Exploring the Spatial Characteristics of MARS

Assessing the Impact of Neural Net Depth Increase and PointNet Architecture Integration on MARS Performance

Kevin Hoxha
Supervisors: Marco Zuñiga Zamalloa, Girish Vaidya
EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
22/06/2024

Name of the student: Kevin Hoxha
Final project course: CSE3000 Research Project
Thesis committee: Marco Zuñiga Zamalloa, Girish Vaidya, Michael Weinmann

Abstract

The modern workplace often exposes individuals to privacy risks, such as the unauthorised visibility of their computer screens. MARS (mmWave-based Assistive Rehabilitation System for Smart Healthcare), coupled with VideowindoW screens, offers an innovative solution to these threats by using mmWave radar to reconstruct human poses and estimate the position of 19 key joints. This enables the screens to become opaque based on the viewer's position, ensuring privacy. Although originally designed as a rehabilitation system, MARS can be utilised for its pose estimation capabilities to enhance workplace privacy. This research explores two modifications to the MARS architecture to assess their impact on the system's accuracy and performance. Specifically, we modify the MARS architecture by increasing the depth of its convolutional neural network (CNN) and integrating the PointNet architecture. Results establish that an optimal CNN configuration with two convolutional and two dense layers, followed by the output layer, modestly improves joint location estimation. However, integrating PointNet does not improve performance, likely due to PointNet's limitations in capturing the necessary local structural details of point clouds. These findings inform future research of possible improvements when leveraging the MARS dataset in the fields of privacy enhancement and smart healthcare applications.

Keywords

Convolutional Neural Networks (CNNs), MARS, millimetre wave (mmWave), PointNet, 3D Human Pose Reconstruction

1 Introduction

The modern workplace often exposes individuals to privacy risks, particularly when colleagues have unrestricted visibility of their screens. This issue is crucial because it compromises the confidentiality of sensitive information and violates individuals' right to privacy in professional settings. Furthermore, in industries where intellectual property and proprietary data are essential, such as finance, healthcare, and technology, the potential consequences of unauthorised screen visibility can be critical, ranging from data breaches to legal liabilities. Thus, addressing these privacy risks is essential for fostering a secure work environment.

One proposed solution involves surrounding workstations with VideowindoW[1] screens that resemble transparent displays but can be programmed to selectively obscure areas corresponding to the gaze directions of the unauthorised individuals, as illustrated in Figure 1. This approach prevents individuals from viewing others' monitors directly, as their line of sight will be met with an obscured region, such as a displayed image or black square, instead of the intended screen content.

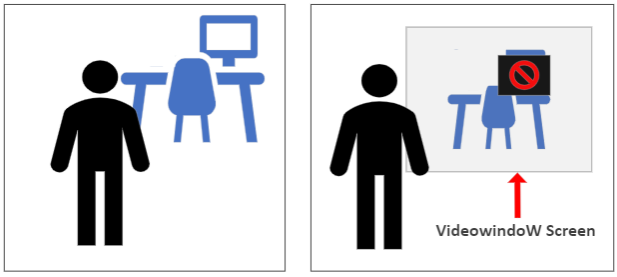

Figure 1: *Illustration of screen exposure in the workplace (left), and the usage of VideowindoW screens to block the intruders' line of sight to the monitor by displaying a black square (right).*

However, to know which part of the screen needs to be obscured, the gaze direction can be estimated through pose reconstruction. While pose reconstruction can be achieved through the use of video cameras [1], opting for this method is costly and limited in practicality, as video cameras require strict lighting settings and introduce additional privacy risks, such as the potential for people monitoring. To alleviate these concerns, we can instead use millimetre wave (mmWave) radars [2], which transmit mmWave signals to produce point clouds of people moving in the range of the radar. The collected point cloud data can then be used to reconstruct the poses of individuals in front of these radars.

There are multiple proposed solutions in the area of pose reconstruction through mmWave-based point clouds, such as **mmPose** [3], **mmBody** [4], **mmMesh** [5], **Pantomime** [6], **SquiggleMilli** [7], **MARS** [8], and others [9]. While all these solutions have their benefits and drawbacks, this paper will solely focus on enhancing MARS (mmWave-based Assistive Rehabilitation System for Smart Healthcare), since it is a low-cost, low-power, and accurate pose reconstruction system with a large *one-of-its-kind* movement dataset that has 40,083 frames [8]. MARS can estimate the accurate location of 19 human joints by first mapping the 5D time-series point cloud from the mmWave radar to a lower dimension and then using a convolution neural network (CNN) [8]. There are currently no existing studies aiming to enhance the architecture of MARS by exploring its spatial features or integrating it with PointNet.

This research aims to explore the spatial dynamics of the MARS framework, evaluating how increasing the neural net depth and integrating the PointNet architecture impact overall model performance. **PointNet** [10] is a neural network architecture specifically designed for processing raw point clouds, commonly used in tasks such as object classification and seg-

[1] https://www.videowindow.eu/

mentation. We aim to investigate whether integrating PointNet enhances the CNN architecture's performance in joint estimation tasks, contributing to more accurate and robust pose reconstruction methods.

To address the research question, this paper presents all the necessary background information in Section 2, outlines the research methodology in Section 3, and presents the results after establishing the experimental setup in Section 4. Section 5 interprets the findings and analyses the results. Finally, Section 6 addresses the ethical considerations of the research, before concluding and outlining potential recommendations for future work in Section 7.

2 Background

This section provides the necessary background information regarding the main concepts explored in this paper.

2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a type of artificial neural network designed for processing structured grid data, such as images. Thus, they are mostly used for computer vision tasks, including object classification, segmentation, and pattern recognition. The structure of a CNN comprises three main elements [11]:

1. **The convolutional layers** apply filters or *kernels* to input data to extract important features. These filters (kernels) slide over the input data and perform mathematical operations (convolutions) to detect patterns like edges, textures, or shapes. The higher the number of convolutional layers, the more high-level patterns the CNN can detect, however, this comes with a substantial increase in computational complexity [12].
2. **The pooling layers** come after the convolutional layers, and they apply downsampling to lower the number of parameters in the input while preserving the most important features. Two of the most common pooling methods are max pooling and average pooling [13].
3. **The dense/fully-connected (FC) layers** are placed at the end of the CNN. They perform either classification or regression tasks, based on the use case of the CNN. They are responsible for translating the extracted features from the previous layers into outputs/predictions.

An example of a simple CNN structure is shown in Figure 2 [14].

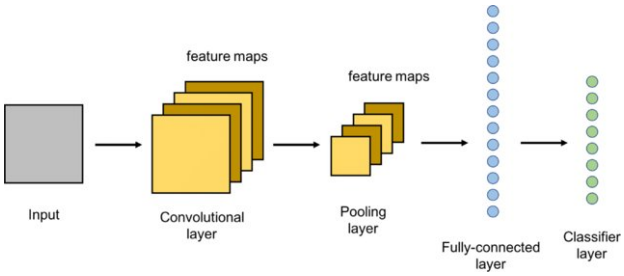

Figure 2: *A simple CNN architecture illustrating its main components [14]. The architecture starts with the input, followed by convolutional and pooling layers for feature extraction and dimensionality reduction, respectively. It concludes with fully-connected and classifier layers that perform high-level reasoning and classification based on the features extracted.*

2.2 MARS

MARS (mmWave-based Assistive Rehabilitation System for Smart Healthcare) is a rehabilitation system that monitors patients' movement in home environments and provides real-time feedback [8]. This system utilises 5D mmWave radar point cloud data to reconstruct 3D human poses by estimating 19 key joint coordinates. The use of mmWave radars [2] ensures that no video images or facial information are collected, hence mitigating any privacy issues. The structure of MARS can be seen in Figure 3.

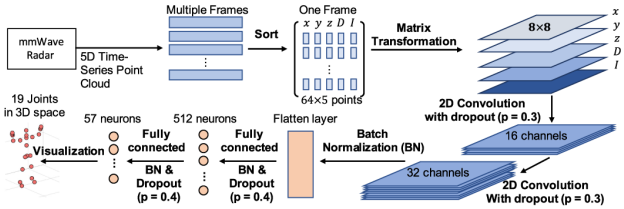

Figure 3: *Point cloud preprocessing and CNN architecture of MARS [8]. The input data is first sorted and stacked on an 8×8 feature map with 5 channels. Then, the feature map passes through two convolutional layers with 16 and 32 channels, respectively, before being flattened and passed through the fully connected layer. The fully connected layer is composed of one dense layer with 512 neurons, and the output layer of 57 neurons, corresponding to the 3D locations of 19 joints.*

MARS preprocesses the input data and passes it to a CNN with two convolutional layers and one dense layer. This dense layer connects to the output layer, which estimates the coordinates of the key joints. The paper on MARS [8] justifies that the best model configuration is the one that uses Batch Normalisation (BN) and a Dropout layer with a probability of 0.4 or 0.3 after every layer. The employment of BN avoids the internal covariate shift [15], while the Dropout layers avoid excessive dependency on specific neurons [16].

2.3 PointNet

PointNet [10] is a deep neural network that directly consumes point clouds without requiring conversion to 3D voxel grids or images, thus preserving the original spatial distribution of the data. It is used for multiple 3D recognition tasks, including object classification, part segmentation and semantic segmentation.

One of the key features of PointNet [10] is its ability to maintain invariance to permutations of the input points. This characteristic makes it robust when dealing with raw point clouds because they are naturally unordered. PointNet [10] achieves this invariance through a symmetric function, specifically a max pooling layer, which down-samples the convolutional outputs by selecting the maximum value within each small window [13], ensuring that the outcome of the neural network is the same regardless of the order of the input points.

Moreover, PointNet [10] is designed to handle common transformations in 3D data, such as translations and rotations. It employs a mini-network called the "T-Net" [17] which regresses a spatial transform matrix to align the points into a canonical, orientation-normalised coordinate frame. This feature enhances the model's robustness and generalisation capabilities across various spatial orientations of objects.

An overview of the structure of PointNet [10] is shown in Figure 4. It is composed of multilayer perceptions (MLPs) with learnable parameters, and the max pooling function. It takes an input of the shape $n \times 3$, where n is the number of points and 3 is the dimension of the data, applies input and feature transformations (T-Nets), and then aggregates point features by max pooling, resulting in the global features [10]. This resulting 1024-dimensional global feature vector describes the features of the input and can be used for classification or segmentation tasks.

3 Methodology

As mentioned in Section 2, the MARS architecture incorporates a CNN with two convolutional layers and one dense layer, followed by the output layer that estimates 19 joint positions. While this architecture is very efficient and effective [8], this research seeks to enhance it by exploring different configurations of the spatial characteristics of the CNN, and then replacing this CNN with PointNet and considering the differences in performance.

To tackle the research question, it is best to split it into two subquestions:

- **SQ1**: What is the performance impact of increasing the depth of the neural network in the MARS architecture? (Section 3.1)
- **SQ2**: What is the performance impact of integrating PointNet instead of using the current CNN in the MARS architecture? (Section 3.2)

3.1 Increasing the Depth of the Neural Network in MARS

While the paper on MARS [8] conducts various ablation studies to demonstrate the necessity of each component, it does not justify the chosen depth of the CNN in the MARS architecture. This subquestion focuses on the methodical approach of conducting such a study.

To investigate the influence of the neural network depth on MARS performance, we initially determine the optimal number of convolutional layers in the CNN architecture. To achieve this, we must manually change the number of layers in the CNN and select the best-performing configuration [18]. Thus, four configurations of the MARS model are trained, each with 1, 2, 3, and 4 convolutional layers, respectively, and 1 dense layer. The number of filters in the first convolutional layer is 16, and doubles after every convolutional layer, consistent with the MARS paper. To mitigate the influence of randomness, each model configuration undergoes ten training iterations, and the performance metrics are averaged.

Following the determination of the optimal convolutional layer configuration, a similar methodology is employed to decide the optimal number of dense layers forming the Fully Connected (FC) layer. Four MARS model configurations are trained with the determined optimal number of convolutional layers and 1, 2, 3, and 4 dense layers, respectively. The number of neurons in the first dense layer is set at 512, halving after each subsequent layer. The output layer consists of 57 neurons, corresponding to the 3-dimensional coordinates of the 19 joint locations. Consistency is maintained in training details and evaluation metrics, similar to the approach used for evaluating different convolutional layer configurations. Since ten iterations do not yield consistent results, each model undergoes fifteen training iterations instead, and the evaluation metrics are averaged to derive appropriate approximations. The specific training details are showcased in Section 4.

3.2 Integrating PointNet in MARS

Another promising way to enhance MARS is to replace the CNN it employs with the PointNet [10] model. As mentioned in Section 2, PointNet's architecture, designed to handle sparse and noisy data efficiently, employs the max pooling symmetric function to focus on dominant features, thus enhancing its capability to learn complex spatial relationships.

To integrate PointNet in MARS, some adjustments

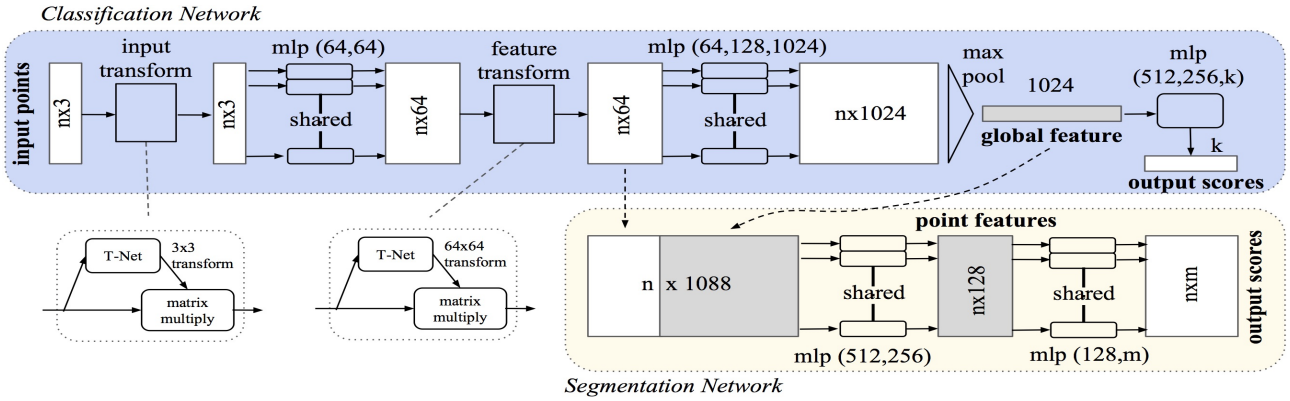

Figure 4: *An overview of the original architecture of PointNet [10]. This diagram showcases the structure, featuring an input transformation network (T-Net), a sequence of multi-layer perceptrons (MLPs) applied to each point independently, a feature transformation network (T-Net), and a max pooling layer that aggregates features into a global descriptor. After the global feature, the model is specialised for either classification or segmentation.*

must be made to the MARS dataset and the PointNet architecture. MARS requires the data to be in an $8 \times 8 \times 5$ shape since the CNN performs 2D convolutions on the input. PointNet, conversely, is specialised in directly consuming point clouds sequentially. Thus, the input data must be reshaped to a 64×5 array.

Then, to effectively adapt PointNet for use in the MARS system, data preprocessing plays a critical role in maximising the network's performance and accuracy in joint estimation tasks. **Firstly**, some noise is added to the point cloud data, since studies suggest that this enhances the model's robustness against real-world sensor noise [19]. The added noise has a Gaussian distribution with a mean of 0 and a standard distribution of 0.2 times the standard distribution of the input point cloud. This ensures that the added noise can simulate variations consistent with the data's natural fluctuations. **Secondly**, MARS reorders all input points ascendingly according to their x, y, and z, coordinates. PointNet, however, is permutation invariant, meaning that it is invariant to the input order. Therefore, the 64×5 input data should be shuffled in a random order to take advantage of the permutation invariance property.

These preprocessing steps are essential to exploit PointNet's potential in accurately estimating joint locations. They align with the inherent capabilities of the network to manage and interpret complex spatial information from sparse and irregular input data.

Lastly, since Section 2 mentioned that PointNet is typically used for classification or segmentation tasks and MARS estimates joint locations, PointNet needs to be adapted for regression. To adjust for this, the classification network of PointNet (Figure 4) is adopted until the global feature aggregation, with the exception that the feature transformation T-Net is removed for computational simplicity. Then, the MLP that is used for classification is replaced with a 4-layer MLP for regression that produces a final linear output layer. This layer has 57 neurons, corresponding to the 3 dimensions of the 19 key joints.

An overview of the modified PointNet architecture is illustrated in Figure 5 and the final MARS architecture after PointNet integration is shown in Figure 6. They illustrate all the modifications mentioned in this section together with the sizes of the data at different parts of the architecture.

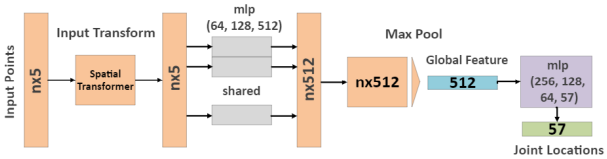

Figure 5: *Architecture of the modified PointNet model. The model takes an input of size $n \times 5$, processes it through the input transformation T-Net, and feeds it to a shared 3-layer multi-layer perceptron (MLP) with layers of size 64, 128, and 512. Then it performs max pooling and aggregates the data into a 512-dimensional global feature vector, which is fed into an MLP with layers sized 256, 128, 64, and, lastly, 57.*

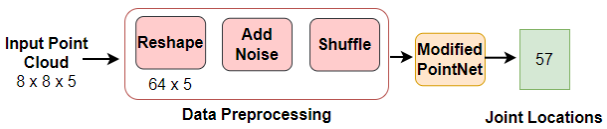

Figure 6: *Architecture of MARS after incorporating the modified PointNet model. The model takes an input point cloud of size $8 \times 8 \times 5$, preprocesses the data in multiple steps to fit PointNet, passes it through the PointNet model, and outputs the 57 three-dimensional coordinates of the 19 key joints.*

3.3 Loss Function & Evaluation Metrics

To maintain consistency with the MARS paper [8], the evaluation metrics that are employed in MARS are also employed in this research. This enables an

appropriate comparison between the results of this paper and that of MARS.

The loss function that MARS proposes to train the models is the mean squared error (MSE) between the reference positions x_i, y_i, z_i of joint i, $1 \leq i \leq N_J$ from the ground truth obtained by the Kinect V2 sensor [20], where N_J is the number of tracked joints, and the estimations $\hat{x}_i$, $\hat{y}_i$, $\hat{z}_i$ from the model. This function is illustrated below:

$$Loss_{coor} = \frac{\sum_{i=1}^{N_J}(x_i - \hat{x}_i)^2 + \sum_{i=1}^{N_J}(y_i - \hat{y}_i)^2}{3N_J} + \frac{\sum_{i=1}^{N_J}(z_i - \hat{z}_i)^2}{3N_J}$$

Furthermore, MARS's accuracy is evaluated using two metrics: the mean absolute error (MAE), and root mean squared error (RMSE). These metrics represent the displacement between the ground truth and the model's estimations in centimetres (cm). The mathematical equations of how they are calculated can be found below:

$$MAE = \frac{\sum_{i=1}^{N_J}|\hat{x}_i - x_i| + \sum_{i=1}^{N_J}|\hat{y}_i - y_i| + \sum_{i=1}^{N_J}|\hat{z}_i - z_i|}{3N_J}$$

$$RMSE = \frac{\sqrt{\frac{1}{N_J}\sum_{i=1}^{N_J}(x_i - \hat{x}_i)^2} + \sqrt{\frac{1}{N_J}\sum_{i=1}^{N_J}(y_i - \hat{y}_i)^2}}{3} + \frac{\sqrt{\frac{1}{N_J}\sum_{i=1}^{N_J}(z_i - \hat{z}_i)^2}}{3}$$

4 Experimental Setup and Results

Following the described methodology in Section 3, this section describes the utilised dataset, and provides the specific evaluation setup used in this research to allow the reader to replicate the results. Finally, the obtained evaluation results are presented.

4.1 Dataset

The dataset used in this research is the one that MARS presented [8]. It is a large-scale dataset containing mmWave point clouds with well-labelled joints. The dataset has 10,000 data frames per user, each containing data for 19 joints, where each user performed ten different kinds of rehabilitation movements in front of a Kinect V2 sensor and a mmWave radar. The data points have five dimensions, namely the 3D coordinates, Doppler velocity, and reflection intensity. In total, there are 2.28 million reference data points from Kinect V2 and 3.81 million data points from mmWave data.

4.2 Experimental Setup

The dataset followed the same split as the MARS paper: *60% for **training** (24,066 frames), 20% for **validation** (8,033 frames), and 20% for **testing** (7,984 frames).*

For the first research subquestion (**SQ1**), consistent with the findings of the ablation study in the original paper, each convolutional and dense layer in the CNN of MARS is followed by a Dropout layer with 30% and 40% dropout probability, respectively, and Batch Normalisation (BN). While testing for the optimal number of convolutional layers, the Fully Connected (FC) Layer maintains a consistent structure across all models. To determine the optimal number of dense layers, we also maintain a consistent number of convolutional layers. This minimises performance discrepancies resulting from changes in other parts of the CNN architecture.

For the second research subquestion (**SQ2**), the convolutional and dense layers in our modified PointNet implementation are followed by BN, but not Dropout. BN is employed because it helps speed up training and enhances the overall performance of the network by reducing the internal covariate shift. Dropout is not included because adding a Dropout layer after each convolutional layer and dense layer did not improve the performance of the model.

The training settings for both subquestions reflect those in the MARS paper, employing an Adam optimiser [21] with a learning rate of 0.001, batch size of 128, and 150 epochs. Since PointNet takes considerably longer to train and reaches saturation quickly, early stopping with a patience of 20 epochs is implemented. Early stopping is a widely used method to prevent over-training neural networks when parameter updates no longer begin to yield improvements on a validation set [22].

For **SQ1** training is executed on one of DelftBlue's supercomputer's[2] Intel XEON CPUs, whereas for **SQ2** training is executed on a Google Colab T4 GPU. The training concludes when the validation loss converges at 0.01.

4.3 Experimental Results

For **SQ1**, models with 1, 2, 3, and 4 convolutional layers are trained following the experimental setup described in Section 4.2. The validation MAE, loss, and time taken to train each model can be found in Figure 7, Figure 8, and Figure 9, respectively. It can be deduced from these plots that the lowest validation MAE and loss from these four configurations is the model with two convolutional layers. The MARS model with only one convolutional layer underfits the training data, while models with more than two

[2] https://www.tudelft.nl/dhpc/system

convolutional layers overfit the training data, thus performing increasingly worse on the validation data. The time taken to train these models increases exponentially fast with the addition of more layers.

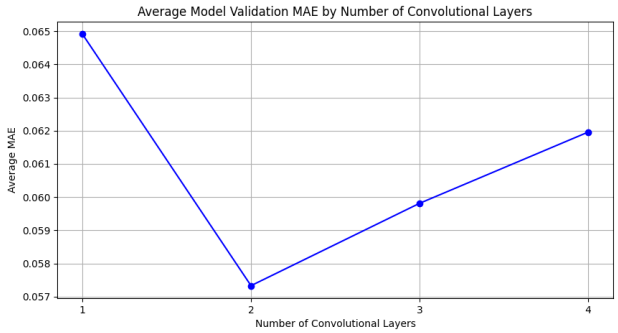

Figure 7: *Average validation MAE for MARS models with 1, 2, 3, and 4 of convolutional layers after 10 runs.*

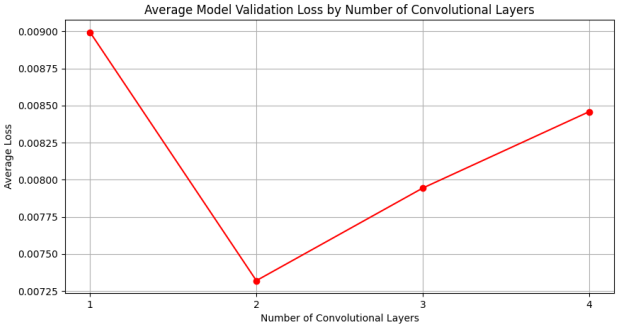

Figure 8: *Average validation loss for MARS models with 1, 2, 3, and 4 convolutional layers after 10 runs.*

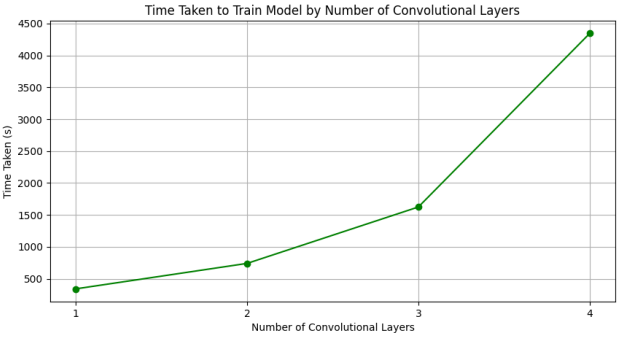

Figure 9: *Average time taken to train MARS models with 1, 2, 3, and 4 convolutional layers after 10 runs.*

After concluding that the optimal number of convolutional layers is two, four configurations with 1, 2, 3, and 4 dense layers, followed by the output layer, are trained to decide the optimal number of dense layers. The validation MAE, loss, and time taken to train each model can be found in Figure 10, Figure 11, and Figure 12, respectively. The lowest validation MAE and loss from these four configurations is the model with two dense layers. These plots showcase a similar trend to the results of the optimal number of convolutional layers, where one dense layer underfits the training data, and more than two dense layers overfit it. The time taken to train the models increases almost linearly with the addition of more layers. Thus, the results show that the optimal setup in MARS, which currently has two convolutional layers, one dense layer, and the output layer, is the setup with two convolutional layers, followed by two dense layers, followed by the output layer. This setup, compared to the results from the MARS paper [8], improves the **MAE** of MARS by 3.2%, and the **RMSE** by 0.09%.

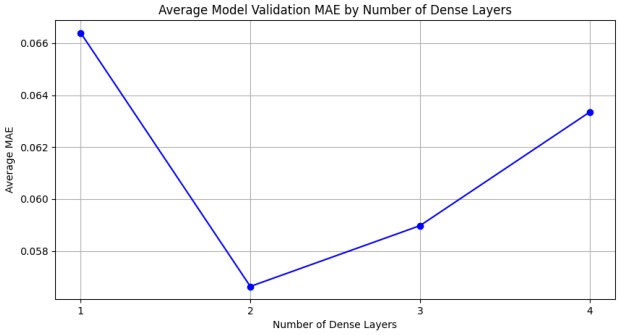

Figure 10: *Average validation MAE for MARS models with 1, 2, 3, and 4 dense layers after 15 runs.*

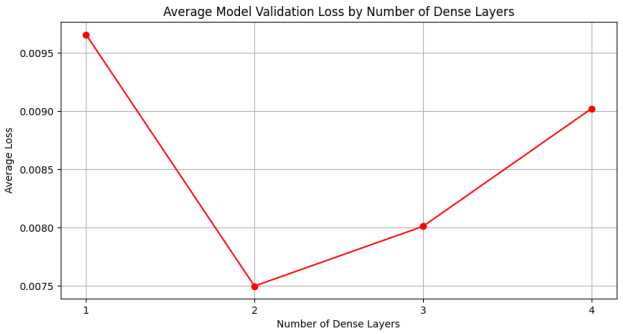

Figure 11: *Average validation loss for MARS models with 1, 2, 3, and 4 dense layers after 15 runs.*

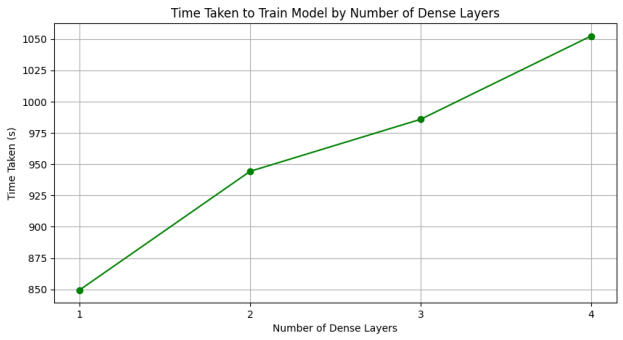

Figure 12: *Average time taken to train MARS models with 1, 2, 3, and 4 dense layers after 15 runs.*

For **SQ2**, the PointNet-integrated model is trained with the configuration outlined in Section 4.2. The

results of the model evaluation in comparison with results from the original MARS model and the optimised MARS model are presented in Figure 13.

These box plots demonstrate that replacing the CNN in MARS with the adapted PointNet described in Section 3.2 does not achieve better performance. The **MAE** is increased by **32.4%**, from 5.87 to 7.77 cm of error, while the **RMSE** is increased by **27.9%**, from 8.10 to 10.36 cm of error. The complete table of the localisation errors for all 19 key joints can be found in Appendix A.

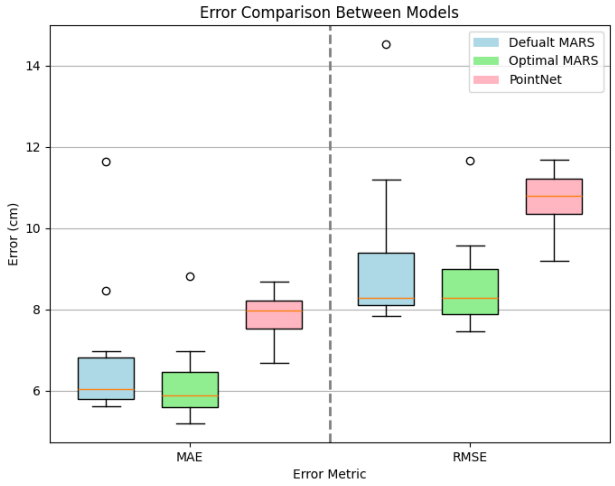

Figure 13: *Comparison of the validation MAE and RMSE in the default MARS, optimal MARS, and PointNet model.*

A summary of the total number of parameters and the training time for each model is showcased in Table 1. The MARS model with PointNet integration has the least number of parameters, followed by the default MARS configuration, followed by the proposed optimal MARS configuration. Regarding the training times, MARS with the default configuration takes the least amount of time to train for 150 epochs, followed by the optimal MARS configuration, followed by PointNet with early stopping (patience of 20 epochs). In contrast, PointNet without early stopping takes around 14 minutes to train and does not improve the accuracy of the model as it overfits the training data.

Model	No. Parameters	Training time
Default MARS	1,085,881	$\sim$ 3 - 4 mins
Optimal MARS	1,203,641	$\sim$ 3.5 - 4.5 mins
PointNet	805,202	$\sim$ 3 - 7 mins

Table 1: *An overview of the total number of parameters and training time for each model. These results were obtained using a T4 GPU from Google Colab.*

5 Discussion

This section analyses the results presented in Section 4 and provides a reflection on what can be concluded. It evaluates the findings in context and highlights significant insights.

5.1 Optimal Setup of MARS

As presented in Section 4.3, the best setup for the MARS system included two convolutional layers, followed by two dense layers, followed by the output layer. The results demonstrated that increasing the number of convolutional layers from one to two significantly improves the model's performance, as evidenced by a decrease in both validation loss and MAE. This suggests that two layers are effective at capturing the essential features in the data, contributing to better generalisation of unseen data. However, further increasing the layers to three and four resulted in poorer performance, with both loss and MAE figures rising. This indicates potential overfitting, where the model learns the noise rather than the underlying data pattern, diminishing its predictive accuracy on new, external datasets. Additionally, the training time increases greatly with more than two convolutional layers, reflecting higher computational demands and operational costs without corresponding benefits in model accuracy.

Subsequently, the study assessed the influence of varying the number of dense layers on the same performance metrics. The findings revealed that two dense layers, as opposed to one dense layer that MARS originally employs, are optimal, where the average model validation loss and MAE are minimised, indicating an enhanced ability of the model to generalise well to new data. Similar to the convolutional layers, adding more than two dense layers leads to an increase in both validation loss and MAE, which can be attributed to overfitting. Moreover, the training duration increases linearly with the number of dense layers. This rise in training time signifies an increase in computational resource utilisation, which becomes unjustifiable given the decrease in model performance with more than two dense layers.

To ensure the reproducibility of results, this setup was run multiple times and the optimal number of convolutional layers was always two. However, when running the models ten times to determine the optimal number of dense layers, the results were inconsistent. Thus, the models were run fifteen times, and the majority of the results suggested that two dense layers perform better.

Despite these findings, the improvements observed with optimal layer configurations- 3.2% in MAE and 0.09% in RMSE- are relatively small and might be at-

tributed to random variation within the experimental data. Such subtle improvements may not represent significant practical enhancements, especially when considering the increased potential for overfitting and the higher computational costs associated with additional layers. However, the box plots in Figure 13 showcase how the optimal MARS, while only modestly improving the model accuracy in comparison with the default MARS, lowers the number of outliers and achieves a smaller interquartile range (IQR). This translates into more stable results with less variance, since the lower the IQR, the more accurate the results.

5.2 MARS with PointNet Integration

The results indicated that PointNet was considerably less effective than both the default and optimal MARS architectures for this study. The primary challenge lies in the task of joint estimation in the human body, which requires precise measurement of small distances in centimetres. A significant limitation of adapting PointNet for such a regression task is its inability to leverage the local structural details of the data. This issue arises from the use of a max pooling layer in PointNet, as shown in Figure 4, which only retains the maximum value from each region, thereby discarding potentially valuable information contained in the other values. Research supports that traditional CNN architectures, like those used in MARS, generally outperform PointNet for tasks involving human pose estimation [23]. Therefore, the observed difference in performance aligns with expectations and reinforces the relevance of CNNs for this type of application.

Furthermore, while PointNet's model has around 200.000 fewer parameters than the default MARS model and 400.000 less than the optimal MARS model, the time taken to train these models does not follow the same pattern, as shown in Table 1. This is due to several factors inherent to its design and the nature of the data it processes. Firstly, PointNet is tailored for unstructured and sparse point cloud data, necessitating additional preprocessing steps. Secondly, the architecture of PointNet includes specialised operations like max pooling across points to handle permutation invariance, which is less computationally efficient than the highly optimised convolution operations used in CNNs. Lastly, PointNet treats data sequentially, while MARS stacks points and treats them in 2D. This can lead to increased training times in comparison with MARS.

6 Responsible Research

Ensuring that the practices are ethical and that others can replicate the results is essential in research. It is vital that any reader can achieve the same outcomes as those presented in this paper. Therefore, to uphold ethical standards and prevent misconduct, this study was conducted with reproducibility in mind, alongside a commitment to ethical research practices.

6.1 Ethical Considerations

The MARS dataset, as described in Section 4.1, contains thousands of frames of point clouds of people captured using a mmWave radar [2]. The main benefit of using a mmWave radar, instead of a video camera, is that it provides anonymous detection. This means that this study does not utilise any sensitive data, and it mitigates any threats to the re-identification of data.

This research also complies with the Netherlands Code of Conduct for Research Integrity [24], which emphasises the importance of honesty, scrupulousness, transparency, independence, and responsibility. The adherence to these principles is demonstrated through a transparent methodology and an impartial analysis of results. We ensure that all data sources are properly referenced.

6.2 Reproducibility

The MARS dataset is public and it is located in an open-source repository[3], where the codebase for this study also forked from. As such, anyone can access the MARS database and codebase, and follow the methodology described in Section 3 to achieve the same results that this paper presents.

PointNet [10] also has numerous publicly available implementations. We document all adjustments from the default PointNet implementation in Section 3, which aims to provide transparency in all choices and reasoning made during all steps of the research.

Furthermore, the experimental setup, such as model parameters and training details, are presented in detail in Section 4.2. This section gives the reader all the necessary technical information to replicate the same results.

7 Conclusions and Future Work

This study aimed to answer the main research question, which explored the spatial characteristics of MARS and reported the performance impact of increasing the neural network depth and integrating the PointNet architecture. The conducted experiments found that the optimal setup for MARS includes two convolutional layers, two dense layers, and the output layer, as opposed to the original setup of MARS, consisting of two convolutional layers, one dense layer, and the output layer. This setup slightly

[3] https://github.com/SizheAn/MARS

improves MARS's accuracy in estimating 19 key joint locations, while producing more consistent results with fewer outliers. Integrating PointNet in MARS, on the other hand, did not improve the performance, due to PointNet's limitations in preserving the local structure information found in point cloud data that is necessary for precise joint estimation.

These findings contribute to research in the field of mmWave-based human pose estimation by exploring the architecture of an existing pose estimation technology, namely MARS. The adaptations and modifications explored in this study show that certain architectural changes in MARS can yield modest performance improvements.

For future work, it is recommended to explore alternative methods to integrate the strengths of PointNet with CNNs, such as developing new models or modifying existing models that can leverage the feature extraction capabilities of PointNet while maintaining the local contextual awareness provided by CNNs. One suggestion is to explore the effectiveness of PointNet++ [25] instead of PointNet since PointNet++ tackles the issue of capturing the local structures. Additionally, investigating the application of these findings in real-world scenarios, such as in-home rehabilitation and workspaces surrounded by VideowindoW displays, could provide deeper insights into their practical effectiveness.

Overall, this study has improved the current understanding of the performance differences between different neural network architecture choices in MARS and their capabilities in handling mmWave data for 3D human pose reconstruction. It has also highlighted the inability of PointNet to properly capture local structures. Following these findings, much remains to be explored in investigating and optimising systems trained with the MARS dataset for practical, real-world applications in smart healthcare and privacy enhancement in the workspace.

Bibliography

[1] A. Yiannakidis, A. Aristidou and Y. Chrysanthou, 'Real-time 3D human pose and motion reconstruction from monocular RGB videos', *Computer Animation and Virtual Worlds*, vol. 30, May 2019. DOI: 10.1002/cav.1887.

[2] C. Iovescu, 'The fundamentals of millimeter wave sensors', 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:30625501.

[3] A. Sengupta, F. Jin, R. Zhang and S. Cao, 'mm-Pose: Real-Time Human Skeletal Posture Estimation Using mmWave Radars and CNNs', *IEEE Sensors Journal*, vol. 20, no. 17, pp. 10 032–10 044, Sep. 2020, Publisher: Institute of Electrical and Electronics Engineers (IEEE), ISSN: 2379-9153. DOI: 10.1109/jsen.2020.2991741. [Online]. Available: http://dx.doi.org/10.1109/JSEN.2020.2991741.

[4] A. Chen, X. Wang, S. Zhu, Y. Li, J. Chen and Q. Ye, *mmBody Benchmark: 3D Body Reconstruction Dataset and Analysis for Millimeter Wave Radar*. Sep. 2022. DOI: 10.48550/arXiv.2209.05070.

[5] H. Xue, Y. Ju, C. Miao *et al.*, 'mmMesh: Towards 3D real-time dynamic human mesh construction using millimeter-wave', in *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '21, event-place: Virtual Event, Wisconsin, New York, NY, USA: Association for Computing Machinery, 2021, pp. 269–282, ISBN: 978-1-4503-8443-8. DOI: 10.1145/3458864.3467679. [Online]. Available: https://doi.org/10.1145/3458864.3467679.

[6] S. Palipana, D. Salami, L. A. Leiva and S. Sigg, 'Pantomime: Mid-Air Gesture Recognition with Sparse Millimeter-Wave Radar Point Clouds', *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 5, no. 1, Mar. 2021, Place: New York, NY, USA Publisher: Association for Computing Machinery. DOI: 10.1145/3448110. [Online]. Available: https://doi.org/10.1145/3448110.

[7] H. Regmi, M. S. Saadat, S. Sur and S. Nelakuditi, 'SquiggleMilli: Approximating SAR Imaging on Mobile Millimeter-Wave Devices', *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 5, no. 3, Sep. 2021, Place: New York, NY, USA Publisher: Association for Computing Machinery. DOI: 10.1145/3478113. [Online]. Available: https://doi-org.tudelft.idm.oclc.org/10.1145/3478113.

[8] S. An and U. Y. Ogras, 'MARS: mmWave-based Assistive Rehabilitation System for Smart Healthcare', *ACM Trans. Embed. Comput. Syst.*, vol. 20, no. 5s, Sep. 2021, Place: New York, NY, USA Publisher: Association for Computing Machinery, ISSN: 1539-9087. DOI: 10.1145/3477003. [Online]. Available: https://doi.org/10.1145/3477003.

[9] J. Zhang, R. Xi, Y. He *et al.*, 'A Survey of mmWave-Based Human Sensing: Technology, Platforms and Applications', *IEEE Communications Surveys & Tutorials*, vol. 25, no. 4, pp. 2052–2087, 2023, Publisher: Institute of Electrical and Electronics Engineers (IEEE), ISSN: 2373-745X. DOI: 10.1109/comst.2023.3298300. [Online]. Available: http://dx.doi.org/10.1109/COMST.2023.3298300.

[10] C. R. Qi, H. Su, M. Kaichun and L. J. Guibas, 'PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation', in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, ISSN: 1063-6919, Los Alamitos, CA, USA: IEEE Computer Society, Jul. 2017, pp. 77–85. DOI: 10.1109/CVPR.2017.16. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2017.16.

[11] K. O'Shea and R. Nash, 'An Introduction to Convolutional Neural Networks', *ArXiv e-prints*, Nov. 2015.

[12] B. Shah and H. Bhavsar, 'Time Complexity in Deep Learning Models', *Procedia Computer Science*, vol. 215, pp. 202–210, 2022, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2022.12.023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050922020944.

[13] H. Gholamalinezhad and H. Khosravi, 'Pooling Methods in Deep Neural Networks, a Review', *ArXiv*, vol. abs/2009.07485, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:221738915.

[14] L. Zaniolo and O. Marques, 'On the use of variable stride in convolutional neural networks', *Multimedia Tools and Applications*, vol. 79, May 2020. DOI: 10.1007/s11042-019-08385-4.

[15] S. Ioffe and C. Szegedy, 'Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift', in *Proceedings of the 32nd International Conference on Machine Learning*, F. Bach and D. Blei, Eds., ser. Proceedings of Machine Learning Research, vol. 37, Lille, France: PMLR, Jul. 2015, pp. 448–456. [Online]. Available: https://proceedings.mlr.press/v37/ioffe15.html.

[16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, 'Dropout: A Simple Way to Prevent Neural Networks from Overfitting', *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html.

[17] M. Jaderberg, K. Simonyan, A. Zisserman and k. kavukcuoglu koray, 'Spatial Transformer Networks', in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama and R. Garnett, Eds., vol. 28, Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2015/file/33ceb07bf4eeb3da587e268d663aba1a-Paper.pdf.

[18] X. Li, 'Exploring the effect of depth and width of CNN models on binary classification of dogs and cats', *Applied and Computational Engineering*, vol. 47, pp. 147–158, Mar. 2024. DOI: 10.54254/2755-2721/47/20241274.

[19] M. Zhou, T. Liu, Y. Li, D. Lin, E. Zhou and T. Zhao, 'Toward Understanding the Importance of Noise in Training Neural Networks', in *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, Eds., ser. Proceedings of Machine Learning Research, vol. 97, PMLR, Jun. 2019, pp. 7594–7602. [Online]. Available: https://proceedings.mlr.press/v97/zhou19d.html.

[20] Microsoft, *Kinect Sensor*, Publisher: Kinect for Windows, 2022. [Online]. Available: https://learn.microsoft.com/en-us/windows/apps/design/devices/kinect-for-windows.

[21] D. Kingma and J. Ba, 'Adam: A Method for Stochastic Optimization', *International Conference on Learning Representations*, Dec. 2014.

[22] Papers with Code, *Early Stopping*, 2024. [Online]. Available: https://paperswithcode.com/method/early-stopping#:~:text=Early%20Stopping%20is%20a%20regularization,improves%20on%20a%20validation%20set.

[23] Y. Zhou, H. Dong and A. El Saddik, 'Learning to Estimate 3D Human Pose From Point Cloud', *IEEE Sensors Journal*, vol. PP, pp. 1–1, Jun. 2020. DOI: 10.1109/JSEN.2020.2999849.

[24] The Netherlands Organisation for Scientific Research (NWO), *The Netherlands Code of Conduct for Research Integrity*, 2018. [Online]. Available: https://www.nwo.nl/sites/nwo/files/documents/Netherlands%20Code%20of%20Conduct%20for%20Research%20Integrity_2018_UK.pdf.

[25] C. R. Qi, L. Yi, H. Su and L. J. Guibas, 'PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space', in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/d8bf84be3800d12f74d8b05e9b89836f-Paper.pdf.

Appendix

A Accuracy of 3D Joint Position Estimation

Joint	X (Horizontal) (cm)		Y (Depth) (cm)		Z (Vertical) (cm)		Average (cm)	
	MAE	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
SpineBase	7.32	10.10	5.54	7.15	7.74	10.03	6.86	9.09
SpineMid	7.58	10.43	5.10	6.43	8.57	11.09	7.09	9.32
Neck	8.71	11.91	5.79	7.35	9.55	12.26	8.02	10.51
Head	9.05	12.35	5.65	7.74	10.19	12.97	8.30	11.02
ShoulderLeft	8.35	11.50	5.86	7.79	8.54	10.96	7.58	10.08
ElbowLeft	9.59	12.69	7.03	9.38	10.54	13.46	9.05	11.84
WristLeft	13.55	17.38	7.82	10.47	17.23	21.93	12.87	16.59
ShoulderRight	8.19	11.25	5.59	7.67	8.90	11.35	7.56	10.09
ElbowRight	9.78	12.90	6.66	9.22	9.81	12.46	8.75	11.53
WristRight	12.69	16.56	7.56	9.97	16.79	21.45	12.35	15.99
HipLeft	6.94	9.58	5.37	7.19	7.84	10.13	6.71	8.97
KneeLeft	7.72	10.58	5.92	8.17	4.57	6.26	6.07	8.33
AnkleLeft	8.33	11.16	6.23	8.58	3.74	6.02	6.10	8.59
FootLeft	8.43	11.43	6.74	9.32	4.04	6.20	6.40	8.99
HipRight	7.24	9.82	5.62	7.25	7.56	9.74	6.81	8.94
KneeRight	7.40	10.14	6.23	8.46	4.99	6.98	6.21	8.53
AnkleRight	8.50	11.31	6.42	8.55	4.68	7.05	6.53	8.97
FootRight	8.91	11.80	6.90	9.36	4.81	7.50	6.87	9.55
SpineShoulder	8.22	11.35	4.87	6.34	9.47	12.18	7.52	9.96
Average	8.76	11.80	6.15	8.23	8.40	11.05	7.77	10.36

Table 2: *Average localisation error for 19 human joints position using MARS with PointNet integration. The results in this table are for 20% test data.*