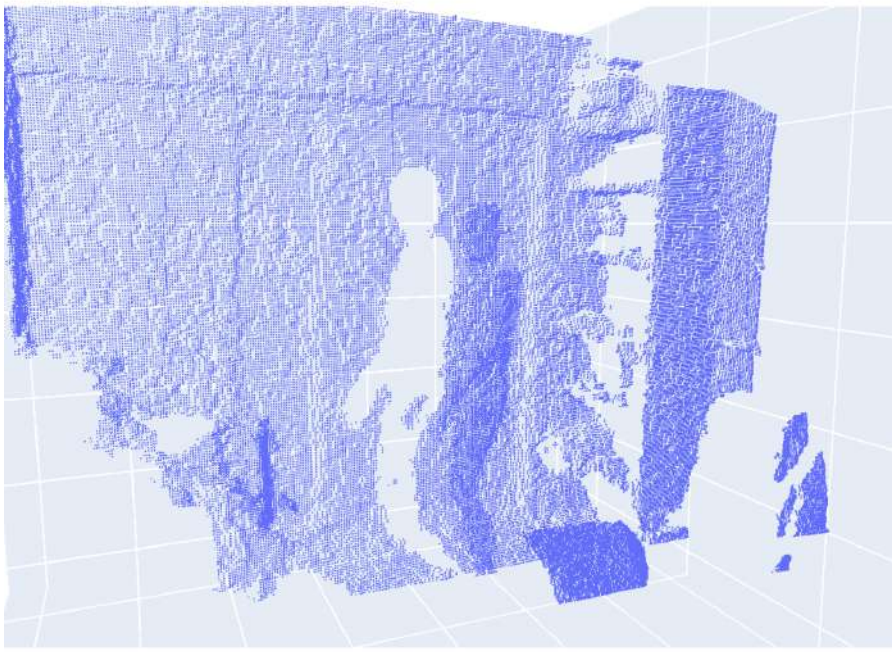

FIGURE 5.2: Example of raw data from ITOP dataset (front view)

Threshold filtering

Threshold filtering is the first step in preprocessing pipeline. The aim of this step is to filter out the most obvious points which don't belong to the human posture. After manual checks we came up to such parameters side view dataset:

$$\begin{aligned} x_{min} &= -1 & x_{max} &= 1 \\ y_{min} &= -1.4 & y_{max} &= 2 \\ z_{min} &= -1.5 & z_{max} &= 3.5 \end{aligned} \tag{5.1}$$

and for top view dataset:

$$\begin{aligned} x_{min} &= -0.91 & x_{max} &= 0.85 \\ y_{min} &= -0.63 & y_{max} &= 0.57 \\ z_{min} &= 0 & z_{max} &= 2.6 \end{aligned} \tag{5.2}$$

Formula 5.1 and 5.2 set min-max distance values from camera view. The camera sensor is considered as center - $(0, 0, 0)$. In this way, x coordinate represents left-right direction from the camera center, y - top-bottom, and z - the depth.
An example of a threshold filtered point cloud could be seen in Figure 5.3.

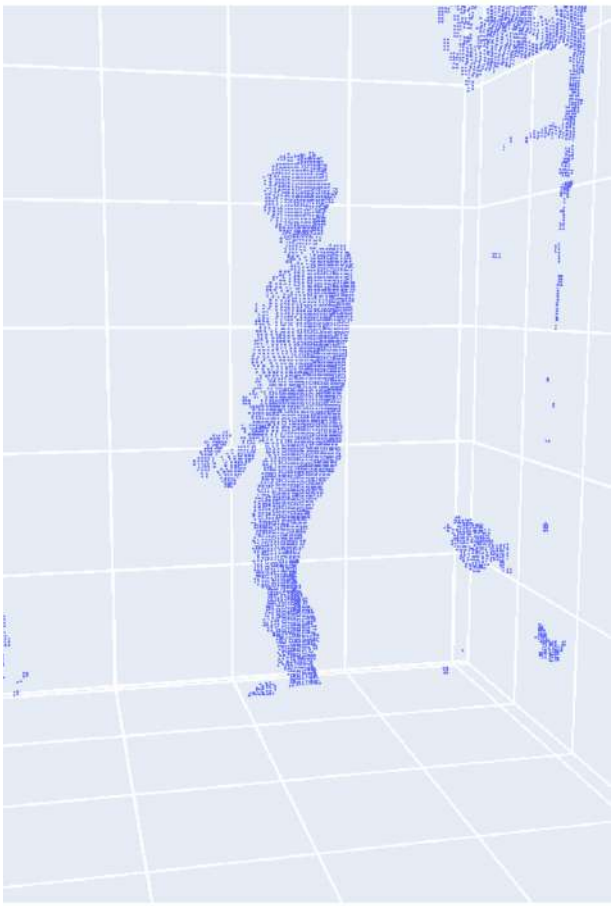

FIGURE 5.3: Example of point cloud after applying of threshold filter

Since thresholds were manually selected it's crucial to validate that all human points clouds fit these boundaries. In our case, we run checks on resulting point clouds after the segmentation step which will be described later. By this check, we confirmed that at the end of our extraction pipeline each example in the dataset contains exactly one big cluster (number of points $>$ 2000). This check doesn't guarantee that our method works well on each example, but it's enough to say with high probability that it works on the vast majority of cases. However, it's worth stressing that this step in our pipeline could be improved in future work.

Clusterization

Clusterization & extraction are the next steps in preprocessing pipeline. In this step, we cluster the point cloud from the threshold filtering step (Subsection 5.1.1) and extract the human cluster. For clusterization step we propose the nearest-neighbour search algorithm (*Nearest neighbor search* 2021) with kd-tree data-structure, using FLANN [1].

The aim of the clusterization is to separate human clusters and small noisy clusters from the point cloud. As we can see from the Algorithm 1, for each point in the initial point cloud P we try to put it to some cluster based on the distance (radius - r) to that cluster. If the distance is less than the radius to each cluster then this point creates a new cluster. At the end of clusterization, we get a set of clusters C.
Since the human cluster always the biggest, at the end of clusterization we take the argmax and get human cluster C_{human}.

In the previous paragraph, we made an assumption that human is the biggest cluster out of all clusters. Although the assumption holds for the ITOP dataset it doesn't guarantee the same performance for other datasets. Thus, this step in preprocessing pipeline could be improved in future work.

In Figure 5.4 we can see the point cloud before clusterization, after, and the resulting human cluster (the biggest cluster).

[1] Fast Library for Approximate Nearest Neighbor

FIGURE 5.4: Example of point cloud after applying human extraction

Algorithm 1: Point cloud clusterization and human extraction (*PCL - Euclidean Cluster Extraction*)

Result: human point cloud cluster - C_{human}

create a Kd-tree representation for the input point cloud dataset P ;
set up an empty list of clusters C, and a queue of the points that need to be checked Q ;
for *every point* $p_i \in P$ **do**
    add p_i to the current queue Q ;
    for *every point* $p_i \in Q$ **do**
        search for the set P_i^k of point neighbors of p_i in a sphere with radius $r < d_{th}$;
        for every neighbor $p_i^k \in P_i^k$, check if the point has already been processed, and if not add it to Q;
    end
    when the list of all points in Q has been processed, add Q to the list of clusters C, and reset Q to an empty list ;
end
the algorithm terminates when all points $\boldsymbol{p}_i \in P$ have been processed and are now part of the list of point clusters C ;
$C_{human} = \arg\max_{x \in C}$

5.1.2 Point cloud normalization

To allow the network to learn more quickly the optimal parameters for each point cloud we normalize the input data to a standard scale.

We use the classic normalization technique where we scale the data to have zero mean and unit variance. We calculate the mean and max values for each axis X, Y, and Z. Then we scale point cloud and ground truth key joints coordinates according to these values. (see Formula 5.3 and Formula 5.4).

$$P_{normalized} = \frac{P - \mathbf{mean}(P)}{\mathbf{max}(P) - \mathbf{min}(P)} \tag{5.3}$$

$$J_{normalized} = \frac{J - \mathbf{mean}(P)}{\mathbf{max}(P) - \mathbf{min}(P)} \tag{5.4}$$

To calculate reconstruction and regression loss for the network we need to denormalize reconstructed point cloud, and regressed key joints coordinates. For denormalization, we use the same mean, max, and min values by which we normalized the data in the first place (see Formula 5.5 and Formula 5.6).

$$\hat{P}_{\mathbf{denormalized}} = \hat{P} \cdot (\mathbf{max}(P) - \mathbf{min}(P)) + \mathbf{mean}(P) \tag{5.5}$$

$$\hat{J}_{\mathbf{denormalized}} = \hat{J} \cdot (\mathbf{max}(P) - \mathbf{min}(P)) + \mathbf{mean}(P) \tag{5.6}$$

5.1.3 Adding noise to data

Noise is the common thing in point cloud data. The origin of the noise could be environmental conditions such as dust, fog, and other particles in the air. Also, the noise appears due to the unperfectness of the lidar and other sensors creating point clouds.

In Experiment 6.4 we investigate how models perform on data with different amounts of artificial noise. For that, we need to generate noise which could be similar to the real world. We propose to use two types of noise which are widely used (Hermosilla, Ritschel, and Ropinski, 2019; Lv and Li, 2020; Rakotosaona et al., 2020):

- Gaussian noise;
- outlier noise.

Gaussian noise

The Gaussian noise adds noise to the initial points in the point cloud P. With probability of p the Gaussian noise (σ, μ) is added to the point $p \in P$. In this way, we simulate the unperfectness of the detecting device when the device makes measurements with some fraction of error. The generation process is described in Algorithm 2

Algorithm 2: Adding Gaussian noise to point cloud (Uchida, 2021)

Result: human point cloud with Gaussian noise - $\hat{P_{human}}$

```
for every point p_i ∈ P do
    if uniRand() < Prob_noise then
        Set p_x to p_x + Gaus(σ, μ) ;
        Set p_y to p_y + Gaus(σ, μ) ;
        Set p_z to p_z + Gaus(σ, μ) ;
    end
end
```

An example of applying the Gaussian noise to the point cloud could is shown in Figure 5.5

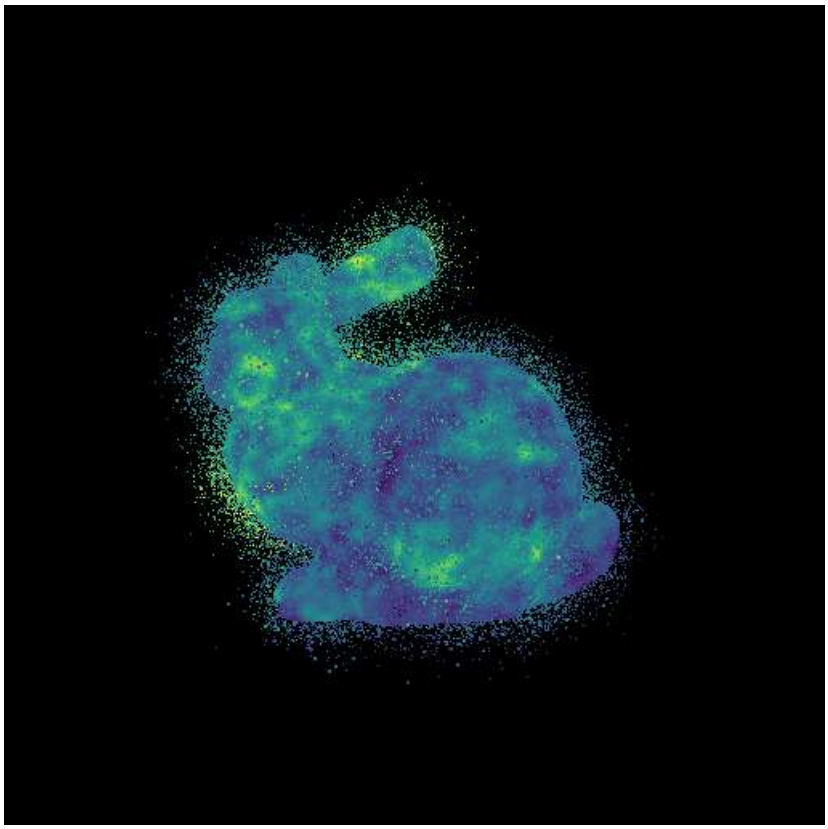

FIGURE 5.5: Example of pllying the Gaussian noise to the point cloud (Uchida, 2021)

Outlier noise

The outlier noise adds new points to the initial point cloud P. The amount of outlier noise is defined by the fraction of the noise points compared to the initial number of human points. The noise point is taken from the uniform distribution $U(a, b)$. Where a and b are min-max values from the initial point cloud. In this way, outlier noise uniformly fills the bounding box of the initial point cloud.

An example of the addition of the outlier noise to the point cloud is shown in the Figure 5.6

5.2 Network architecture

In this section, we cover the network architecture for the task of human pose estimation, the loss functions, and the process of training.

We designed a network based on the work of Wu et al., 2020 to estimate the human pose. In this work, we perform a one-stage training strategy compared to the previous two-sage approach. The aforementioned work performs two-stage training. The first stage consists of training an autoencoder part (based on capsules) of the network to recreate the input human point cloud. The second stage takes an autoencoder with frozen layers from the first stage and trains a regression part of the model which uses capsule outputs as input data. Such an approach requires two steps in the training pipeline and increases the number of hyperparameters to tune the network. In this work, we propose a one-stage training scheme which is described in Section 5.3.

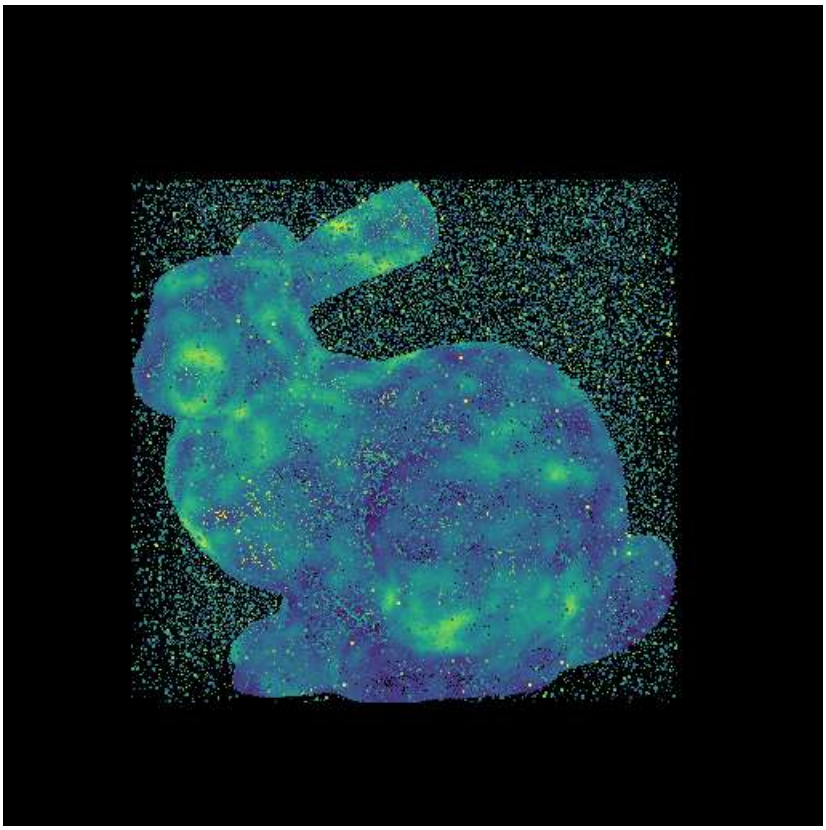

FIGURE 5.6: Example of addition the outlier noise to the point cloud (Uchida, 2021)

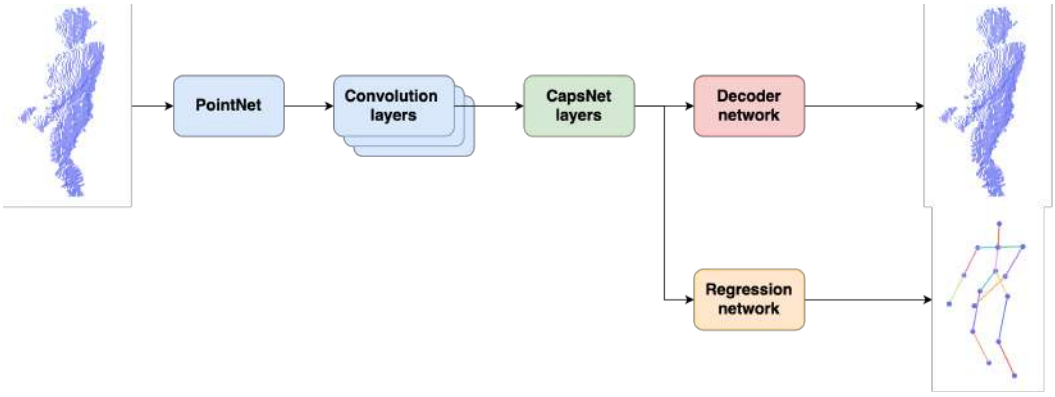

FIGURE 5.7: Network architecture

5.2.1 Loss aggregation

In this work, we propose one-stage training based on loss aggregation. We will compare the results of the training with both a two-stage training scheme and a simple "loss sum" aggregation strategy in Section 6.2. We use Chamfer distance as a loss function for encoder-decoder part. The Chamfer distance is defined as:

$$d_{CD}\left(P_1, P_2\right) = \sum_{x \in P_1} \min_{y \in P_2} \|x - y\|_2^2 + \sum_{y \in P_2} \min_{x \in P_1} \|x - y\|_2^2 \tag{5.7}$$

Where $P_1 \in R^3$ and $P_2 \in R^3$ represent point clouds from the human point cloud and the point cloud recovered from latent capsules.

For the regression part of the network, we use classical MSE[2] loss. The regression loss is defined as:

$$\text{Loss}_E(T, J) = \frac{1}{N} \sum_{i=1}^{N} \left(\|j_i - F\left(t_i\right)\|^2 \right) + \lambda \|w\|^2 \tag{5.8}$$

Where T is a feature vector from latent capsules. J is the ground truth human joints. And F is the regression network.

Since two losses have a different magnitude of values and at different points of time could differ in terms of "picking the low hanging fruit", we need aggregation to mitigate such issues.

The common approach of loss aggregation is usage of the weighted sum of losses (Redmon et al., 2016; Cipolla, Gal, and Kendall, 2018; Zhao et al., 2018) (e.g. $L = (af_1 + bf_2)$ where L is loss function, a, b - loss weights, and f_1, f_2 - aggregated loss functions). These approaches struggles from increased huperparameter size since weights for losses should be chosen by hand.

We have selected an aggregation based on logarithms of losses' product. Such a decision is based on the assumption that our losses greater than 0 and the magnitude is not drastic. The logarithm, in this case, mitigates the issue of vanishing gradient in the case when two losses approaching zero.

The result loss function is shown in Formula 5.9.

$$L = \log\left(\text{Loss}_E(T, J)\right) + \log\left(d_{CD}\left(P_1, P_2\right)\right) \tag{5.9}$$

The methodology of comparing different approaches is covered in 5.3. The experiment comparison between one-stage training vs two-stage training is covered in 6.2.

5.3 One stage network training

In this section, we describe the method of training the model in one stage manner compared to two staged in Wu et al., 2020.

The work Wu et al., 2020 proposes to split the training phase into two steps: the auto-encoder training phase and the regression training phase.

In the first phase regression network is frozen (Figure 5.7) and PointNet, convolution layers, capsule layers, and decoder are trained.

In the second phase, Wu et al., 2020 propose taking the trained auto-encoder part from the previous step, freeze, and train only the regression part.

[2] mean squared error

Our proposed solution is to train two forks of the network simultaneously. To achieve this we propose to aggregate two losses from each network's heads (decoder and regression). Doing this we expect to improve the performance of the network in terms of speed and accuracy.

The details of losses aggregation are covered in Section 5.4.

To verify the hypothesis that such an approach should improve the model's performance we will compare two cases: classical two-stage, and our proposed one stage. As a benchmark, we will use the ITOP dataset with the mAP metric described in Formula 4.2. The experiment results could be found in Section 5.3.

5.4 How noise affects model's performance

In this section, we describe a pipeline of models' evaluation with noisy data. The noise generation technique is described in Section 5.1.3. We compare our model with SOTA model for the task of human pose estimation on ITOP (Haque et al., 2016) dataset. Current SOTA is PoseNet (Moon, Chang, and Lee, 2018) model.

The assumption we try to test is that a capsule-based network should perform better with noisy data due to the internal representation of the point cloud inside the network. The internal representation is held by latent capsules. In this way, the capsule-based network, due to its reconstruction part, should act like a denoiser which should help in the regression part of the task. The expected behavior of the model is shown in Figure 6.9. Contrariwise, models which don't contain internal representation should perform worse on the same dataset.

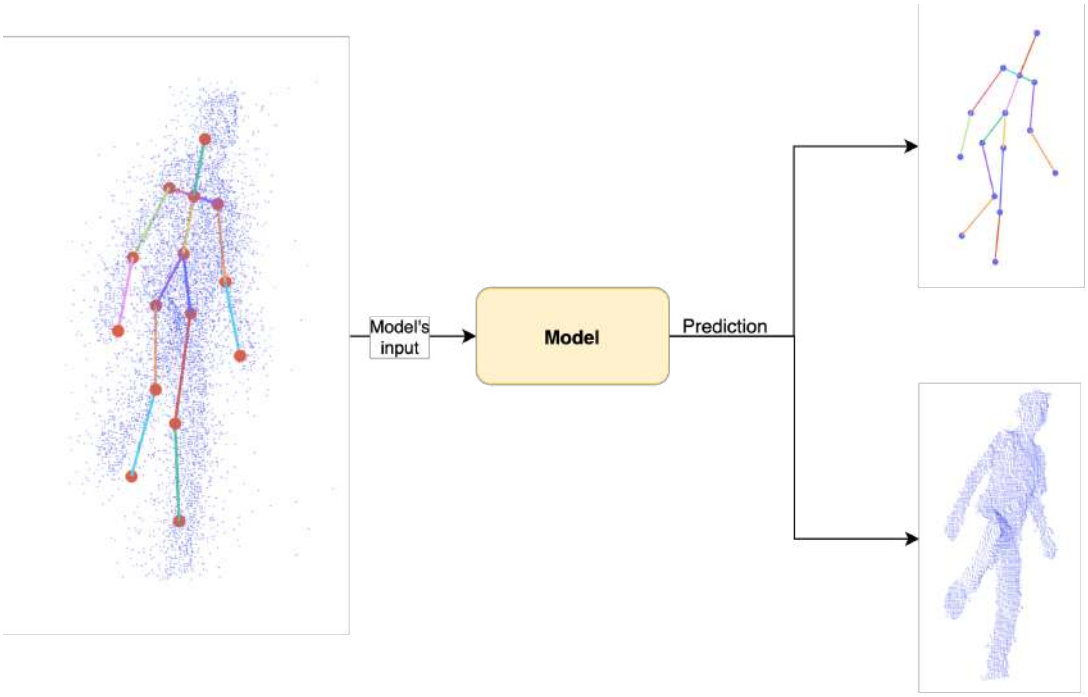

FIGURE 5.8: Capsule model trained on noiseless dataset expected to act like denoiser on noisy data

To test the above assumption we use a capsule-based model which was trained on the dataset without noise and evaluate it on an evaluation set with a different amount of uniform and Gaussian noise. Also, we use the pre-trained PointNet

model as a reference. We will measure the absolute drop of mAP (described in Formula 4.2) with each increase of the noise amount.

5.5 Influence of dataset size on model's performance

In this section, we describe the pipeline of model evaluation for the experiment with a reduced training set.

In this part, we try to test the assumption that capsule-based models need less training data to perform equally on the evaluation set compared to non-capsule-based models. This assumption was proven for classification tasks on the MNIST dataset in Sabour, Frosst, and Hinton, 2017, but wasn't reproduced on more complex data.

The capsule-based model due to its internal representation should faster come up with point cloud patterns compared to classical approaches. In this way, capsule-based models need less training data to perform equally on the evaluation dataset.

To test the above assumption we will train capsule-based model and SOTA model (PoseNet) with different fractions of training set till convergence. Then, we will evaluate the performance of both models on the evaluation set. We will measure the absolute drop of mAP (described in Formula 4.2) with each fraction of the training set.

The dataset reduction is done by removing unique people from the set. In the ITOP training dataset there are 16 unique people which result in 39,795 point clouds.

We will use the model's performance on 100% dataset (full train dataset) as reference values for both models. Then we will decrease the train dataset size to $15/16 - 93\%$, $12/16 - 75\%$, and $8/16 - (50\%)$ from the overall size. The schematic visualization of dataset reduction is shown in Figure 5.9.

FIGURE 5.9: Visualization of different dataset fractions. Each squad represent subset of point cloud for human model in dataset. Yellow - subset is used for training. Red - subset is not used for training. Green - test subsets

Chapter 6

Experiments

In this chapter we describe experiment results based on methodology stated in Chapter 5. We use ITOP dataset (Haque et al., 2016) for comparison with different models. To compare our results with existing models we use as a reference such models: RF (Shotton et al., 2011), RTW (Ho Yub Jung et al., 2015), IEF (Carreira et al., 2016), VI (Haque et al., 2016), REN (Chen et al., 2020), Pose-net (Moon, Chang, and Lee, 2018), all of which were overviewed in Chapter 2.

In Section 6.2 we show our results with one stage straining scheme compared to the two-stage described in Section 5.3.

In Section 6.3 we show the performance of the one-stage training model on the ITOP dataset (Haque et al., 2016). Also, we compare our results with state-of-the-art models for the same task.

In Section 6.4 we show a comparison of models' performance on a dataset with additional noise described in Section-5.4. As a reference model we use SOTA model - Pose-net (Moon, Chang, and Lee, 2018).

In Section 6.5 we compare our proposed model with the SOTA Pose-net model with truncated dataset size to assess models' convergence.

6.1 Experiment setup

In this section, we describe the basic setup for our experiments. Some experiments have additional changes to the training/evaluation pipelines described in dedicated subsections.

All experiments were performed on a server with 8 CPU core, 1 GPU card Nvidia Tesla P100 (16.2 Gb of video memory), and 60 Gb of RAM. As a training framework, Pytorch 1.8.1 was used based on CUDA 10.1 video driver. This setup was powered by the Google Colab backend.

All point clouds from the ITOP dataset were preprocessed according to Section 5.1. For threshold filtering and normalization we use NumPy Python library, and for point cloud clusterization and human extraction we use C++ PCL (*strawlab/python-pcl* 2021) library with Python bindings. After preprocessing we reduce the number of points in the human cloud to 2,000 to speed up the model's training time. The Adam optimizer (Kingma and Ba, 2017) was used for all experiments. The initial learning rate is $lr = 10^{-4}$ with learning rate decay of 10^{-1} each 30 epochs. The average number of epoch to a fully trained model is 80, after this time of epoch our model starts to oscillate on the plateau. The average training time in such a setup is 16 machine hours. The batch size of 12 was used. After each epoch of the training dataset, we run evaluation pipelines to measure the model's performance, on each evaluation run we save such metrics: reconstruction loss (Formula 5.7), regression loss (Formula 5.8), total loss (Formula 5.9), mAP for 10 cm distance (Formula 4.2).

For visualization purposes, we also save input and reconstructed point clouds (P and $\hat{P}$ accordingly).

6.2 Effectiveness of one stage training

In this section, we compare our proposed approach of a one-stage training scheme with the original two-stage one (Wu et al., 2020).

The main objective of one-stage training is to reduce the number of hyperparameters and speed up training via aggregation strategy of reconstruction and regression losses in the network. There are two experiments conducted on the original architecture described by Wu et al., 2020 and our proposed method described in Section 5.3.

At first, we train capsule networks with a two-stage setup. For the first stage, we use the setup described in Section 6.1 and freeze the regression part of the network, thus we train only the reconstruction part. In this setup, we run 70 epochs and save the model. The second stage uses weights of the model from the first step, but now freeze feature extractor and reconstruction parts of the network and train only regression. In this setup, only 20 epochs are needed to fully converge.

The second experiment adopts our proposed solution with loss aggregation. In this experiment, we train all the networks at once, without freezing any layers. This experiment is done according to Section 6.1.

The comparison of the two experiments is shown in Table 6.1.

TABLE 6.1: The comparison of one stage and two stage training pipeline (based on Formula 4.2 metric)

Dataset	Two stage training	One state training (our method)	Difference
ITOP front view	79.6%	83.1%	3.5%
ITOP side view	70.8%	74.2%	3.4%

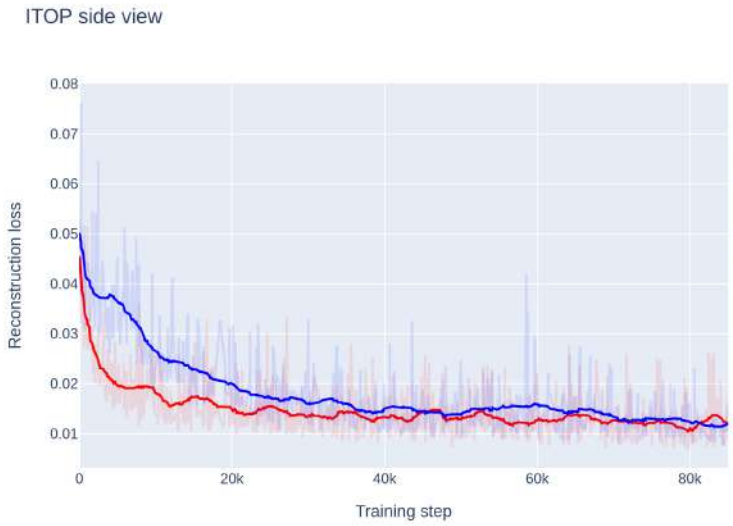

FIGURE 6.1: Reconstruction loss function for two stage (red) model and one stage model(blue)

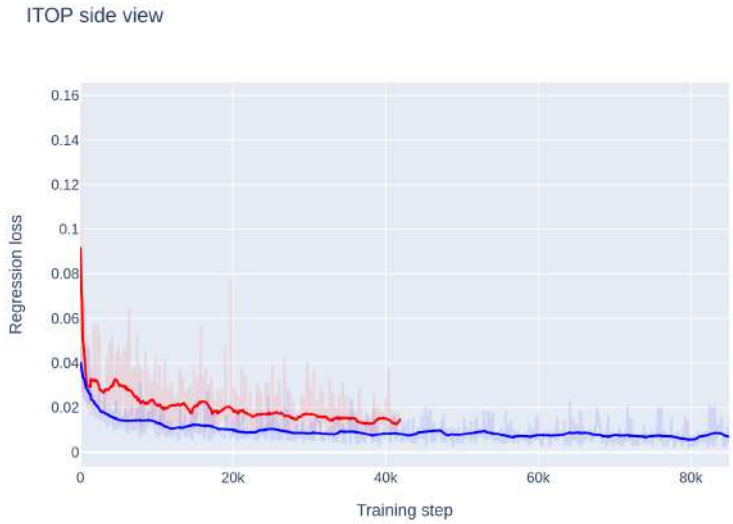

FIGURE 6.2: Regression loss function for two stage (red) model and one stage model(blue)

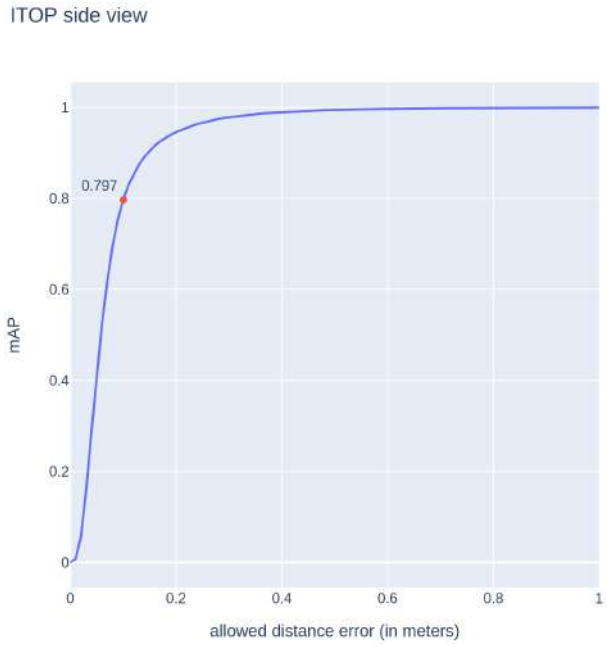

FIGURE 6.3: Two stage model for side view. mAP for different distance errors for two stage model. 10 cm distance mAP is highlighted

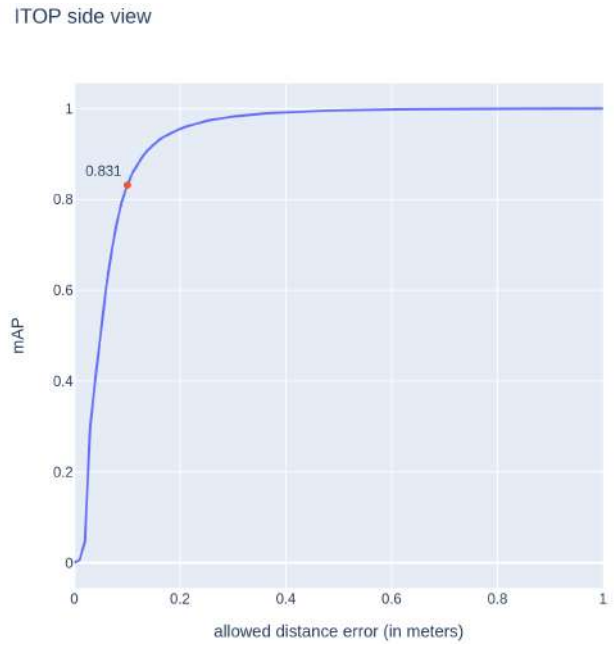

FIGURE 6.4: One stage model for side view. mAP for different distance errors for two stage model. 10 cm distance mAP is highlighted

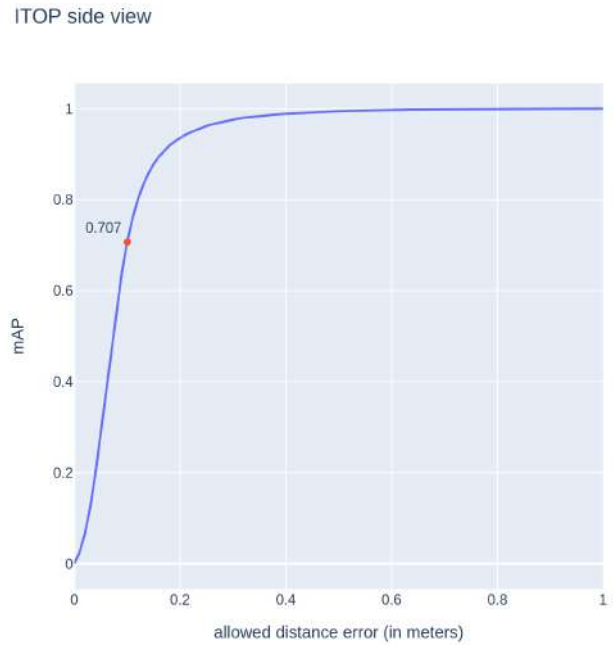

FIGURE 6.5: Two stage model for top view. mAP for different distance errors for two stage model. 10 cm distance mAP is highlighted

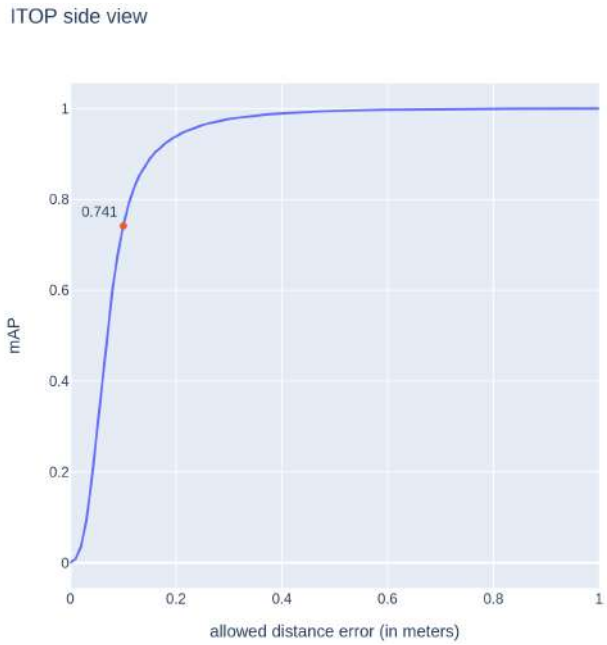

FIGURE 6.6: One stage model for top view. mAP for different distance errors for two stage model. 10 cm distance mAP is highlighted

6.3 Results on ITOP dataset

We have conducted training and evaluation of our capsule-based model on the ITOP dataset (side and top views). The results of the evaluation, as well as the performance of SOTA models, are shown in Table 6.2 for side view, and in Table 6.3 for top view.

Our proposed model shows competitive results both on ITOP side view and front view. Our model outperforms RF (Shotton et al., 2011), RTW (Ho Yub Jung et al., 2015), IEF (Carreira et al., 2016) models on both datasets, and aslo VI (Haque et al., 2016) on side-view only. The model still underperform REN (Chen et al., 2020), Pose-net (Moon, Chang, and Lee, 2018) models.

Our model gives good results on head, neck, shoulders, torso body parts, and struggle to estimate hands and feet. It's difficult for a model to predict a key joint

TABLE 6.2: Comparison of proposed model with SOTA models on ITOP side view dataset

	mAP (side-view)						
Body part	RF	RTW	IEF	VI	REN-9x6x6	PoseNet	Our model
Head	63.8	97.8	96.2	98.1	98.7	98.29	94.7
Neck	86.4	95.8	85.2	97.5	99.4	99.07	96.9
Shoulders	83.3	94.1	77.2	96.5	96.1	97.18	93.7
Elbows	73.2	77.9	45.4	73.3	74.7	80.42	73.9
Hands	51.3	70.5	30.9	68.7	55.2	67.26	58.0
Torso	65.0	93.8	84.7	85.6	98.7	98.73	97.0
Hip	50.8	80.3	83.5	72.0	91.8	93.23	89.5
Knees	65.7	68.8	81.8	69.0	89.0	91.80	88.1
Feet	61.3	68.4	80.9	60.8	81.1	87.6	80.0
Mean	65.8	80.5	71.0	77.4	84.9	88.74	83.1

TABLE 6.3: Comparison of proposed model with SOTA models on ITOP top view dataset

	mAP (top-view)						
Body part	RF	RTW	IEF	VI	REN-9x6x6	PoseNet	Our model
Head	95.4	98.4	83.8	98.1	98.2	98.4	94.2
Neck	98.5	82.2	50.0	97.6	98.9	98.91	96.0
Shoulders	89.0	91.8	67.3	96.1	96.6	96.87	89.2
Elbows	57.4	80.1	40.2	86.2	74.4	79.16	67.6
Hands	49.1	76.9	39.0	85.5	50.7	62.44	48.9
Torso	80.5	68.2	30.5	72.9	98.1	97.78	94.0
Hip	20.0	55.7	38.9	61.2	85.5	86.91	79.3
Knees	2.6	53.9	54.0	51.6	70.0	83.28	80.3
Feet	0.0	28.7	62.4	51.5	41.6	69.62	67.4
Mean	47.4	68.2	51.2	75.5	75.5	83.44	74.1

that is located in a relatively small point cloud (compared to whole-body).

In the Figure 6.7 we could see how the model reconstructs the human body with each epoch. On first epochs, the model outputs just a random point cloud, and with each new epoch, it trains to reproduce the input point cloud. As we can see from d) in Figure 6.7 on epoch 60 the model's approximation is pretty solid.

6.4 How noise affects models' performance

In this section, we evaluate a capsule-based model (proposed solution) and a reference model (PoseNet) on a dataset with a different amount of noise. The experiment's description is covered in Section 5.4.

We use Gaussian noise and uniform noise as a mixin to the initial data. For the Gaussian noise, we add noise with a probability of $\frac{1}{3}$ (to roughly 30% of points). We use two types of values for σ - 0.1 for mild noise and 0.2 for more aggressive.

For uniform distribution we use constant number of additional points $N_P = 300$. That is roughly 15% of additional points out of mean human point cloud size. In Figure 6.8 is shown an example of human point clouds with additional noise. We don't add noise to ground truths key joints.

As for the capsule-based model we use the model trained on the ITOP side view dataset from the Experiment 6.3. We will use the ITOP side view dataset as a benchmark for this experiment.

As for reference model we use pretrained[1] PoseNet.

We inference the ITOP side view dataset with different noise parameters for both models and measure mAP for each experiment. All experiments are collected in Table 6.4.

[1] https://github.com/mks0601/V2V-PoseNet_RELEASE

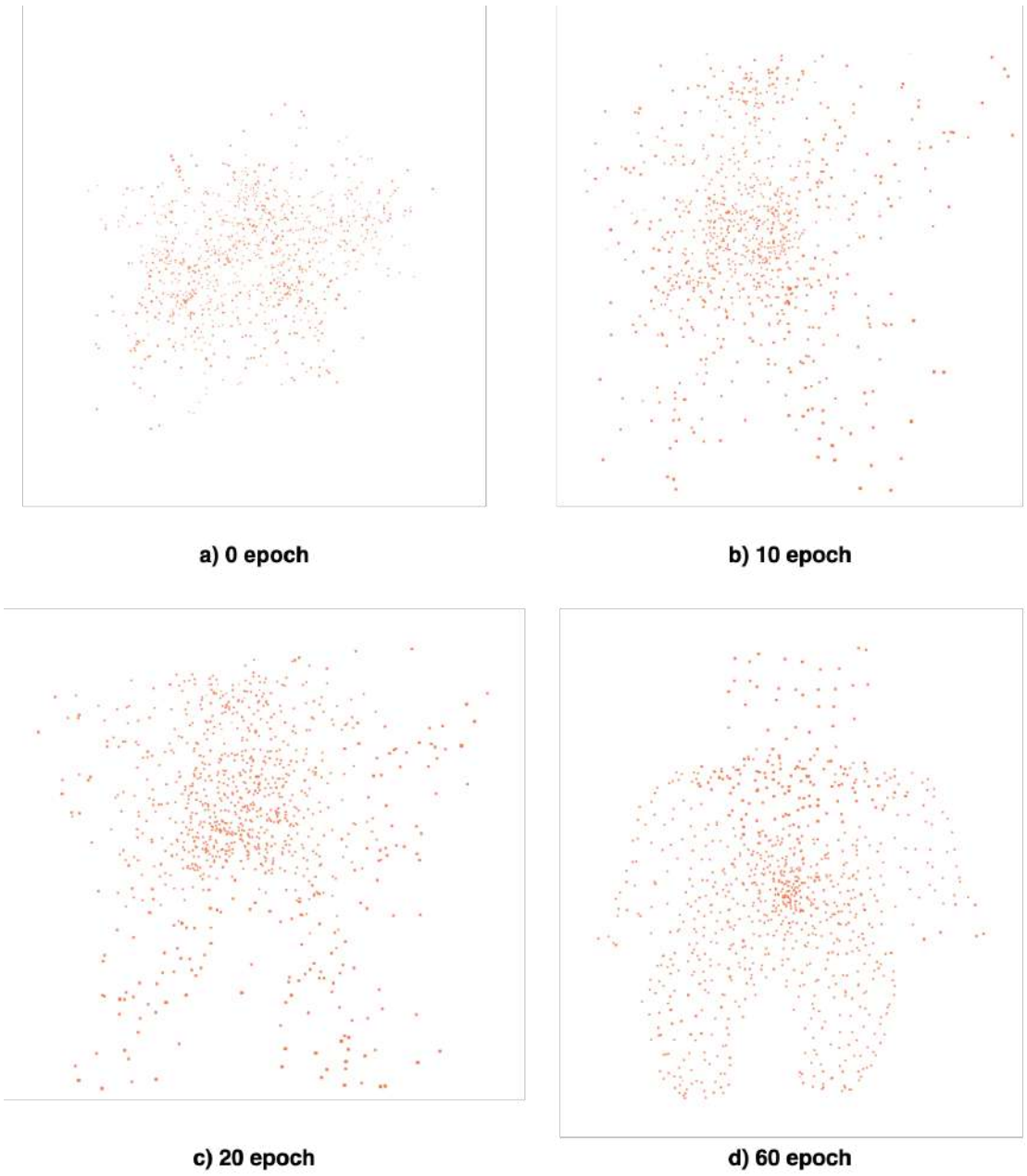

FIGURE 6.7: One stage model for top view. mAP for different distance errors for two stage model. 10 cm distance mAP is highlighted

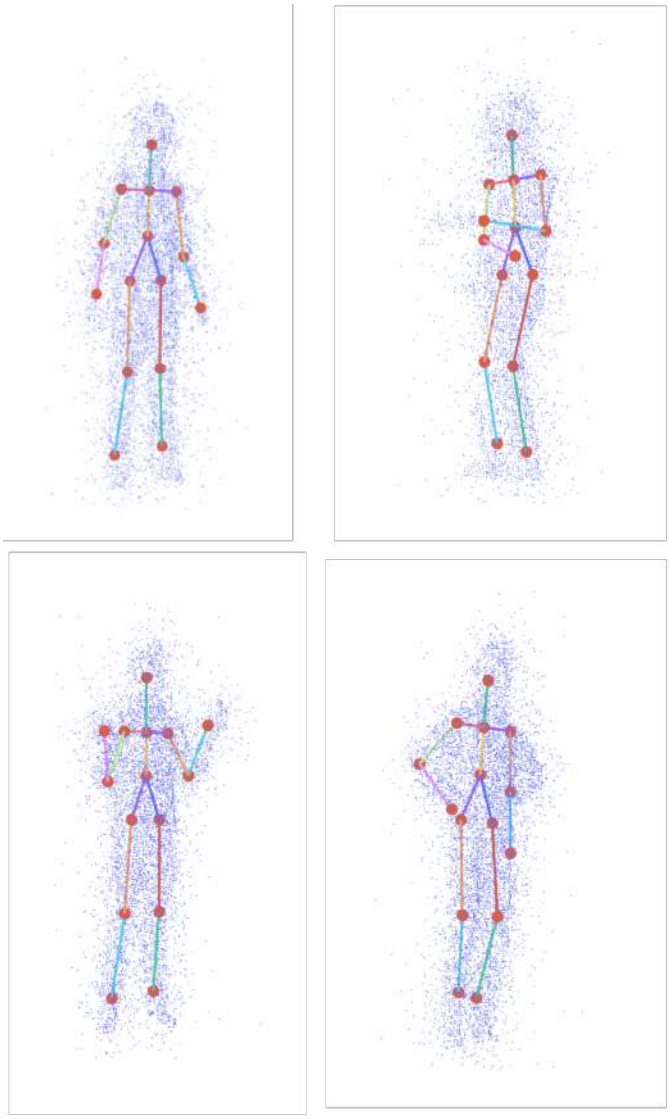

FIGURE 6.8: Example of human points clouds with added noise. $\sigma - 0.1$, $N_P = 300$

TABLE 6.4: The comparison of capsnet model and PoseNet on dataset with different amount of noise on ITOP side view dataset

Noise setup	PoseNet	CapsNet	PoseNet difference	CapsNet difference
No noise (reference)	86.2%	83.1%	0%	0%
$\sigma = 0.1, N_P = 0$	82.1%	81.6%	4.1%	1,5%
$\sigma = 0, N_P = 300$	84.9%	81.9%	1,3%	1,2%
$\sigma = 0.1, N_P = 300$	80.9%	79.2%	5,3%	3,9%
$\sigma = 0.2, N_P = 300$	74.5%	70.4%	11,7%	12,7%

The reference value (dataset without noise) for the capsule-based model is 83.1% mAP, and for PoseNet is 86.2% mAP. The PoseNet's results are different compared to Table 6.2 - 88.74% in table vs 86.2% in our experiment. Such difference could be due to the fact that we don't use an ensemble of PoseNet models for the evaluation. This difference isn't significant for the methodology of our experiment since we use the difference of mAP as a metric.

As we can see capsule-based model performs better on noisy datasets compared to the reference model. For $\sigma = 0.1$ the drop of mAP for capsule network is $1,5\%$ vs $4,1\%$ in PoseNet.

The only experiment where PoseNet outperforms capsule-based network is with significant amount of noise $\sigma = 0.2$ and $N_P = 300$. For this experiment capsule network has a drop of $12,7\%$ of mAP vs $11,7\%$ for PoseNet.

To verify an assumption made in Section 5.4 that capsule-based model could act like a denoiser due to internal latent space, we plot an input noisy point cloud and reconstructed point cloud by the network. (Figure 6.9)

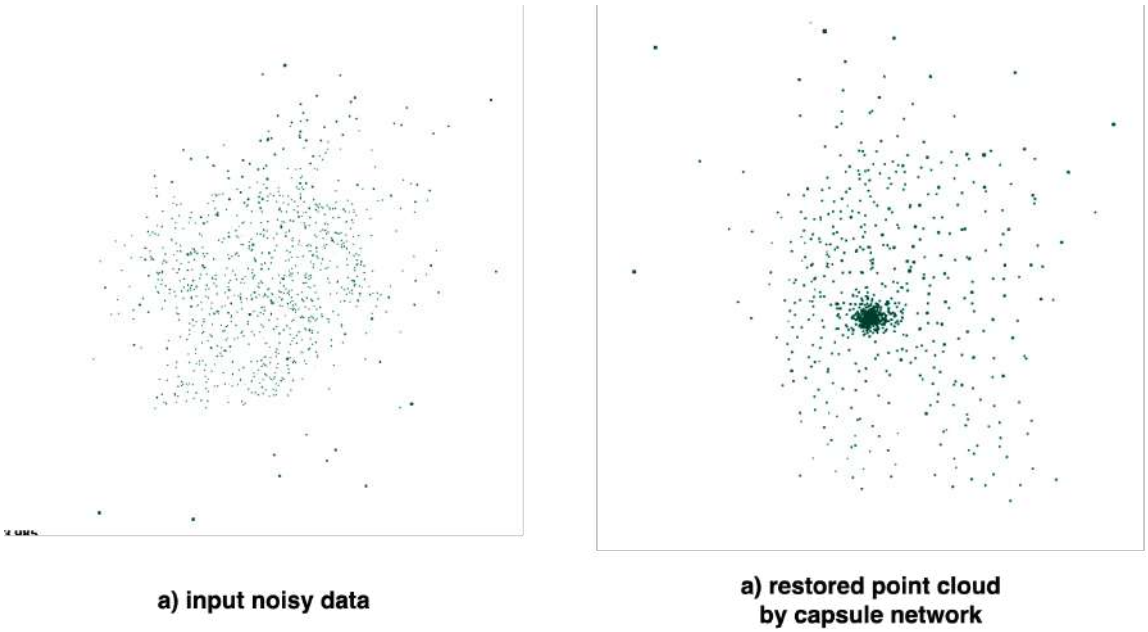

FIGURE 6.9: Example of noisy point cloud (left), and restored point cloud by capsule network (left)

Based on the visual analysis we could say that a capsule-based network filters the majority amount of noisy data and potentially could be used as a denoiser.

6.5 Models' performance with the lack of data

In this section, we evaluate the capsule-based model with different amounts of training data. Also, we compare the performance with the SOTA model - PoseNet.

For the capsule-based model, we run training pipelines with different training dataset sizes (16/16 as a reference, 15/16, 12/16, and 8/16). The pipeline setup is described in Section 6.1. After each run of training, we measure the mAP for 10 cm distance on the test dataset.

For PoseNet we use the same approach of retraining the model with different train sizes. The training code was used from the original[2] implementation of the model. Also, we use default parameters for the PoseNet network.

As a benchmark dataset, we use the ITOP side view.

The comparison table for the set of experiments could be found in Table 6.5.

TABLE 6.5: The comparison of capsnet model and PoseNet trained on different amound of data (ITOP side view)

Dataset size	PoseNet	CapsNet	PoseNet difference	CapsNet difference
16/16 (full dataset)	86.2%	83.1%	0%	0%
15/16	86.0%	82.8%	0.2%	0,3%
12/16	80.2%	74.3%	6,0%	8,8%
8/16	63.7%	56.0%	30,2%	27,1%

As we can see from the results capsule-based model underperform compared to PoseNet model. The only experiment where the capsule-based model shows a better mAP difference is the case with 8/16 of the dataset. In this experiment capsule model shows $27,1\%$ of mAP drop compared to $30,2\%$ in PoseNet.

Taking everything into account we can't prove a hypothesis that a capsule-based network works better in data lack environment compared to another SOTA model. However, this statement is hold only for current experiment setup and could change on other datasets.

[2] https://github.com/mks0601/V2V-PoseNet_RELEASE

Chapter 7

Conclusions

7.1 What was done?

In this section we sum up our work which was concentrated on main objectives:

- Design a capsule-based model for the task of human pose estimation using point cloud data. Compare model's performance with SOTA models
- Verify the hypothesis that capsule-based models perform better compared to others on noisy data
- Verify the hypothesis that capsule-based models better generalize and need less training data compared to other models

Further in the text, we describe our results for each objection.

Capsule-based model for human pose estimation

We designed a capsule-based neural network for human pose estimation using point clouds. We took the work Wu et al., 2020 of as a baseline architecture for our problem. We proposed the new method of one-stage training of the model which showed improved performance both in training speed and in the model's accuracy. We compared our model with SOTA models on the well-known dataset. Our proposed network shows competitive results outperforming such architectures as use as a reference such models: RF (Shotton et al., 2011), RTW (Ho Yub Jung et al., 2015), IEF (Carreira et al., 2016), and VI (Haque et al., 2016). But still our proposed model underperform in comparison with REN (Chen et al., 2020) and Pose-net (Moon, Chang, and Lee, 2018) models.

Capsule-based model and noisy data

We designed a methodology and conducted experiments to verify the hypothesis that capsule-based networks are more noise agnostic in comparison with non-capsule-based models.

We have evaluated our proposed models and the SOTA PoseNet model on the ITOP dataset with different amounts of artificial noise (from Gaussian and uniform distributions). Based on our experiments capsule-based model shows better results in most of the cases in comparison with PoseNet. Also, we have visually proved that a capsule-based model could denoise the point cloud which we associate with the ability of the capsule network to hold internal representation.

Capsule-based model and train dataset size

We designed a methodology and conducted experiments to verify the hypothesis that capsule-based networks need fewer data to train in comparison with non-capsule-based models.

We have evaluated the capsule-based model and SOTA PoseNet model on the ITOP dataset. We have conducted multiple experiments where we reduce the training dataset size. Based on our results we don't see any proofs that capsule-based models could better generalize the data and thus need less training data.

7.2 Future work

Algorithms on point clouds are a really promising field. Capsule-based networks are interesting models for this particular task. Capsule-based models could hold an internal representation of the input data and thus reuse it to improve performance for a variety of tasks like classification, segmentation, regression, etc.

We have shown that capsule-based models are applicable for the task of human pose estimation, and could show compatible results.

As future work we see such areas:

Data preprocessing

We proposed a method where we preprocess the initial point cloud and extract a person point cloud, and only then use it as an input to the network. Such an approach is not applicable to the real-world data since our techniques are greatly tight to the ITOP dataset.

To better adapt a pipeline to the real-world use case the special extraction DNN models could be used (Shi et al., 2020; Yang, Luo, and Urtasun, 2019). Such models don't need handcrafted thresholds and parameters thus could be used for more diverse datasets.

New datasets

Our research was focused on the ITOP dataset. But the EVAL dataset mentioned in Section-4 also could be used for the evaluation.

New loss aggregation strategies

In our work, we use the product of losses as an aggregation strategy. Also, we use the logarithm to mitigate the issue of vanishing gradient. This approach is quite primitive and some advanced loss aggregations methods could be used (*Optimizing Multiple Loss Functions with Loss-Conditional Training*).

Bibliography

Cao, Zhe et al. (May 2019). "OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields". In: *arXiv:1812.08008 [cs]*. arXiv: 1812.08008. URL: http://arxiv.org/abs/1812.08008 (visited on 12/25/2020).

Carreira, Joao et al. (June 2016). "Human Pose Estimation with Iterative Error Feedback". In: *arXiv:1507.06550 [cs]*. arXiv: 1507.06550. URL: http://arxiv.org/abs/1507.06550 (visited on 05/23/2021).

Chen, Xinghao et al. (June 2020). "Pose Guided Structured Region Ensemble Network for Cascaded Hand Pose Estimation". In: *Neurocomputing* 395. arXiv: 1708.03416, pp. 138–149. ISSN: 09252312. DOI: 10.1016/j.neucom.2018.06.097. URL: http://arxiv.org/abs/1708.03416 (visited on 05/23/2021).

Cheraghian, Ali and Lars Petersson (Nov. 2018). "3DCapsule: Extending the Capsule Architecture to Classify 3D Point Clouds". In: *arXiv:1811.02191 [cs]*. arXiv: 1811.02191. URL: http://arxiv.org/abs/1811.02191 (visited on 12/14/2020).

Cipolla, Roberto, Yarin Gal, and Alex Kendall (June 2018). "Multi-task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics". en. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT, USA: IEEE, pp. 7482–7491. ISBN: 978-1-5386-6420-9. DOI: 10.1109/CVPR.2018.00781. URL: https://ieeexplore.ieee.org/document/8578879/ (visited on 05/21/2021).

Diaz Barros, Jilliam Maria, Frederic Garcia, and Désiré Sidibé (Mar. 2015). "Real-Time Human Pose Estimation from Body-Scanned Point Clouds". In: vol. 1. DOI: 10.5220/0005309005530560.

Duarte, Kevin, Yogesh S. Rawat, and Mubarak Shah (May 2018). "VideoCapsuleNet: A Simplified Network for Action Detection". In: *arXiv:1805.08162 [cs]*. arXiv: 1805.08162. URL: http://arxiv.org/abs/1805.08162 (visited on 12/14/2020).

Fang, Hao-Shu et al. (Feb. 2018). "RMPE: Regional Multi-person Pose Estimation". In: *arXiv:1612.00137 [cs]*. arXiv: 1612.00137. URL: http://arxiv.org/abs/1612.00137 (visited on 12/25/2020).

Gritsevskiy, Andrew and Maksym Korablyov (Apr. 2018). "Capsule networks for low-data transfer learning". In: *arXiv:1804.10172 [cs]*. arXiv: 1804.10172. URL: http://arxiv.org/abs/1804.10172 (visited on 05/23/2021).

Guo, Hengkai et al. (July 2017). "Towards Good Practices for Deep 3D Hand Pose Estimation". In: *arXiv:1707.07248 [cs]*. arXiv: 1707.07248 version: 1. URL: http://arxiv.org/abs/1707.07248 (visited on 05/19/2021).

Haque, Albert et al. (July 2016). "Towards Viewpoint Invariant 3D Human Pose Estimation". In: *arXiv:1603.07076 [cs]*. arXiv: 1603.07076. URL: http://arxiv.org/abs/1603.07076 (visited on 05/23/2021).

Hermosilla, Pedro, Tobias Ritschel, and Timo Ropinski (Oct. 2019). "Total Denoising: Unsupervised Learning of 3D Point Cloud Cleaning". In: *arXiv:1904.07615 [cs]*. arXiv: 1904.07615. URL: http://arxiv.org/abs/1904.07615 (visited on 05/23/2021).

Ho Yub Jung et al. (June 2015). "Random tree walk toward instantaneous 3D human pose estimation". en. In: *2015 IEEE Conference on Computer Vision and Pattern*

Recognition (CVPR). Boston, MA, USA: IEEE, pp. 2467–2474. ISBN: 978-1-4673-6964-0. DOI: 10.1109/CVPR.2015.7298861. URL: http://ieeexplore.ieee.org/document/7298861/ (visited on 05/23/2021).

Kakillioglu, Burak et al. (2020). "3D Capsule Networks for Object Classification With Weight Pruning". In: *IEEE Access* 8, pp. 27393–27405. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2971950. URL: https://ieeexplore.ieee.org/document/8984369/ (visited on 12/14/2020).

Kingma, Diederik P. and Jimmy Ba (Jan. 2017). "Adam: A Method for Stochastic Optimization". In: *arXiv:1412.6980 [cs]*. arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980 (visited on 05/23/2021).

LaLonde, Rodney and Ulas Bagci (Apr. 2018). "Capsules for Object Segmentation". In: *arXiv:1804.04241 [cs, stat]*. arXiv: 1804.04241. URL: http://arxiv.org/abs/1804.04241 (visited on 12/14/2020).

Liu, Yipeng et al. (Dec. 2020). "Point Cloud Quality Assessment: Large-scale Dataset Construction and Learning-based No-Reference Approach". In: *arXiv:2012.11895 [eess]*. arXiv: 2012.11895. URL: http://arxiv.org/abs/2012.11895 (visited on 05/20/2021).

Lv, Chaohui and Min Li (Oct. 2020). "Point Cloud Denoising Algorithm Based on Noise Classification". In: *2020 International Conference on Culture-oriented Science Technology (ICCST)*, pp. 123–127. DOI: 10.1109/ICCST50977.2020.00029.

Maturana, D. and S. Scherer (Sept. 2015). "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. DOI: 10.1109/IROS.2015.7353481.

Moon, Gyeongsik, Ju Yong Chang, and Kyoung Mu Lee (Aug. 2018). "V2V-PoseNet: Voxel-to-Voxel Prediction Network for Accurate 3D Hand and Human Pose Estimation from a Single Depth Map". In: *arXiv:1711.07399 [cs]*. arXiv: 1711.07399 version: 3. URL: http://arxiv.org/abs/1711.07399 (visited on 05/23/2021).

Nearest neighbor search (May 2021). en. Page Version ID: 1020801927. URL: https://en.wikipedia.org/w/index.php?title=Nearest_neighbor_search&oldid=1020801927 (visited on 05/17/2021).

Optimizing Multiple Loss Functions with Loss-Conditional Training. en. URL: http://ai.googleblog.com/2020/04/optimizing-multiple-loss-functions-with.html (visited on 05/24/2021).

PCL - Euclidean Cluster Extraction. URL: https://pcl.readthedocs.io/en/latest/cluster_extraction.html (visited on 05/17/2021).

Qi, Charles R. et al. (June 2017a). "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space". In: *arXiv:1706.02413 [cs]*. arXiv: 1706.02413. URL: http://arxiv.org/abs/1706.02413 (visited on 12/14/2020).

Qi, Charles R. et al. (Apr. 2017b). "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *arXiv:1612.00593 [cs]*. arXiv: 1612.00593. URL: http://arxiv.org/abs/1612.00593 (visited on 12/14/2020).

Qin, Yao et al. (Feb. 2020). "Detecting and Diagnosing Adversarial Images with Class-Conditional Capsule Reconstructions". In: *arXiv:1907.02957 [cs, stat]*. arXiv: 1907.02957. URL: http://arxiv.org/abs/1907.02957 (visited on 12/14/2020).

Rakotosaona, Marie-Julie et al. (2020). "PointCleanNet: Learning to Denoise and Remove Outliers from Dense Point Clouds". en. In: *Computer Graphics Forum* 39.1. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13753, pp. 185–203. ISSN: 1467-8659. DOI: https://doi.org/10.1111/cgf.13753. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13753 (visited on 05/23/2021).

Redmon, Joseph et al. (May 2016). "You Only Look Once: Unified, Real-Time Object Detection". In: *arXiv:1506.02640 [cs]*. arXiv: 1506.02640. URL: http://arxiv.org/abs/1506.02640 (visited on 05/21/2021).

Riegler, Gernot, Ali Osman Ulusoy, and Andreas Geiger (Apr. 2017). "OctNet: Learning Deep 3D Representations at High Resolutions". In: *arXiv:1611.05009 [cs]*. arXiv: 1611.05009. URL: http://arxiv.org/abs/1611.05009 (visited on 12/25/2020).

Rovai, Marcelo (Aug. 2020). *Realtime Multiple Person 2D Pose Estimation using TensorFlow2.x*. en. URL: https://towardsdatascience.com/realtime-multiple-person-2d-pose-estimation-using-tensorflow2-x-93e4c156d45f (visited on 12/25/2020).

Sabour, Sara, Nicholas Frosst, and Geoffrey E. Hinton (Nov. 2017). "Dynamic Routing Between Capsules". In: *arXiv:1710.09829 [cs]*. arXiv: 1710.09829. URL: http://arxiv.org/abs/1710.09829 (visited on 12/14/2020).

Shi, Shaoshuai et al. (Mar. 2020). "From Points to Parts: 3D Object Detection from Point Cloud with Part-aware and Part-aggregation Network". In: *arXiv:1907.03670 [cs]*. arXiv: 1907.03670. URL: http://arxiv.org/abs/1907.03670 (visited on 05/24/2021).

Shotton, Jamie et al. (June 2011). "Real-Time Human Pose Recognition in Parts from Single Depth Images". In: vol. 56, pp. 1297–1304. ISBN: 978-3-642-28660-5. DOI: 10.1109/CVPR.2011.5995316.

strawlab/python-pcl (May 2021). original-date: 2012-05-16T10:26:35Z. URL: https://github.com/strawlab/python-pcl (visited on 05/23/2021).

Su, Hang et al. (Dec. 2015). "Multi-view Convolutional Neural Networks for 3D Shape Recognition". en. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. Santiago, Chile: IEEE, pp. 945–953. ISBN: 978-1-4673-8391-2. DOI: 10.1109/ICCV.2015.114. URL: http://ieeexplore.ieee.org/document/7410471/ (visited on 12/25/2020).

Uchida, Tomomasa (Mar. 2021). *tom-uchida/Add_Noise_to_Point_Cloud*. original-date: 2018-09-22T08:50:24Z. URL: https://github.com/tom-uchida/Add_Noise_to_Point_Cloud (visited on 05/09/2021).

Wang, Peng-Shuai et al. (July 2017). "O-CNN: octree-based convolutional neural networks for 3D shape analysis". en. In: *ACM Transactions on Graphics* 36.4, pp. 1–11. ISSN: 0730-0301, 1557-7368. DOI: 10.1145/3072959.3073608. URL: https://dl.acm.org/doi/10.1145/3072959.3073608 (visited on 12/25/2020).

Wang, Yiwei et al. (2020). "Capsule Networks Showed Excellent Performance in the Classification of hERG Blockers/Nonblockers". English. In: *Frontiers in Pharmacology* 10. Publisher: Frontiers. ISSN: 1663-9812. DOI: 10.3389/fphar.2019.01631. URL: https://www.frontiersin.org/articles/10.3389/fphar.2019.01631/full (visited on 05/23/2021).

Wei, Xin, Ruixuan Yu, and Jian Sun (June 2020). "View-GCN: View-Based Graph Convolutional Network for 3D Shape Analysis". en. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA: IEEE, pp. 1847–1856. ISBN: 978-1-72817-168-5. DOI: 10.1109/CVPR42600.2020.00192. URL: https://ieeexplore.ieee.org/document/9156567/ (visited on 12/25/2020).

Wu, Yiqi et al. (Oct. 2020). "3D Capsule Hand Pose Estimation Network Based on Structural Relationship Information". en. In: *Symmetry* 12.10. Number: 10 Publisher: Multidisciplinary Digital Publishing Institute, p. 1636. DOI: 10.3390/sym12101636. URL: https://www.mdpi.com/2073-8994/12/10/1636 (visited on 12/25/2020).

Yang, Bin, Wenjie Luo, and Raquel Urtasun (Mar. 2019). "PIXOR: Real-time 3D Object Detection from Point Clouds". In: *arXiv:1902.06326 [cs]*. arXiv: 1902.06326. URL: http://arxiv.org/abs/1902.06326 (visited on 05/24/2021).

Yang, Jiancheng et al. (Apr. 2019). "Modeling Point Clouds with Self-Attention and Gumbel Subset Sampling". In: *arXiv:1904.03375 [cs]*. arXiv: 1904.03375. URL: http://arxiv.org/abs/1904.03375 (visited on 12/25/2020).

Yu, Tan, Jingjing Meng, and Junsong Yuan (June 2018). "Multi-view Harmonized Bilinear Network for 3D Object Recognition". en. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT: IEEE, pp. 186–194. ISBN: 978-1-5386-6420-9. DOI: 10.1109/CVPR.2018.00027. URL: https://ieeexplore.ieee.org/document/8578125/ (visited on 12/25/2020).

Zhao, Hang et al. (Apr. 2018). "Loss Functions for Neural Networks for Image Processing". In: *arXiv:1511.08861 [cs]*. arXiv: 1511.08861. URL: http://arxiv.org/abs/1511.08861 (visited on 05/21/2021).

Zhao, Yongheng et al. (July 2019). "3D Point Capsule Networks". In: *arXiv:1812.10775 [cs]*. arXiv: 1812.10775. URL: http://arxiv.org/abs/1812.10775 (visited on 12/14/2020).

Zhou, Y., H. Dong, and A. E. Saddik (Oct. 2020). "Learning to Estimate 3D Human Pose From Point Cloud". In: *IEEE Sensors Journal* 20.20. Conference Name: IEEE Sensors Journal, pp. 12334–12342. ISSN: 1558-1748. DOI: 10.1109/JSEN.2020.2999849.