



Informatics

Monokulare 3D-Schätzung menschlicher Körperhaltungen zur Beobachtung von Fahrzeuginnenräumen unter Verwendung synthetischer Bilder

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Data Science

eingereicht von

Thummanoon Kunanuntakij, B.Eng.

Matrikelnummer 12122522

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof.in Dipl.-Ing.in Mag.a rer.nat. Dr.in techn. Margrit Gelautz

Mitwirkung: Dipl.-Ing. Dominik Schörkhuber, BSc.

Wien, 2. Februar 2026

Thummanoon Kunanuntakij

Margrit Gelautz

Technische Universität Wien

A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.



Monocular 3D Human Pose Estimation for In-cabin Monitoring Utilizing Synthetic Images

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Data Science

by

Thummanoon Kunanuntakij, B.Eng.

Registration Number 12122522

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof.in Dipl.-Ing.in Mag.a rer.nat. Dr.in techn. Margrit Gelautz

Assistance: Dipl.-Ing. Dominik Schörkhuber, BSc.

Vienna, February 2, 2026

Thummanoon Kunanuntakij

Margrit Gelautz



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Erklärung zur Verfassung der Arbeit

Thummanoon Kunanuntakij, B.Eng.

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, haben ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT- Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 2. Februar 2026

Thummanoon Kunanuntakij



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Declaration of Authorship

Thummanoon Kunanuntakij, B.Eng.

I hereby declare that I have written this thesis independently, that I have completely specified the utilized sources and resources and that I have definitely marked all parts of the work - including tables, maps and figures - which belong to other works or to the internet, literally or extracted, by referencing the source as borrowed.

I further declare that I have used generative AI tools only as an aid, and that my own intellectual and creative efforts predominate in this work. In the appendix “Overview of Generative AI Tools Used” I have listed all generative AI tools that were used in the creation of this work, and indicated where in the work they were used. If whole passages of text were used without substantial changes, I have indicated the input (prompts) I formulated and the IT application used with its product name and version number/date.

Vienna, February 2, 2026

Thummanoon Kunanuntakij



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Danksagung

Ich möchte meinen aufrichtigen Dank an meine Betreuerin, Prof. Margrit Gelautz, und meinen Ko-Betreuer, Dominik Schörkhuber, aussprechen für die Möglichkeit, an diesem interessanten Projekt zu arbeiten, sowie für ihre Anleitung während des gesamten Weges. Ich möchte auch meinen Kolleg*innen Raquel Panadero Palenzuela, Lukas Brunner und anderen Forschenden der Forschungseinheit für Computer Vision an der TU Wien für ihre Unterstützung und die fruchtbaren Diskussionen danken. Ich bin Sophie Kroiss und Katharina Pois für ihre Hilfe bei Büroangelegenheiten und der Verwaltung dankbar.

Diese Arbeit wurde durch die Projekte SyntheticCabin (Projektnummer 884336) und UNISCOPE-3D (Projektnummer 911019) unterstützt, die durch die Österreichische Forschungsförderungsgesellschaft (FFG) im Auftrag des Bundesministeriums für Klimaschutz, Umwelt, Energie, Mobilität, Innovation und Technologie (BMK) über das Förderprogramm „Mobilität der Zukunft“ beziehungsweise das FFG-Basisprogramm finanziert wurden. Die synthetischen Datensätze wurden von unserem Partner emotion3D zur Verfügung gestellt. Vielen Dank für die wunderbare Zusammenarbeit.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acknowledgements

I would like to express my sincere gratitude to my supervisor Prof. Margrit Gelautz and my co-supervisor Dominik Schörkhuber for the opportunity to work on this interesting project and for their guidance throughout the journey. I also would like to thank my colleagues Raquel Panadero Palenzuela, Lukas Brunner, and other researchers at the Research Unit of Computer Vision at TU Wien, for their assistance and fruitful discussions. I am thankful to Sophie Kroiss and Katharina Pois for their support with office matters and administration.

This thesis was supported by the projects SyntheticCabin (Project Number 884336) and UNISCOPE-3D (Project Number 911019), which were funded through the Austrian Research Promotion Agency (FFG) on behalf of the Austrian Ministry of Climate Action (BMK) via its Mobility of the Future funding program and the FFG General Program, respectively. The synthetic datasets were provided by our partner emotion3D. Thank you for your wonderful collaboration.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Kurzfassung

Einer der entscheidendsten Aspekte der Fahrzeugherstellung ist die Gewährleistung der Sicherheit der Fahrzeuginsassen. Da fahrerbedingte Faktoren wie Müdigkeit und Ablenkung zu einem Großteil der Unfälle beitragen, ist die Überwachung der Fahrer*innen wesentlich, um die Verkehrssicherheit zu verbessern. Fortschritte im Bereich des maschinellen Sehens haben den Einsatz kostengünstiger Bildsensoren zur Implementierung von Fahrerüberwachungssystemen ermöglicht. In dieser Arbeit interessierten wir uns für die Schätzung der 3D-Pose von Fahrer*innen mit dem Ziel, menschliche Skelettdarstellungen aus Eingabebildern mithilfe von Deep-Learning-Methoden zu rekonstruieren. Da Deep Learning jedoch große Datenmengen erfordert, ist die Erfassung realer Datensätze kostspielig und herausfordernd. Synthetische Daten bieten eine attraktive Alternative, die die Menge an benötigten realen Daten verringern kann, ohne die Genauigkeit zu beeinträchtigen.

Unser Ansatz folgt einem dreistufigen Framework zur 3D-Pose-Schätzung. Die Pose-Schätzungspipeline besteht aus vorgefertigten Modellen für die Personenerkennung und die 2D-Pose-Schätzung. Anschließend verwendeten wir synthetische Daten, um verschiedene 2D-zu-3D-Human-Pose-Lifting-Modelle basierend auf unterschiedlichen neuronalen Netzwerkarchitekturen für die letzte Stufe vorzutrainieren. Schließlich wurden diese Modelle mit zunehmenden Mengen realer Daten feinabgestimmt.

Ein Experiment mit Drive&Act als Benchmark-Datensatz zeigte Genauigkeitsgewinne für vortrainierte Modelle bei jeder Menge realer Daten, obwohl diese Gewinne mit zunehmender Menge realer Daten abnahmen. Hybride Modelle wie GraphMLP und GraFormer erzielten die besten Ergebnisse, wenn sie mit geringen bis mittleren Mengen realer Daten trainiert wurden, während JointFormer, ein Transformer-Modell, die anderen übertraf, wenn das vollständige reale Datenset verwendet wurde. Darüber hinaus stellten wir fest, dass das nur mit dem synthetischen Datensatz vortrainierte Lifting-Modell selbst dann eine angemessene Pose-Schätzungsleistung erreichte, wenn keine 3D-Pose-Annotationen für die Ziel-Realweltdaten verfügbar waren, beispielsweise wenn deren Erfassung zu kostspielig ist. Insgesamt deuten die Ergebnisse klar auf den Vorteil der Verwendung synthetischer Daten zur Verbesserung der Genauigkeit der 3D-Fahrer*innen-Pose-Schätzung hin, insbesondere wenn 3D-Pose-Annotationen für reale Datensätze nur eingeschränkt verfügbar sind.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

One of the most crucial aspects of vehicle manufacturing is ensuring passenger safety. As driver-related factors such as fatigue and distraction contribute to a majority of accidents, monitoring drivers becomes essential to improve road safety. Advances in computer vision have enabled the use of affordable image sensors to implement driver monitoring systems. In this work, we were interested in estimating 3D driver pose with the goal of reconstructing human skeletal representations from input images using deep learning methods. However, deep learning requires large amounts of data, and real-world dataset collection is expensive and challenging. Synthetic data offers an appealing alternative that might reduce the amount of real-world data needed while maintaining accuracy.

Our approach adopts a three-stage 3D pose estimation framework. The pose estimation pipeline consists of off-the-shelf models for both human detection and 2D pose estimation. Then, we used synthetic data to pre-train various 2D-to-3D human pose lifting models based on different neural network architectures for the last stage. Finally, we fine-tuned these models with increasing amounts of real-world data.

An experiment with Drive&Act as a benchmark dataset revealed accuracy gains for pre-trained models with any amount of real-world data, though these gains diminished as more real data became available. Hybrid models like GraphMLP and GraFormer performed best when trained on low to moderate amounts of real-world data, while JointFormer, a transformer model, outperformed others when trained with the full real-world dataset. In addition, we found that the lifter pre-trained only with the synthetic dataset still achieved reasonable pose estimation performance even when 3D pose annotations for the target real-world data were not available, such as when they are too costly to obtain. Overall, the findings clearly suggest the advantage of using synthetic data for improving the accuracy of 3D driver pose estimation, especially when 3D pose annotations for real-world datasets are limited.

Contents

Acknowledgements	xi
Abstract	xv
Contents	xvii
1 Introduction	1
1.1 Aim of the Work	2
1.2 Research Questions	3
1.3 Contribution	3
1.4 Thesis Structure	4
2 Background & Related Works	5
2.1 Deep Learning Architectures	5
2.2 Training Paradigms for Deep Learning	11
2.3 Human Pose Estimation	14
2.4 Datasets for Human Pose Estimation	17
2.5 Vehicle Driver Pose Estimation	23
3 Deep Learning Architectures for Human Pose Estimation	27
3.1 Three-stage 3D Human Pose Estimation	27
3.2 2D Pose Estimation	28
3.3 2D-to-3D Pose Lifter Models	30
4 3D Pose Estimation Pipeline with 2D-to-3D Pose Lifters	39
4.1 Objective & Approach	39
4.2 Dataset: SyntheticCabin IR	40
4.3 Experiment Setup	40
4.4 Results	46
5 Transfer Learning from Synthetic Data	51
5.1 Objective & Approach	51
5.2 Datasets	52
5.3 Experiment Setup	53

5.4	Results	59
5.5	Discussion	59
6	Leveraging Synthetic Data for Unlabeled Datasets	63
6.1	Objective & Approach	63
6.2	General Setup	64
6.3	Approach 1: Synthetic-to-real Generalization	65
6.4	Approach 2: Semi-Supervised Learning	66
6.5	Discussion	70
7	Conclusion and Future Works	73
7.1	Conclusion	73
7.2	Future Work	74
A	Experimental Results from Chapter 5	77
A.1	Detailed Test Results	77
A.2	Comparison of Models With and Without Pre-training	81
A.3	Example 3D Pose Outputs	88
B	Overview of Generative AI Tools Used	95
	List of Figures	99
	List of Tables	101
	Glossary	103
	Acronyms	105
	Bibliography	107

Introduction

In recent years, the way we drive our cars has rapidly transformed. At the beginning of the 21st century, we saw vehicles with simple parking sensors that sounded when other objects came too close. Some cars also came with rear cameras to help drive backward. However, these features only provide information to make driving easier. Newer cars offer even more advanced driving assistance features that control our vehicles directly. These features include lane control, automatic slowing down, and braking when too close to the vehicle in front. We move closer and closer to manufacturers' ultimate goal: to build fully autonomous vehicles. Having such a system will be beneficial to road users. Not only is it a convenience for passengers, it is also safer. Dingus et al. [DGL⁺16] estimated that driver-related factors such as fatigue and distraction lead to 87.7% of crashes. People sometimes use phones or messaging applications while driving. At other times, they may be fatigued and fall asleep. These behaviors could lead to fatal accidents. Self-driving vehicles have the potential to mitigate such risks. Moreover, the economy could benefit from higher efficiency. Trucks can deliver goods without rest, and taxis and buses can service passengers all night.

There is still a long way to reach the goal of fully autonomous vehicles. Currently, many car manufacturers are offering semi-autonomous systems for certain high-end automobiles. These systems, at their best, can drive the car for us most of the time, but they still require drivers to pay full attention to the road and take control when necessary, as the systems may make mistakes in unexpected situations. However, maintaining this level of vigilance is increasingly difficult due to human nature. We tend to become bored or sleepy when we need to sit still without activity, causing us to shift our attention to distractions, such as mobile phones [KSZ15, AMA⁺23].

Because human behaviors are a major factor in road accidents, a necessary step to prevent risky actions is to implement a driver monitoring systems (DMS). Car manufacturers have been constantly improving these systems to enhance passengers' safety. Initially,

cars simply checked if occupants used seat belts. Then, in 2008, they began checking for drowsiness from steering patterns, and by 2016, vehicles were equipped with in-cabin cameras facing the driver to monitor face and eyelid movements [MBB⁺21]. These systems can alert passengers [NPN⁺24], or even apply emergency brakes if they determine that a driver is unable to respond to dangerous scenarios [MKM22]. The potential for enhanced safety is so significant that the European New Car Assessment Programme (Euro NCAP) included the system as part of its safety rating in 2023, and the European Union has mandated all new vehicle models to include the DMS from 2024 in the recent General Safety Regulation [The23].

Recent advancements in computer vision have opened up vast possibilities for the DMS. One exciting task for such systems is human pose estimation (HPE), a classic computer vision challenge. It involves calculating the position in either 2D or 3D coordinates of critical key points of the human body, such as the head, arms, and hands. The results are typically visualized as stick figures with dots representing anatomical joints and lines representing bones. We can use these poses as a compact representation of the body, free from challenges associated with raw images, such as lighting conditions, varied environments, and large data sizes. With pose data, we can classify human actions [XLL⁺23, RLDL24], perform human tracking [ZML⁺23], and enable gesture recognition [DSWV16]. All of these capabilities provide valuable information for creating a sophisticated DMS.

3D human pose estimation (3D HPE) is the ultimate goal as it is less ambiguous than 2D approaches. Traditionally, we can use multiple cameras to determine the accurate position of each joint using triangulation methods. However, there are practical obstacles, as we need to carefully calibrate the cameras' parameters, and even slight perturbations in camera position or angle can lead to significant errors. Advancements in deep learning now allow us to achieve comparable results with only monocular images [ZWC⁺23]. Using a single camera could simplify installation and reduce costs. Yet, a significant barrier remains: the lack of labeled data. Recording drivers in various environments, poses, and actions can be expensive and potentially dangerous. Furthermore, annotating 3D poses is inherently challenging. We typically acquire high-quality ground-truth data using motion capture technology, which is generally confined to controlled studio environments [WTZ⁺21, LBSM23].

1.1 Aim of the Work

This thesis investigates the possibility of using synthetic datasets to alleviate the need for extensive real-world data acquisition. We aim to evaluate the effectiveness of synthetic data as a foundation for transfer learning of monocular 3D HPE models to real-world scenarios. Using synthetic data provides significant advantages. Not only do we know the exact position of every joint, even when occluded, but it is also cheaper to produce more samples with higher variation. Pre-training with synthetic data may reduce the quantity of real-world data needed, with potentially only a minor trade-off in accuracy.

Our research will clarify how effectively synthetic images can support deep learning model training, determine the minimum amount of real-world data required, and assess whether generated images remain valuable when real-world 3D annotations are unavailable.

The scope of this study focuses on a top-down multi-staged approach, employing an off-the-shelf single-person 2D pose estimator followed by a 2D-to-3D pose lifter model to perform the final 3D estimation. This approach offers a key advantage: synthetic and natural 3D poses share greater similarity than their corresponding 2D images. Consequently, models trained with synthetic data may transfer to real-world scenarios with minimal or no fine-tuning. Furthermore, the 3D skeletal representation is significantly more compact and computationally efficient. This efficiency allows us to train deep learning models more rapidly and results in smaller model sizes. Such characteristics could enable automobile manufacturers to implement these systems in resource-constrained environments typical of in-vehicle computing platforms.

This work is part of the Austrian Research Promotion Agency (FFG)-supported SyntheticCabin and UNISCOPE-3D projects, which are collaborative initiatives between TU Wien and industrial partners. The projects aim to develop a cost-effective and time-efficient in-cabin simulation framework capable of generating synthetic data for training, validating, and testing machine learning models for 2D and 3D driver monitoring.

1.2 Research Questions

This thesis aims to determine whether monocular synthetic images can effectively reduce the dependence on real-world data for training machine learning models in 3D HPE by addressing the following research questions:

In the context of vehicle driver monitoring:

- RQ 1:** To what extent can we estimate 3D pose from monocular synthetic data?
- RQ 2:** To what extent can knowledge of 3D HPE be transferred from synthetic to real-world data?
- RQ 3:** To what extent can 3D poses be estimated from real-world data without 3D pose annotations?

1.3 Contribution

We can summarize the contributions of this thesis as the following:

- We compared the accuracy of five different deep learning networks for 2D-to-3D pose lifters. All of the models take a single 2D pose as an input.
- We evaluated the usefulness of synthetic data for pre-training 3D pose estimators.

- We studied the usefulness of synthetic data generated by the SyntheticCabin project for 3D HPE for the case where we can finetune the model on real-world data and when we cannot.

1.4 Thesis Structure

This thesis discusses the background and related works in Chapter 2. This chapter distinguishes different approaches and methodologies in pose estimation research. We explain our setup for the pose estimation pipeline and review deep learning architectures used across all experiments in Chapter 3. The subsequent three chapters detail the experimental setup, results, and discussion for each investigation. In Chapter 4, we begin by applying monocular 2D-to-3D pose lifters to the SyntheticCabin IR dataset to evaluate the performance of monocular approaches compared to triangulation methods. In Chapter 5, we assess the benefits of pre-training lifter models with synthetic data to characterize the accuracy trade-offs when reducing reliance on real-world data. Chapter 6 explores the feasibility of applying the synthetic data to real-world datasets whose 3D poses have not been annotated. Finally, in Chapter 7, we summarize our key findings and discuss directions for future research.

Background & Related Works

In this chapter, we discuss the necessary background and provide an overview of related work in HPE. In Section 2.1, we present an overview of important deep learning architectures that form the foundation of the 2D-to-3D pose lifters used in our experiments. Next, we discuss a couple of training paradigms that help guide our experiments in Section 2.2. Then, we introduce the task of HPE in Section 2.3. The following section outlines the approaches commonly used for addressing this task and the metrics used for benchmarking their accuracy. In Section 2.4, we review common benchmark datasets for HPE, including both real and synthetic images. Finally, we summarize related works in driver pose estimation in Section 2.5.

2.1 Deep Learning Architectures

Deep Learning has revolutionized the world in the past decade, particularly in the fields of natural language processing (NLP) and computer vision. Several factors contributed to this boom, such as faster and cheaper computer hardware, affordable memory storage, widespread usage of the internet, and algorithmic advancements. This section briefly reviews important deep learning concepts and architectures which are particularly important to breakthroughs in computer vision.

2.1.1 Perceptron

Deep learning is based on a mathematical structure called the artificial neural network. Originally, it was inspired by how information signals are processed by neuron cells in animal brains. Brain cells receive and transmit information at a gap between two

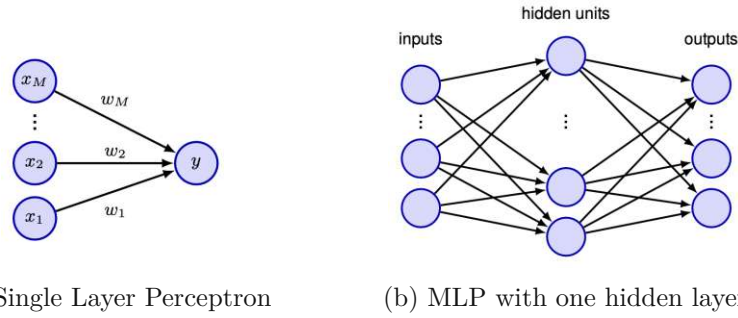


Figure 2.1: Perceptron Models. Figure obtained from [BB24].

neurons called a synapse when the electric potential exceeds a certain threshold [BB24]. However, the artificial neural network only superficially resembles the biological one [Pri23]. Practical implementations of neural networks and deep learning diverge far from how the biological ones behave.

In 1962 Rosenblatt [Ros63] pioneered a learning algorithm called the Perceptron. The Perceptron, depicted in Figure 2.1a, is a non-linear function that can be expressed as,

$$a = \sum_i^M w_i x_i \tag{2.1}$$

$$y = f(a), \tag{2.2}$$

where M is the dimension of the input, x_i represents the value of the input, w_i represents model weight, and $f(\cdot)$ is an activation function. Originally, $f(\cdot)$ was a step function with $f(x) = 1$ when $x > 0$ and $f(x) = 0$ otherwise. Note that there are many other activation functions used in modern architectures, notably rectified linear unit (ReLU) [GBB11], Sigmoid function, and hyperbolic tangent function [LBOM12], which allow more flexible output.

2.1.2 Multi-layer Perceptron

The single-layer Perceptron model was criticized by Minsky et al. [MP69] for its limitation in being unable to learn a simple XOR (logical exclusive-OR) operation. This led to a loss of popularity for the algorithm during the 1970s. However, this criticism applied only to the single-layer model. Rosenblatt [Ros63] had already addressed this limitation in the original text and discussed how to mitigate it by stacking additional "hidden" layers, resulting in what is generally called the multi-layer perceptron (MLP), depicted in Figure 2.1b. Hornik et al. [HSW89] later showed that by adding just one hidden layer with sufficient width and appropriate activation functions, the resulting model can approximate any continuous functions on compact subsets of R^n to arbitrary precision. This result is known as the universal approximation theorem.

Neural network models learn through the back-propagation method [RHW86]. This method computes weight gradients backward layer by layer from the output using partial derivatives of error terms with respect to each weight. In practice, training a neural network model utilizes an optimization algorithm called gradient descent, along with dynamic programming to compute the gradients efficiently.

2.1.3 Deep Learning

As its name suggests, Deep Learning is a neural network with several hidden layers. One notable characteristic of deep learning models is that they are built by stacking multiple simpler modules [LBH15]. More layers help the model to learn semantically meaningful representations of input data. These new features enable the model to perform complex tasks more easily [BB24].

AlexNet, a convolutional neural network variant proposed by Krizhevsky et al. [KSH12], was one of the most influential works that led to a boom in deep learning. It produced a significant improvement in image classification tasks on the ImageNet dataset [DDS⁺09] by reducing the error rate by approximately 8%. Since then, there has been rapid improvement in techniques and creation of new architectures that deliver state-of-the-art results for several tasks in computer science.

2.1.4 Convolutional Neural Network

Convolutional neural network (CNN) is an architecture designed to take advantage of the inherent properties of image data, as shown in Figure 2.2. The network's core feature is its use of convolution kernels, linear filters that learn to detect patterns by sliding across images. Stacking multiple convolutional layers enables later kernels to operate on larger receptive fields, making the model incorporate wider and more complex visual features. The architecture periodically employs max-pooling layers, which retain maximum values to strengthen key features and ensure translational invariance. This combination of convolution and pooling operations allows CNNs to extract more meaningful image features than traditional MLP models.

Throughout the years, several influential architectures have originated and contributed important ideas to CNN development. In 1998, LeNet-5 [LBBH98] was one of the first models to inspire the use of CNNs for image classification tasks, specifically handwritten digit recognition. It comprised convolution and average pooling layers for feature extraction, with fully connected layers for classification.

However, it was AlexNet [KSH12] in 2012 that demonstrated the effectiveness of CNNs in computer vision by achieving accuracy that surpassed conventional methods in the ImageNet dataset. Several techniques were responsible for this breakthrough: the use of max-pooling for better signal retention, ReLU activation functions [GBB11] for improved gradient propagation in deeper networks, and regularization methods including data augmentation, dropout, and Local Response Normalization. Advances in computation also enabled deeper models to be trained on graphics processing units rather than CPUs.

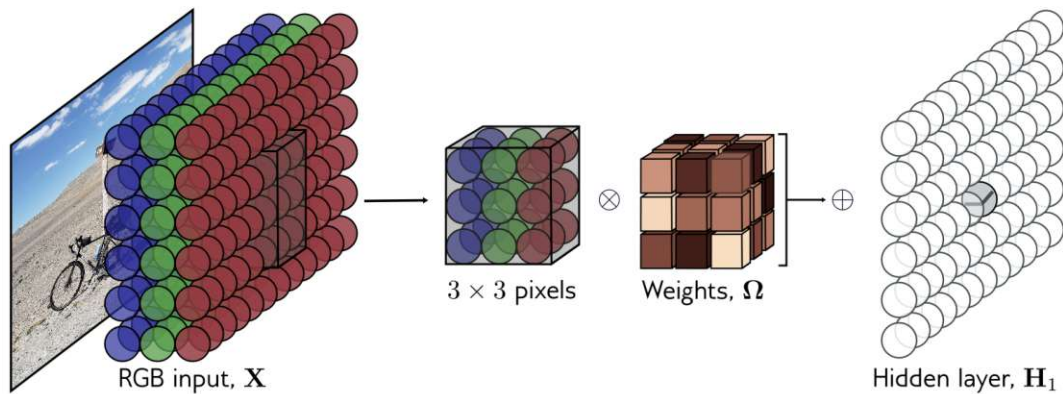


Figure 2.2: Convolution Mechanism of Convolutional Neural Network. Figure obtained from [Pri23].

Since AlexNet, several improvements have been proposed for CNN models. ResNet [HZRS16] introduced skip connections that addressed vanishing and exploding gradient problems while smoothing the loss landscape for better convergence [LXT⁺18], enabling deeper networks. VGG [LD15], Inception [SVI⁺16], and DenseNet [HLVDMW17] focused on increasing the depth of the network while improving the learning processes and computational efficiency.

Due to its success in computer vision, CNNs have been applied to other domains as well. They have been used in audio signal processing through spectrograms, visual representations of audio based on short-time Fourier transformation [AHMJ⁺14]. WaveNet [ODZ⁺16] applied CNN to text-to-speech generation, while Gehring et al. [GAG⁺17] utilized it for machine translation and Borovykh et al. [BBO17] for time-series forecasting.

2.1.5 Graph Neural Network

A graph is a mathematical structure consisting of nodes and edges. The nodes represent elements, while the edges describe relationships between pairs of nodes. This versatile data representation can model various scenarios, including grammatical structures, road networks, social relationships, chemical molecules, point cloud images, and electrical circuits [Pri23]. In computer vision, graphs can represent scene structures, anatomical relationships in humans and animals, and even pixels in images. The ability to apply deep learning models to analyze and make predictions on graph data benefits numerous domains.

The graph neural network (GNN) [SGT⁺09, GSR⁺17] is a deep learning architecture that operates on graph data through iterative message passing. Initially, each node and edge can be attached with information (e.g., user profiles in a social network and types of relationship). During each iteration, the nodes transmit information to their neighbors,

and then they weight and aggregate incoming messages with their own data to create new node representations. Repeated iterations increase the receptive field and the depth of the network. Not only on nodes, a similar idea can be applied on edges or to aggregate the information into a representation of a graph as a whole.

As an example, we can summarize the concept of message-passing graph convolutional network (GCN) [KW16] for learning node representation in simple mathematical notation as follows:

$$\Phi_{k+1}(v) = \text{Update}_k(\Phi_k(v), \text{Aggregate}(\Phi_k(w) | w \in \mathcal{N}(v))), \quad (2.3)$$

where $\Phi_k(x)$ is the embedding of node x in layer k . $\mathcal{N}(v)$ is a set of neighboring nodes of v . $\text{Aggregate}(\cdot)$ is a function that aggregates information from the input set of the node. Examples of this function are weighted average and maximum pooling functions. $\text{Update}(\cdot)$ is a function which fuses aggregated information of the neighbor with the current node.

The development of GCN led to diverse architectural variants. The GCN is one of the simplest message-passing neural networks. It featured the use of learnable shared weights applied on each node before aggregation. Graph Attention Networks (GAT) [VCC⁺17] built on GCN by introducing an attention mechanism that allows the model to specify different weights to different neighbors. GraphSAGE (SAmple and aggreGatE) [HYL17] scaled the GNN to a large graph by, instead of using all neighbor nodes, sampling only a subset of them. This also allowed GraphSAGE to handle unseen nodes better.

2.1.6 Transformer

The transformer model [VSP⁺17] was first introduced as a sequence-to-sequence language translation model in NLP. Its performance surpassed the recurrent neural network (RNN) [RHW86] in both output quality and computation time. While RNN processes each token of a sequence one by one and builds up the representation state of the sequence, the transformer takes in the whole sequence at the same time and uses a self-attention mechanism to find relations between each token. This parallel processing capability significantly improves training efficiency.

Originally, the transformer model consisted of two parts, called an encoder and a decoder, depicted in Figure 2.3. The encoder is responsible for learning the representation of the input while the decoder processes the past output and performs inference to produce the output. However, the encoder and decoder are already useful on their own. NLP researchers trained one of them to create foundation language models known as large language models [ZZL⁺24]. The decoder alone has been widely and successfully adapted by researchers in other fields, such as computer vision, recommendation systems, and audio processing, for learning feature representation as it is simple and versatile.

The transformer model takes a sequence of tokens as input. In NLP, a token usually represents a word in a sentence. In the first step, each token is embedded as a numerical

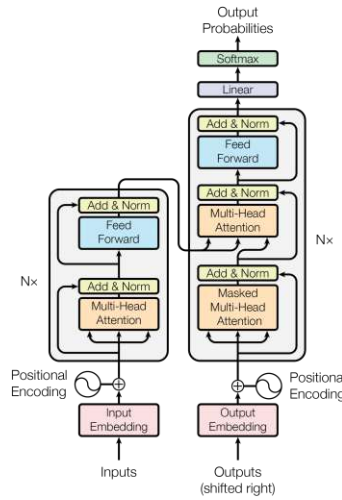


Figure 2.3: Architecture of the transformer model. Figure obtained from [VSP⁺17]

vector, usually with a simple linear layer. Then, information about the relative position between each token is added by positional encoding. The core component of the transformer model is the attention mechanism, which allows the model to weigh the importance of different input elements dynamically. The attention mechanism [BCB15] is mathematically expressed as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \quad (2.4)$$

where Q , K , and V represent the query, key, and value vectors of dimension d_k , respectively. Each vector is a token in an input sequence that is scaled by learnable weight matrices for query (W_Q), key (W_K), and value (W_V). Conceptually, the attention function calculates the similarity between the query, Q , and the key, K , and uses the result to scale the value vectors, V , for the output. The result is effectively a weighted vector of the input that strengthens values of useful tokens and dampens the irrelevant ones.

The transformer implements multi-head attention, where the attention mechanism is performed multiple times (eight in the original model) in parallel. Each attention head can learn to focus on different aspects of the input, producing a diverse set of representations that are then concatenated together. This multi-head approach allows the model to capture various types of relationships in the data simultaneously.

The transformer architecture has led to state-of-the-art models across various fields. In computer vision, the Vision Transformer [DBK⁺21] adapts the transformer encoder for image classification by processing 16×16 image patches as input tokens. In audio processing, Music Transformer [HVU⁺18] processes sequences of Musical Instrument

Digital Interface (MIDI) data. In biology, AlphaFold [JEP⁺21] modified the transformer architecture for protein structure prediction.

2.1.7 Generative Adversarial Network

A generative model learns the data distribution of the training set and produces new instances that mimic the set [BB24]. Practical applications of such models include generating new images and music. Deep learning models are powerful enough to learn such distributions. However, training a generative model is not trivial.

generative adversarial network (GAN) [GPAM⁺14] is an approach to train a generative model by having two deep learning models compete against each other. The first model is the generator network that takes a latent vector as input. This input introduces randomness to the model, which should make the generator produce diverse results. The second model is called the discriminator network. The discriminator learns to distinguish generated images from real images. During training, each model takes turns being trained. When training the generator model, we let it generate a new image, then use the discriminator to decide whether it believes this is a fake image or not. The generator learns directly from the result of this classification. This makes the discriminator act as a loss function for the generator. Then, during the discriminator's turn, we feed it with either real images from the training dataset or fake images from the generator. The discriminator learns through binary classification whether it can correctly classify images.

Even using GAN, training a generative model remains non-trivial. We need to maintain balance throughout the process to arrive at a good generator. First, both models should be equally capable. If one model is more powerful than the other, it will tip the balance, resulting in one model constantly winning over the other and halting training progress. Another common problem for GAN is called mode collapse, which occurs when a generator finds a small set of solutions that the discriminator fails to classify and decides to stick with them. Thus, it constantly produces the same set of samples and lacks variety.

GAN has been a popular approach in image generation. Initially, it was used to generate random images that mimic the input dataset. Researchers then created Conditional GAN [MO14] that also takes an image label as input and produces only images of the desired class. Fast Style Transfer [JAFF16] used GAN to transform an input image to look like another image. For example, it can transform a photo to look like an oil painting. Beyond image generation, GAN is also used to generate other synthetic data such as time series [YJvdS19] and human motion [XSW⁺23].

2.2 Training Paradigms for Deep Learning

Due to the large number of parameters, training deep learning models requires a substantial amount of data. However, obtaining a large quantity of labeled data is often challenging, as data annotation is typically a labor-intensive and time-consuming process. Researchers and practitioners have explored various strategies to address this limitation.

In this section, we review several learning paradigms applicable when labeled data is scarce. These approaches are particularly relevant to tasks such as 3D pose estimation, where acquiring accurate annotations is both challenging and expensive. Table 2.1 summarizes the characteristics of each paradigm.

2.2.1 Supervised Learning

In supervised learning, a model is trained by feeding it a training set of input–output pairs. A learning algorithm adjusts the model’s parameters until its predictions are as close as possible to the true outputs. If the training is successful, the model generalizes well and produces accurate predictions on new, unseen inputs from a separate testing set [Pri23]. The power of deep learning was initially demonstrated through this paradigm, solving complex computer science problems on benchmark datasets [BB24], including tasks in computer vision [LBBH98, KSH12] and natural language processing [HS97]. As deep learning research advanced, deeper architectures such as VGG [LD15] achieved higher performance than shallower models. However, increasing the number of parameters in supervised learning necessitated larger training sets to fully train the models. Each computational task also required specifically designed labels. Moreover, even when such datasets were available, larger models demanded substantially more computational power. Because of these challenges, alternative training paradigms were developed to address these limitations. The subsequent subsections introduce some of these paradigms in more detail.

2.2.2 Transfer Learning

Because there are countless tasks we may want deep learning models to perform, it is not feasible to obtain sufficient data for each task individually. However, the representations learned by a model on one task often capture features useful for related tasks [BB24]. To address this, models trained on tasks with abundant data—often referred to as pre-trained models—can be adapted to tasks with scarce data. This process can be viewed as initializing training with well-learned parameters rather than random ones [Pri23]. The result is a reduction in the amount of task-specific data required and a shorter fine-tuning time. In deep learning, transfer learning is usually implemented by updating only part of the architecture, most commonly the final layer of the network [BB24]. This paradigm has shown success across multiple domains. For instance, in medical imaging, Esteva et al. [EKN⁺17] employed the Inception model [SVI⁺16], originally trained on the ImageNet dataset [DDS⁺09], to classify skin cancer. Similarly, in machine translation, Zoph et al. [ZYMK16] trained a Long Short-Term Memory (LSTM) model [HS97] on high-resource languages such as English and French and successfully transferred it to low-resource languages such as Turkish and Uzbek.

2.2.3 Semi-supervised Learning

In situations where a large quantity of data is available but only a small portion is labeled, we may wish to utilize the unlabeled data, although it may not be feasible to label more samples manually. For example, we may have thousands of hours of audio recordings or surveillance videos but can label events in only a few tens of hours due to the associated cost. In such cases, the task can be approached in a semi-supervised manner. One common strategy is to train a model on the labeled portion of the dataset and subsequently perform inference on the unlabeled data to obtain pseudo-labels [Pri23], which are then used to further train the model. Alternatively, graph-based or clustering methods can be employed to link related examples. In this setting, the labels of unlabeled samples can be inferred from their connections to labeled ones. Ge et al. [GGJY20] employed a graph-based semi-supervised method for brain tumor classification. They first trained a CNN on labeled data and then extracted features from both labeled and unlabeled samples. Next, they constructed a graph connecting similar samples based on these features. Their key innovation was introducing a 3D–2D consistency constraint to ensure that all 2D slices from the same patient’s 3D brain scan received the same tumor classification.

2.2.4 Weakly-supervised learning

In a different scenario, we may have labeled data, but the labels are not sufficiently detailed or are only indirectly related to the target task. For example, we may want to detect objects and their locations in images. The ideal labels would be bounding box coordinates around each object. However, we may only have image-level annotations indicating whether an image contains a car. Such problems can be addressed through weakly supervised learning.

One strategy is to train a classification model on these coarse labels and then use attention visualization techniques, such as Grad-CAM [SCD⁺17], to generate heatmaps highlighting the regions most relevant to the model’s predictions. These heatmaps can subsequently be processed to produce pseudo-bounding boxes, providing finer-grained supervision for training an object detection model.

An alternative approach leverages contrastive learning methods such as CLIP [RKH⁺21], which learn from images and their associated captions. The captions typically provide incomplete descriptions, omitting certain visual details or emphasizing only specific aspects of the image. CLIP employs contrastive learning by treating each image–caption pair within a batch as a positive example while considering mismatched pairs as negatives. The model is then optimized to make matching image–caption pairs more similar in embedding space while pushing non-matching pairs further apart.

Paradigms	Characteristics
Supervised Learning	Use large quantity of labeled data.
Transfer Learning	Use pre-trained models and adapt them with small quantity of labeled data.
Semi-supervised Learning	Use small quantity of labeled data to help label a large quantity of unlabeled data.
Weakly-supervised Learning	Use imperfect supervision (noisy, incomplete, indirect, or coarse-grained labels).

Table 2.1: Summary of Training Paradigms.

2.3 Human Pose Estimation

Human pose estimation (HPE) is a computer vision task that involves detecting and localizing the key body joints of people from visual input that allows reconstruction of postures and body configuration. The resulting postures are in the form of a certain representation such as skeletal key points or body mesh. Good estimation enables various applications including human movement tracking and action recognition.

2.3.1 Approaches

Throughout history, researchers have attempted to improve the accuracy of HPE using several different methods. There are a handful of common practices that researchers have employed to approach the problem. This section briefly reviews general approaches and discusses the trade-offs that need to be considered when choosing an approach to implement a solution based on deep learning HPE.

2D and 3D Pose Estimation

In general, we can divide HPE into two groups: 2D human pose estimation (2D HPE) and 3D HPE. For 2D HPE, the goal is to localize the pose in an image using the image space (pixel coordinates). The input is usually captured with a single RGB camera. 2D HPE is relatively simpler compared to 3D HPE. The output can be either the exact coordinates of the human joints or the heatmaps of each joint. The annotation for 2D poses can be done accurately by humans using the image alone. 2D HPE, especially for the case of a single person, can be done particularly well with recent models [CTH20, ZWC⁺23]. However, 2D poses are ambiguous. The same 2D human poses might correspond to different human poses, especially when some parts of the body are occluded.

In contrast, the output of 3D HPE can provide complete information. The 3D output can map to an unambiguous human pose in a world coordinate system with a defined origin. Thus, it is the ultimate goal of HPE. However, data annotation is much more difficult. The ground truth annotation can only be done using motion capture (MoCap) technology. In situations where MoCap is not possible, the 3D position can be estimated

using the triangulation of images or point clouds from different views. However, the sensors for such images and point clouds need to be well calibrated for reliable results [WTZ⁺21, LBSM23].

Single Image, Multi-view Images and Video Sequences

We can feed different numbers of images into HPE models. More images allow a model to utilize more information and produce more accurate results. However, the drawbacks are the drop in speed, the higher cost of the equipment, and the complexity of the setup. Feeding in a single image is the simplest approach, but it contains incomplete information about the human pose. People in the scene might move parts of their body to occlude other body parts. Also, we cannot determine the depth of each part. A common solution is to add more images from different views. When a body part is occluded in one view, it may be visible in others. In addition, we can calculate the depth through triangulation if the cameras are well calibrated. Another common practice is to utilize images from a video sequence. A pose that is obscured in one frame may be clearer when taking into account the movement and other frames in the same sequence [WTZ⁺21, LBSM23, XLL⁺23].

Model-based and Model-free

When the goal is to estimate the pose of the human, it can be beneficial to incorporate our prior knowledge of the human body into the estimator. A model-based approach [CTH20] uses a human body model and lets an estimator estimate the parameters of such a model. There are different common body models such as SMPL [LMR⁺15], and MANO [RTB17]. The usage of the body model restricts estimators to output only realistic poses. The drawback is the lesser flexibility.

On the other hand, a model-free approach trains a machine learning model to directly output human poses usually in the form of a skeletal figure or body mesh. This approach is more flexible, but it also requires much more training data, and it may produce unrealistic poses in unseen cases. There are some hybrid approaches. We can introduce prior knowledge to model-free estimators, for example, by forcing symmetric lengths in the left and right sides of the body using an additional loss function [ZPT⁺19]. Alternatively, we can build a pipeline which uses a model-free estimator to fit an intermediate pose and use a human model to finalize the output. Metapose [UTSS22] is an example of the latter.

Top-down and Bottom-up

In the real world, it is more common for a scene to contain multiple people. The formulation of HPE for multiple people is more complex as the number, size, and pose of people in the scene can vary. Part of a person can partially cover another person and cause confusion for the model when it needs to associate the body parts to each person.

There are two common approaches for multi-person HPE: top-down and bottom-up. For top-down, a model starts by detecting each person in the image and draws a bounding box around them, then it essentially performs single-person HPE only over the region inside the box. While the bottom-up approach locates each body part independently first, then it groups them together. The top-down approach is generally simpler and more accurate but it could be inefficient when there are many people in the scene [CTH20]. Openpose [CHS⁺19] is an example of bottom-up 2D HPE model, while Stacked Hourglass Networks [NYD16] is a top-down 2D HPE model.

Multi-staged and End-to-end

The multi-stage approach breaks down HPE tasks into a sequence of simpler sub-tasks and employs a different architecture for each. By using this approach, we decouple the process of locating 2D keypoints from estimating their 3D counterparts. This allows us to measure the accuracy of each stage separately and focus improvements on the bottleneck stage. We can also leverage pre-trained state-of-the-art models that are highly optimized and specialized for their respective stages. At present, 2D HPE is highly accurate, especially for images containing a single person, whereas 3D HPE remains comparatively less reliable [WTZ⁺21, ZWC⁺23]. Thus, researchers can focus their efforts on improving the depth estimation from 2D poses directly [CTH20].

An end-to-end approach attempts to train a model that makes the final prediction, 2D or 3D poses, from the input images directly. In deep learning, it means that the gradient values, which are used by back-propagation, can flow back from the prediction head to the part of the network responsible for learning the representation of the image.

The advantage of the end-to-end approach is that the model can be trained more easily, as it constitutes a single unified network. It also enables low-latency estimation, especially in the case of multi-person HPE, as demonstrated for pedestrian 3D HPE in [WHG23]. However, multi-stage approaches offer greater flexibility by allowing the use of 2D HPE datasets and the incorporation of intermediate processing steps [CTH20].

2.3.2 Evaluation Metrics

This section lists common evaluation metrics used in 3D HPE for poses represented with a skeletal figure.

Mean Per-Joint Position Error

mean per-joint position error (MPJPE) is the most widely used evaluation metric for 3D HPE. It is an average Euclidean distance between the annotated keypoints and the predicted keypoints. Both sets of keypoints need to be centered at the root before the calculation. MPJPE is often referred to as Protocol-I [MHRL17]. If we perform 3D HPE with K keypoints, we can calculate the metric with

$$\text{MPJPE} = \frac{1}{K} \sum_{i=1}^K \|x_i - x_i^*\|_2, \quad (2.5)$$

where x_i refers to the position of each annotated key point and x_i^* denotes the estimated positions. Both are represented in a 3D vector. $\|\cdot\|_2$ is the L^2 norm.

Per-Joint Position Error

The per-joint position error (PJPE) is the Euclidean distance between the annotated position and predicted position of a single body joint. This metric enables detailed evaluation of predictions to identify which body parts are more accurately estimated and which present greater challenges.

$$\text{PJPE} = \|x_i - x_i^*\|_2 \quad (2.6)$$

Procrustes Aligned Mean Per-Joint Position Error

Procrustes Aligned mean per-joint position error (PA-MPJPE) is another commonly used metric. For this metric, we perform the Procrustes alignment of the predicted pose to the annotated pose first. The alignment removes the scale and translational difference between two poses. Then, we calculate MPJPE. Effectively, the PA-MPJPE represents how well a 3D pose estimator predicts a shape of the skeleton without taking into account the difference in position and angle. PA-MPJPE is often referred to as Protocol-II [MHRL17] in literature.

The Procrustes alignment is done by solving a minimization problem [TGA19],

$$\min_{s, \mathbf{R}, \mathbf{t}} \left[\sum_{i=1}^K \|x_i - (s\mathbf{R}x_i^* + \mathbf{t})\|_2^2 \right], \quad (2.7)$$

where s is a scaling factor, R is a rotation matrix and t is a translation vector.

2.4 Datasets for Human Pose Estimation

We know that machine learning algorithms, especially deep learning ones, require large amounts of data to learn. Datasets play a crucial role in the advancement of the field, as they not only allow researchers to use them for training new models but also serve as benchmarks to measure progress and compare performance of different models and training approaches. This section reviews some commonly used datasets for HPE as well as datasets related to driver pose estimation. We can generally group them into 2D HPE and 3D HPE pose datasets according to the availability of the annotation.

2.4.1 2D Pose Estimation Datasets

2D HPE datasets usually contains images from real-world scenario gathered from the internet. The annotation of the image is relatively easy and can be done by crowd sourcing platform such as Amazon Mechanical Turk. Although there are multiple datasets for 2D HPE, the differences in annotation conventions such as the number of annotated joints and the location of joints, especially on the face and neck, might hinder researchers from combining and using them together for model training without lowering the number of joints. Example images of the datasets are shown in Figure 2.4.

Leeds Sports Pose

Leeds Sports Pose (LSP) [JE10] is a collection of 2,000 single-person images of athletes across eight sports gathered from Flickr, an online image hosting website. The dataset focuses on challenging poses found in real life. The pose in each image is annotated with 14 keypoints representing the whole body in image coordinates. The dataset was later extended to 10,000 images in [JE11], incorporating even more challenging poses from sports like gymnastics and parkour.

MPII Human Pose

MPII Human Pose [APGS14] is a dataset containing 28,821 and 11,701 images for training and testing respectively. These images were queried from YouTube, a video hosting website. The images feature single humans performing more than 800 activities against backgrounds from both indoor and outdoor environments in various lighting conditions. The poses in the dataset are annotated with 16 keypoints.

Common Objects in Context

Common Objects in Context (COCO) [LMB⁺15] was originally created as an object detection dataset containing 328,000 images across 80 categories of objects. From 2016 till 2020, the dataset was used in various computer vision workshops and additional annotations, including 2D human pose keypoints, were added. By 2020, the keypoint dataset contained 200,000 labeled images with 250,000 pose annotations. The poses were annotated with 17 joints and were also categorized into visible and occluded keypoints. Some occluded keypoints were also annotated if they were deducible. Unlike previous datasets, images in the COCO dataset may contain multiple people who can vary in size. Like LSP, the images in COCO were also sourced from Flickr. The curators aimed to gather non-iconic images which, in simple terms, referred to images that have no particular main object (i.e., not a portrait image of people or objects). Consequently, the images appear more natural and contain more context.

Driver Pose Estimation

Driver Pose Estimation (DriPe) [GCJT21] is a 2D HPE dataset capturing real-life drivers in consumer road vehicles. The dataset contains 10k images captured from 19 drivers across over 100 hours of video clips. Unlike other datasets, DriPe focuses specifically on driver monitoring. The videos were captured by installing an RGB camera at the top of the passenger's door pointing toward the driver while they operated the car through closed tracks or actual roads. The dataset was annotated in the same manner as the COCO dataset.



Leed Sports Pose



MPII Human Pose



COCO



DriPe

Figure 2.4: Example images from 2D HPE datasets.

2.4.2 3D Pose Estimation Datasets

Unlike 2D HPE, the annotations of 3D HPE datasets are much more demanding. Ground truth is usually obtainable only in controlled environments such as MoCap studios. Datasets that capture people in open environments might need to be annotated with some error due to estimation, typically via multi-view images and triangulation. The annotation conventions in 3D HPE also vary from one dataset to another, which makes

it difficult to combine them together. Example images of the dataset are shown in Figure 2.5.

HumanEva

HumanEva [SBB10] is one of the earliest benchmark 3D HPE datasets. The dataset captured four actors performing five actions using three RGB and four MoCap cameras. There are two versions of the dataset called HumanEva-I and HumanEva-II. Both contain the same training set composed of 6,800 frames of synchronized RGB images and motion-captured ground truth poses, with an additional 37,000 poses from only the MoCap which can be used for learning pose distribution. Both versions also use the same 6,800 frames for validation. For HumanEva-I, the testing set contained 24,000 frames, while HumanEva-II included 2,460 frames for testing. The latter test set was captured with better equipment, and the actors performed an extended sequence of actions.

Human3.6m

Human3.6m [IPOS14] is a dataset containing 3.6 million human poses of 11 professional actors captured from four different angles with a MoCap system. The acting covers 17 general real-life actions such as using a phone, greeting, and taking photos. Each pose is annotated with 32 joints.

CMU Panoptic Studio Dataset

CMU Panoptic Studio Dataset [JLT⁺15] is a multi-person 3D HPE dataset. The images in this dataset were captured in a MoCap environment featuring 480 VGA cameras, 31 HD cameras, and 10 RGB-D sensors (Microsoft Kinect). The result was highly accurate annotated 3D poses. The dataset focuses on capturing 3D poses and motions of social interactions such as playing social games. In terms of size, the dataset contains 1.5 million frames from 65 videos, totaling 5.5 hours.

3D Poses in the Wild

3D Poses in the Wild (3DPW) [vMHB⁺18] is one of the first datasets with 3D poses for in-the-wild images. The dataset utilizes 6-17 Inertial Measurement Units (IMUs) and a phone camera. During recording, one or two actors were equipped with the IMUs, while the camera was used for 2D HPE. Both inputs were combined to recover the 3D poses using a model called Video Inertial Poser (VIP). The resulting dataset contains 51,000 frames from 7 actors with 18 different clothing styles.

Drive&Act

Drive&Act [MRH⁺19] is a fine-grained driver behavior monitoring dataset. For this dataset, 15 subjects were asked to perform 34 fine-grained activities while sitting inside a static vehicle with a simulated outside environment. The dataset focuses on activities in



Human3.6m



3D Poses in the Wild



CMU Panoptic



MPII



Drive&Act

Figure 2.5: Example images from 3D HPE datasets.

the context of autonomous driving, such as working on a laptop while the vehicle is still moving. The scenes were captured with multi-modal sensors including RGB, infrared, and depth cameras placed at different angles, which resulted in over 9.6 million frames. For 3D pose annotations, the researchers used Openpose [CHS⁺19], a 2D pose estimator model, to detect 2D coordinates of 13 body joints on associated images from three frontal views. They then used triangulation to calculate the 3D coordinates.

2.4.3 Synthetic Data for Pose Estimation

As we have seen, 3D HPE datasets are difficult to create - we either need to measure 3D coordinates in a MoCap studio or estimate them through other methods while accepting some margin of error. An alternative approach is to use synthetic data, which consists of computer-generated images created using graphics programs. With these images, we can obtain precise 3D coordinates of all body joints. This approach offers several advantages: there are no privacy concerns when releasing the dataset to other users, the cost of generating additional data is lower, and the generated images can represent any scenario or environmental condition. However, this method also has several drawbacks: the image textures still differ from real images, there could be inherent biases in the dataset due to the assumptions made by its creators, and the initial cost of developing the generation



Figure 2.6: Example images from synthetic HPE datasets.

system can be high. This section has reviewed some synthetic datasets related to 3D HPE. Example images are shown in Figure 2.6.

Synthetic Humans for Real Tasks

Synthetic Humans for Real Tasks (SURREAL) [VRM⁺17] is a synthetic dataset containing 6.5 million images. The dataset was built by fitting the SMPL human model [LMR⁺15] to MoCap data from CMU MoCap [Hod]. The authors used CAESAR¹, a human whole-body scan dataset, for generating different body shapes. The background images were sourced from the LSUN dataset ². Finally, they generated synthetic images using Blender³, a 3D rendering software, by sampling different poses, lighting conditions, backgrounds, camera angles, and positions.

Synthetic Vehicle Interior Rear Seat Occupancy

Synthetic Vehicle Interior Rear seat Occupancy (SVIRO) [DCWB⁺20] is a synthetic dataset for vehicle passengers in the rear seat. Each scene could contain either an adult or a child in a child safety seat. The dataset was built by combining various realistic assets to produce synthetic sceneries in the passenger compartment. The human models were generated in MakeHuman⁴, a 3D human generator software, with poses interpolated from a fixed pre-determined set. The car seat models were modeled from real products using depth cameras and structured light scanners, while car models, textures, and environments were sourced from high quality providers. The result was a total of 25,000 sceneries. The aim of the dataset was to provide a quality synthetic dataset that could be used for benchmarking machine learning models in new real-world situations with minimized bias induced from context in the backgrounds.

¹<https://humanshape.org/CAESAR/>

²<https://complexity.cecs.ucf.edu/lsun/>

³<https://www.blender.org/>

⁴<https://static.makehumancommunity.org/>

Human in Vehicles

Human in Vehicles (HIVE) [KYH⁺24] is a synthetic dataset produced by randomly selecting human models from the AGORA dataset [PHT⁺21] and putting them into vehicle backgrounds from the SVIRO dataset, Section 2.4.3. The unique aspect of HIVE is that the annotation contains not only 2D and 3D keypoints but also body silhouettes and parameters of SMPL models [LMR⁺15]. These annotations enable the training for both 3D HPE and 3D mesh estimation.

2.5 Vehicle Driver Pose Estimation

The task of vehicle driver pose estimation presents unique challenges. Drivers sit in a confined space which limits their range of motion. Cameras need to be mounted in certain spaces that are away from blocking street views. Parts of their bodies can be hidden by self and object occlusion. Some body parts can also be off the scene, such as feet and hands when drivers are reaching outside of side windows. This is also a reason why most 3D HPE methods in this application aim to produce poses only for the upper body keypoints. Data collection can be difficult as it can be dangerous when we want actors to perform risky actions such as falling asleep or using a phone while driving. The interior of vehicles can be significantly different from one to another, which may cause issues when a model is deployed in new cars. Because of these reasons, we review selected works that particularly approach the task of 2D HPE or 3D HPE in the context of DMS.

2.5.1 3D Driver Pose Estimation Based on Joint 2D-3D Network

Yao et al. [YLJ⁺19] proposed a CNN model which combined image-based and point-cloud based information for 3D HPE on drivers. Both inputs were captured simultaneously by a Time-of-flight camera. The model consisted of two branches. The first branch, PointConvNet, took point cloud as an input. These convolution layers used 1D kernels to operate on each point individually before aggregated them by a max-pooling layer for each feature dimension. The second branch, ImgConvNet, was a conventional CNN. It was fed with an infrared image. The output of these two branches were concatenated, and went through a MLP classification layer for inference. The advantage of jointly using image and point cloud data is that the model can operate in real time, since the total number of parameters is lower than that of a sole image-based CNN model.

2.5.2 Real-Time Human Body Pose Estimation for In-Car Depth Images

Torres et al. [TOF⁺19] proposed a method for 2D HPE for a vehicle driver using CNN on a depth image. The architecture consisted of a feature extractor and three convolution sub-networks. For input, it was fed with a depth image acquired from a time-of-flight camera, a depth sensor, placed near the windshield in front of the driver. A VGG-19 [LD15], a CNN, was used as a feature extractor. Then, the output features went through

three simultaneous prediction branches. The first sub-network predicted heat maps for each of the 14 body joints. The second sub-network produced Part Affinity Field, which is a vector field indicating the location of body limbs connecting two joints. The last sub-network was a label detection that determined if a joint was present in the image. The overall network was trained with combined losses from each part. During testing, the model outputted the prediction from the first sub-network but only for the joints which were flagged as visible from the third sub-network. In addition, this work also incorporated data augmentation by translating and re-projecting 3D point clouds from the original depth images into new depth images using extrinsic camera parameters.

2.5.3 An Evaluation of Different Methods for 3D-Driver-Body-Pose Estimation

Martin et al. [MVS21] compared different 3D HPE methods based on triangulation of estimated 2D keypoints and on using depth images. They hand-annotated 1,500 poses from the Drive&Act (Section 2.4.2) by manually annotating 2D upper-body keypoints on multiple views and using triangulation methods for the 3D keypoints. These annotations were used as a test set for evaluating two families of methods. The input was the 2D poses produced from OpenPose [CHS⁺19] for each input frame. The first family involved triangulation using different combinations of views, while the second family was based on the depth images that corresponded with the frame. The simplest method was using the depth value at the same coordinate as the detected 2D keypoints. However, the values were inaccurate as they represented the depth at the body surface, not at the center of the joint. The authors found that they could produce more accurate results by using a simple MLP, referred to as Depth Fix, to estimate offset. Although the results were not as accurate as using the triangulation of inputs from multiple views, the Depth Fix method required only a depth image from a single view.

2.5.4 GTFormer

GTFormer [HZG⁺23] is a 3D HPE spatial-temporal neural network using a 2D video sequence as an input. The skeleton model used in GTFormer contains 17 joints. The network is a two-stage 3D HPE model. The 2D HPE was done using the Cascaded Pyramid Network (CPN) [CWP⁺18]. For 2D-to-3D pose lifter, it utilized a graph neural network for spatial features while using a transformer network for temporal features. The unique aspect of GTFormer is that it was trained in two phases. The first was the pre-train phase where the model was trained to recover a sequence of poses by randomly masking a fixed number of joints and frames. This phase was intended to warm up the weights of the model. Then, the training of the second phase was for 3D HPE as usual. In this latter phase, another transformer encoder was used as a regression head which made the model resilient to redundant frames. The model was benchmarked against the Human3.6m dataset and achieved 33.84 mm and 44.50 mm MPJPE for the 2D pose inputs from ground truth and CPN, respectively.

2.5.5 Transfer Learning for Driver Pose Estimation from Synthetic Data

Sagmeister et al. [SSN⁺23] studied the use of synthetic data in the task of driver pose estimation. These synthetic images imitated the images of vehicle drivers in various poses and backgrounds from three different camera positions. To produce the images, they randomized human models, driver poses, car models, light and camera extrinsic parameters. This data was used for pre-training object detection models and 2D HPE models. They evaluated the 2D HPE methods by comparing models pre-trained with this dataset to those pre-trained with other generic datasets. Then, they fine-tuned the model using varying amounts of real-world images from the DriPE dataset (Section 2.4.1). As a result, the fine-tuned models performed much better when real-world data was limited than models trained from scratch. Additionally, the models fine-tuned with their specialized dataset also produced higher accuracy with the same amount of real-world data.

2.5.6 Integrated Driver Pose Estimation for Autonomous Driving

Cao et al. [CHL24] demonstrated an integration of existing deep learning models to simultaneously infer both body and hand poses of vehicle drivers. The model consisted of three parts. The first part used Fast R-CNN [Gir15] to estimate a bounding box of the driver, then used PoseNet [SXW⁺18] for 3D HPE. Similarly, the second part used YOLO-V3 [RF18] for hand detection and SO-HandNet [CTG⁺19] for pose estimation of the hands. The final part was an estimation of the distance of body root (usually located at the pelvis) from the camera using RootDepth [MCL19]. The model then combined the skeleton model by aligning the hand key points and transforming them into the coordinate system of the body. Due to the lack of benchmark data, the study evaluated the model qualitatively by judging the quality of the output with images from frontal view.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Deep Learning Architectures for Human Pose Estimation

This chapter introduces the approach used to construct the 3D HPE pipeline and presents the deep learning architectures employed in our experiments. In Section 3.1, we discuss in detail how top-down 3D HPE can be accomplished through a three-stage pipeline. We then review the models chosen to perform 2D HPE in Section 3.2, which serves as a preprocessing stage. Finally, in Section 3.3, we explore a set of 2D-to-3D pose lifters selected for comparison in the experiments. These lifters were chosen for their diverse deep learning architectures, allowing for a more comprehensive evaluation. We conclude this chapter with a summary table that highlights the key characteristics of these lifting models.

3.1 Three-stage 3D Human Pose Estimation

As we discussed in the previous chapter, there are many approaches to performing 3D HPE, each of which has different advantages. In this work, we chose to perform a top-down and multi-stage approach. The advantage is that we can break 3D HPE down into smaller tasks. These tasks can then be solved with specialized pre-trained models. With this approach, we can focus on the part of the pipeline that we aim to

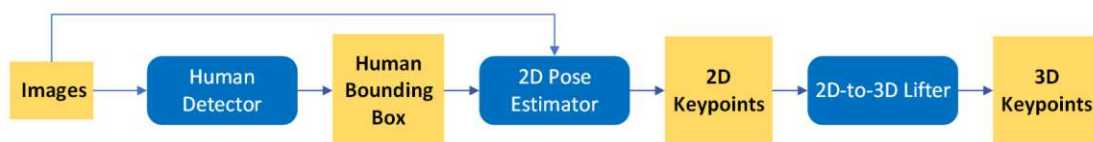


Figure 3.1: General structure of three-stage 3D Human Pose Estimation.

investigate. The top-down HPE starts by locating all people in the scene. Then, the subsequent models can perform pose estimation for each person individually. In this work, we employed a three-stage pipeline: human detection networks for localizing each human, 2D HPE models for extracting joint coordinates in image space, and a 2D-to-3D pose lifter for estimating relative 3D joint positions.

Figure 3.1 outlines the structure of our driver 3D pose estimation pipeline. We can see two types of symbols. The blue boxes represent different machine learning models, while the yellow boxes represent data. We start by employing an object detection model in the first stage. Generally, the model takes an image as input and attempts to locate all of the foreground objects found in the scene. A general object detection model can distinguish several classes of objects; for example, there are 80 classes in the COCO dataset, one of the most used benchmarks [LMB⁺15]. However, we care only about detecting the driver in our case, so only the human class is necessary. The output of the model is a list of bounding boxes, their associated predicted labels, and confidence levels. The list will contain multiple instances if there are numerous humans present in the scene.

For each detection, we feed the bounding boxes and the original images to the next stage: the 2D HPE. The estimator attempts to find the location of every pre-defined anatomical key point, such as eyes, shoulders, and hips. As we can imagine, the visual textures of some body parts are not prominent. Thus, accurate bounding boxes significantly improve the accuracy. Advanced estimators are also able to infer the locations of occluded parts and those located outside of the captured image as well. The model's output is a list of the locations of the key points in pixel coordinates with their confidence scores.

The final part of the pipeline, and the focus of this thesis, is the 2D-to-3D pose lifter. This model accepts 2D key points and produces 3D key points. To improve the prediction, we preprocess the inputs by centering and normalizing them. The 3D coordinates of the output depend primarily on the annotation of scale, origin, and rotation during training.

The following sections briefly review the deep learning architectures selected for each sub-task. In Section 3.2, we start with 2D HPE where we describe our human detector and 2D pose estimator. Both models were considered state-of-the-art at the time of the study. Then, we examine six 2D-to-3D pose lifters consisting of various neural network architectures in Section 3.3 that we will evaluate in our pipeline in the following chapters.

3.2 2D Pose Estimation

Accurate 2D HPE forms the foundation of the estimation pipeline. It serves as a preprocessing step to extract pose information from images, allowing the resulting outputs to be fed into the lifter models. Because object detection and 2D pose estimation, particularly for single-person scenarios, are well-studied problems, existing models perform remarkably well. We used Faster Region-based Convolutional Neural Network (Faster R-CNN) [GDDM14] as the human detector and High-Resolution Network (HRNet) [SXLW19] for 2D HPE. Both are widely employed in the literature [HG17, NTN⁺21,

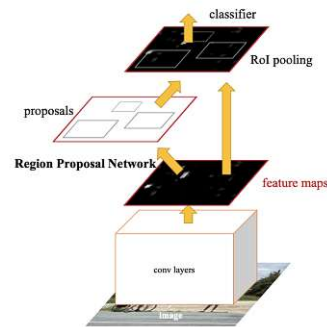


Figure 3.2: Architecture of Faster R-CNN. Figure from [RHGS15].

LXD⁺22, AYKY24, SWS25, GMS25] and demonstrate robust performance, with pre-trained models readily available. Using them also enables direct comparison between our pipeline and the results presented in [SSN⁺23] in the next chapter. This section provides an overview of both models.

3.2.1 Human Detection with Faster R-CNN

The introduction of the Region-based Convolutional Neural Network (R-CNN) [GDDM14] in 2013 proved the usefulness of employing neural networks and region proposals for feature extraction in the object detection task. Since then, several models have built upon the idea, including Fast R-CNN [Gir15], Faster R-CNN [RHGS15], and You Only Look Once (YOLO) [RDGF16]. These models addressed issues in R-CNN for speed and accuracy.

The model, shown in Figure 3.2, operates by performing feature extraction using convolutional layers on an input image to obtain feature maps. A sub-network called the Region Proposal Network then infers the shape and size of a region-of-interest (ROI) that could contain a target object. Next, the ROI pooling layer performs max pooling on the feature maps to produce consistent features before feeding them to the object classifier and the bounding box regressor to obtain the final output. The model outputs the bounding box coordinates indicating where humans might be located, along with their confidence scores.

In our work, we use Faster R-CNN to perform human detection. In situations where there might be multiple humans present, we would need to apply a threshold on the scores to filter out false positives. However, in the datasets we studied, there is at most one person, the driver, in the scene, so we can simply select the bounding box with the highest score.

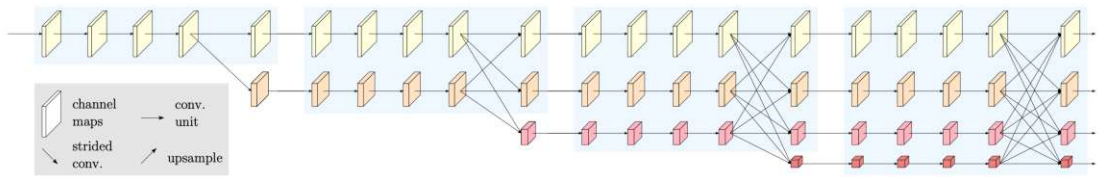


Figure 3.3: Architecture of HRNet. Figure from [SXLW19].

3.2.2 2D Pose Estimation with High-Resolution Network (HRNet)

HRNet [SXLW19] is a 2D HPE network. The model employs a high-to-low and low-to-high resolution approach similar to preceding architectures like the stacked hourglass network [NYD16]. However, the novel difference is that HRNet keeps the high-resolution features throughout the network instead of reconstructing them from lower-resolution representations. This approach enables the precise location of prediction heatmaps.

From Figure 3.3, we can see four segments of the networks highlighted in blue. Each plate in the segments represents features. The size of the plate denotes the difference in resolution. From left to right, the model processes the features with convolution. At the connecting point between each segment, features with different resolutions are added together by applying strided convolution to reduce the resolution or upsample (downward arrow) to increase it (upward arrow). The process continues until the fourth segment, in which the features with the highest resolution are used for prediction. The model also performs the convolutional operations on all resolutions in parallel, resulting in deeper subnetworks for higher-resolution features.

3.3 2D-to-3D Pose Lifter Models

After we have obtained 2D poses, we can then feed the results into a 2D-to-3D pose lifter to infer the 3D poses. Here, the main assumption is that the information contained in the 2D skeletal representation of human postures is sufficient for inferring accurate 3D coordinates of the joints. Figure 3.4, which depicts the model architecture proposed by Martinez et al. [MHRL17], reinforces this assumption and has inspired several follow-up works [WTZ⁺21]. This section provides a summary of selected models based on different types of deep learning architectures. These models are also used to demonstrate the benefits of using synthetic data in subsequent chapters.

3.3.1 Simple Baseline Model for 2D-to-3D Pose Estimation

Martinez et al. [MHRL17] proposed a simple deep learning model that takes 2D poses as input and outputs 3D poses. It was constructed mainly from fully connected layers. The goal was to split the task into 2D pose estimation from an image, which was well-studied, and the uplifting of 2D coordinates to 3D. Most deep learning-based networks in the past aimed to predict the 3D poses directly from images, making it hard to judge whether

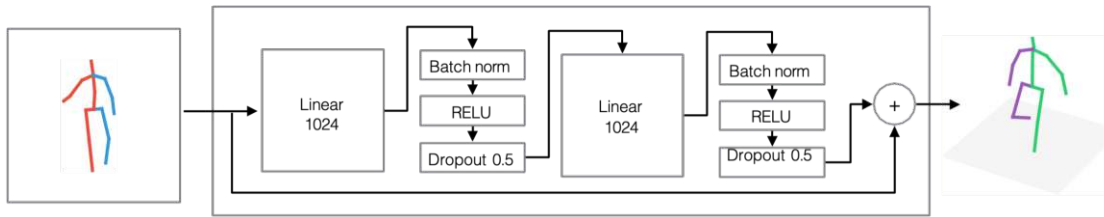


Figure 3.4: Architecture of the Simple Baseline Model. Figure from [MHRL17].

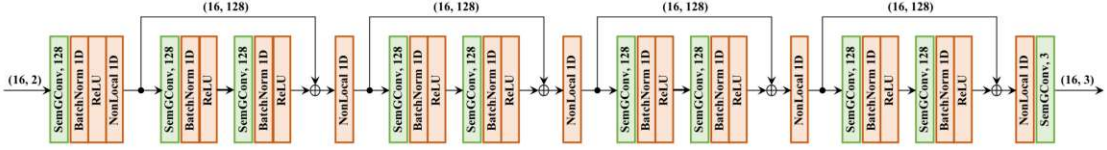
image understanding or depth prediction was the source of inaccuracy. The authors decoupled these components by using the stacked hourglass network [NYD16] to extract 2D poses from images, and a 2D-to-3D pose lifter to predict 3D coordinates. They then compared the predictions with 3D poses obtained from feeding the lifter with ground truth inputs. Their results indicated that 2D pose estimation remains the main cause of error for 3D HPE. They showed how a simple architecture could beat the state-of-the-art models at the time. This success sparked interest in 2D-to-3D pose lifters, and other researchers built upon the idea using more complex architectures like GNNs and transformers, as discussed in the following subsections.

From Figure 3.4, the Simple Baseline Model for 2D-to-3D Pose Estimation (SimpleBL) consists of a linear layer that transforms the input into a 1024-dimensional representation, an output linear layer for regression, and two intermediate building blocks. Each block contains two 1024-dimensional linear layers. The output of each layer is passed through a batch normalization layer followed by a rectified linear unit (ReLU) activation. A residual connection enables the model to bypass the input directly to the output of the second layer when necessary. The model was trained with a dropout probability of 0.5 for regularization. Despite its simplicity, this multi-layer perceptron performs surprisingly well and serves as a strong baseline for evaluating more complex models discussed below.

3.3.2 Semantic Graph Convolutional Networks (SemGCN)

GNN [GSR⁺17] is a type of neural network designed to extract features from graph-structured data. Representing data instances as graphs allows us to encode prior information about the relationships between different entities. We can leverage the graph-like structure of human anatomy by building a graph where nodes represent body joints and edges represent bones connecting those joints, as illustrated in Figure 3.5. Additionally, graphs provide feature order invariance: if all other factors remain the same, we can swap nodes corresponding to left and right body parts without affecting the final result.

Semantic Graph Convolutional Networks (SemGCN) [ZPT⁺19] builds upon the principles of GNN while addressing two key architectural drawbacks when applied to human-body modeling. First, standard GNNs use the same weight matrix for every neighbor, which prevents the network from learning distinct local relationships between specific pairs of nodes. For instance, the relation between the left and right shoulders differs


 Figure 3.5: Architecture of the SemGCN. Figure from [ZPT⁺19].

from that between the right shoulder and right elbow. Second, GNNs may fail to capture dependencies between distant nodes if insufficient message-passing iterations are performed, meaning that relations between the left and right arms could be lost.

SemGCN introduces two components that distinguish it from a vanilla GNN: Semantic Graph Convolutions (SemGConv) and the Non-local Mean (NonLocal) module. SemGConv addresses the first issue by employing a learnable weight matrix that captures relationships for each pair of nodes. The operation can be written as

$$X^{(l+1)} = \sigma(WX^{(l)}\rho(M \odot A)), \quad (3.1)$$

where the representation of every node as a matrix, $X^{(l+1)}$, at the $(l+1)$ th iteration, is computed from its previous value, $X^{(l)}$. W is a shared weight matrix, M is the learnable weight matrix, \odot denotes element-wise multiplication, and A represents a mask indicating whether two nodes are neighbors. $\rho(\cdot)$ refers to the Softmax function, while $\sigma(\cdot)$ denotes the ReLU activation function.

The second component, NonLocal, captures long-range dependencies. Each node representation, $x_i^{(l+1)}$ for i in K nodes, is updated according to the following rule:

$$x_i^{(l+1)} = x_i^{(l)} + \frac{W_x}{K} \sum_{j=1}^K f(x_i^{(l)}, x_j^{(l)}) \cdot g(x_j^{(l)}), \quad (3.2)$$

where $f(\cdot)$ computes the relationship between x_i and x_j , and $g(\cdot)$ transforms the representation of x_j . W_x is a learnable weight matrix for this global operation.

The architecture of SemGCN is illustrated in Figure 3.5. The network begins by encoding the input 2D coordinates with a SemGConv layer followed by a NonLocal layer. It then alternates between two SemGConv blocks with residual connections and a NonLocal block. These blocks are repeated multiple times to increase the network depth. Finally, the model concludes with another pair of SemGConv and NonLocal layers to produce the predicted 3D coordinates.

3.3.3 Graph MLP-Like Architecture for 3D Human Pose Estimation

Li et al., the authors of Graph MLP-Like Architecture for 3D Human Pose Estimation (GraphMLP) [LLG⁺25], borrowed the idea from the MLP-Mixer architecture [THK⁺21], a model originally developed for computer vision. The distinctive feature of MLP-Mixer is

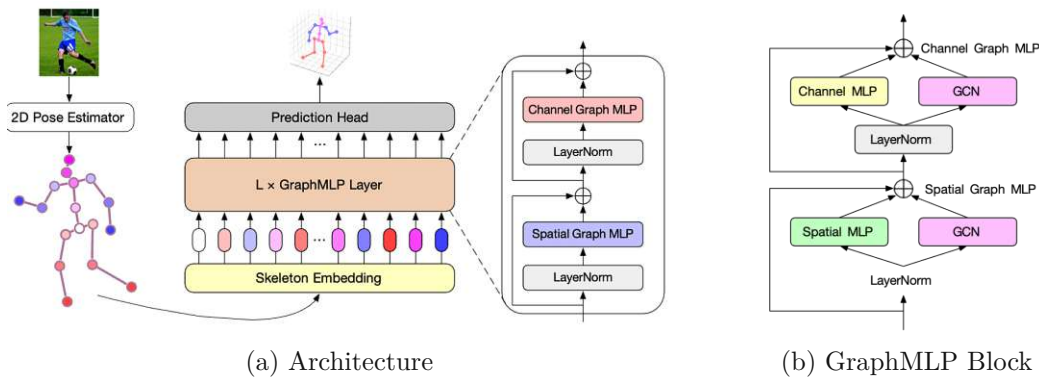


Figure 3.6: Architecture of the GraphMLP. Figures from [LLG⁺25].

that it consists solely of MLP layers, without any convolutional or attention components. Each block employs two types of alternating MLP layers: one processes information within locations (across channels), while the other handles cross-location processing (across tokens).

GraphMLP is a hybrid architecture that incorporates both MLP and GNN components. It was constructed by adding an additional GCN layer to each MLP block in the original MLP-Mixer model. As shown in Figure 3.6b, each GraphMLP block places an MLP layer in parallel with a GCN layer. The rationale behind this design is to provide the network with the ability to capture local interactions between joints through the GCN layers, while simultaneously receiving global contextual information from the MLP layers, particularly the spatial ones.

As illustrated in Figure 3.6a, GraphMLP receives 2D pose coordinates from a 2D pose estimator. The model then embeds the coordinates of each joint into a high-dimensional latent space. Subsequently, it learns both global and local interactions between joints using L layers of GraphMLP blocks. Finally, a linear layer produces the final 3D pose prediction.

3.3.4 Graph Convolution Transformer for 3D Pose Estimation

Graph Convolution Transformer for 3D Pose Estimation (GraFormer) [ZWT22] is a hybrid model that fuses the transformer model [VSP⁺17] with the GNN model. The top part of Figure 3.7 illustrates the overall architecture of GraFormer, while the bottom part shows the details of the network’s main components: GraAttention blocks and ChebGConv.

GraFormer adopts only the encoder part of the original transformer. Initially, the encoder contained two main components: multi-head attention and a linear layer. However, GraFormer replaces the linear layer with a GCN layer equipped with a learnable adjacency matrix, collectively called LAM-GConv. This modified encoder, known as the GraAttention Block, learns the global relationships of body joints. Multi-head attention

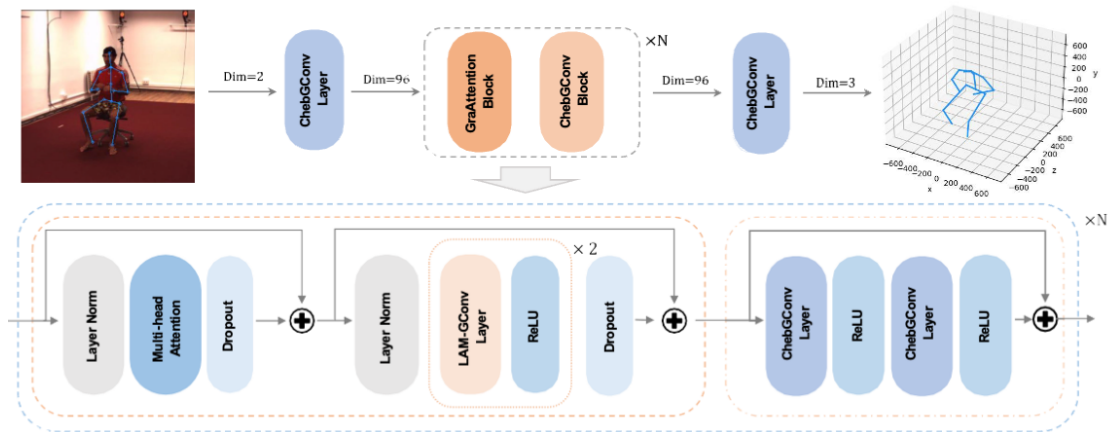


Figure 3.7: Architecture of the GraFormer. Figure from [ZWT22].

aims to detect whether any other body joints contain helpful information for each joint, and then it uses a GCN layer to further determine the relationship between each pair of joints by learning the adjacency matrix.

The other component, ChebGConv, is a type of graph convolution network based on Chebyshev polynomial [DBV16]. The Chebyshev polynomial is a K -order polynomial expressed in a recurrent form. Unlike the LAM-GConv, ChebGConv utilizes a fixed adjacency matrix, which enables the fusion of prior knowledge about human skeletal structure. ChebGConv employs the polynomial as a convolution filter to expand the receptive field of a simple GCN to encompass all neighbors within K hops. This mechanism allows the component to learn from high-order graph structure.

As depicted in the top part of Figure 3.7, GraFormer can be built straightforwardly with all the components described above. It first processes the 2D pose with a ChebGConv layer. Then, it repeatedly applies GraAttention and Chebyshev to deepen the network. Finally, it employs a final ChebGConv layer to predict the 3D pose output.

3.3.5 Single-Frame Lifting Transformer with Error Prediction and Refinement for 3D Human Pose Estimation

Single-Frame Lifting Transformer with Error Prediction and Refinement for 3D Human Pose Estimation (JointFormer) [LBG⁺22] is a pure transformer model equipped with a handful of fine-tuning mechanisms. At first glance, the model resembles a vanilla transformer encoder [VSP⁺17], with each token representing a joint. The model uses an embedding layer to transform the input token into a high-dimensional vector in latent space. Then, a transformer encoder, consisting of a multi-head attention and a linear layer, learns the relationship between each pair of tokens. However, as shown in Figure 3.8, JointFormer incorporates several extra components.

For the embedding layer, the model’s authors opted to use 1D convolutions with a kernel

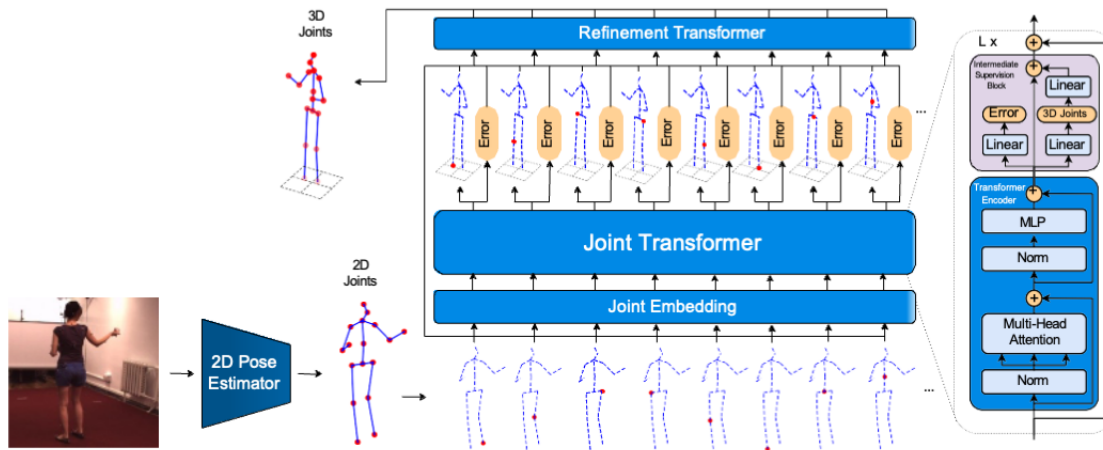


Figure 3.8: Architecture of the JointFormer. Figure from [LBG⁺22].

size of 1 instead of the conventional linear layer. They argue that this approach allows weight sharing across all joints and remains independent of the actual location.

The top-right section of Figure 3.8 illustrates the intermediate supervision block. Intermediate supervision is a technique that shortcuts the loss to each encoder layer. Each encoder must be capable of predicting the 3D joints directly to calculate a loss. This is accomplished by adding a linear layer (the right path) and calculating the loss against the annotated 3D poses. By introducing this loss, each layer of the encoder learns to predict the 3D pose directly, thereby forcing the latter layers to learn more fine-grained discriminate features.

Additionally, we can observe that part of the output of the Joint Transformer section in Figure 3.8 consists of error terms. JointFormer enables the model to predict its errors by incorporating another linear layer similar to the 3D pose prediction head. The model then feeds these error terms together with the input 2D poses and the predicted 3D poses to the final section of the model, called Refinement Transformer. The purpose is to allow this last transformer to improve the prediction by considering its uncertainty given the input 2D poses.

3.3.6 Reprojection Network

Reprojection Network (RepNet) [WR19] is a 2D-to-3D pose lifter that is weakly supervised using unpaired 2D and 3D poses. The network was designed to mitigate overfitting in the lifter, which can occur when the model memorizes the input data. Figure 3.9 illustrates the structure of the training process. There are four main components in RepNet: the pose generator network, the critic network, the camera network, and the re-projection layer. These components are trained jointly using a GAN-based approach.

As is typical for GAN training, there are two alternating phases in training RepNet:

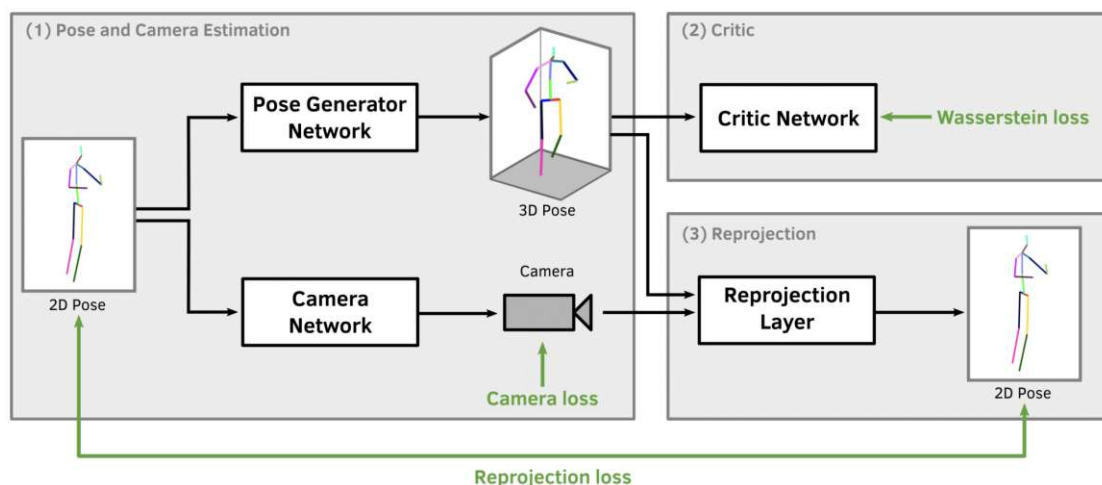


Figure 3.9: Training of the RepNet. Figure from [WR19].

the generator phase and the discriminator phase. In the generator phase, a 2D pose is fed into the pose generator and camera networks. The pose generator network can be any 2D-to-3D pose lifter; in RepNet, the SimpleBL model described previously in Section 3.3.1 is used as the lifter. This network produces a possible 3D pose for the input. The camera network then predicts weak-perspective camera parameters for the 2D pose. These parameters are used to re-project the 3D pose back to 2D through the re-projection layer. The resulting 2D pose is compared with the original input 2D pose to compute a re-projection loss, defined as the Euclidean distance between corresponding joints. Simultaneously, a discriminator loss is calculated from the critic and camera networks. These losses are combined and used to train both the pose generator and the camera networks.

During the discriminator phase, the authors train the critic network to distinguish real 3D poses from those predicted by the pose generator network. First, they feed the 2D pose through the pose generator to produce a predicted 3D pose, which serves as a fake sample. They use any available ground-truth 3D pose as a real sample. The 3D pose does not need to be associated with the 2D pose or originate from the same dataset; it only needs to expose the critic network to the distribution of plausible human 3D poses so that it can classify the fake samples produced by the generator. For each training step, they feed both real and fake samples into the critic network. The goal is to train the critic to output the probability that a given 3D pose is real—producing a high value for real 3D poses and a low value for fake ones. In this setup, RepNet employs the Wasserstein loss function [ACB17] for training the critic, which further encourages the generation of realistic poses.

Additionally, RepNet incorporates a Kinematic Chain Space (KCS) [WAR19] layer into the model. The KCS layer computes additional features related to bone lengths and angular constraints of the human skeleton, helping the model to learn structural symmetry

and the natural limitations of joint angles more effectively.

3.3.7 Comparison of 2D-to-3D Pose Lifter Models

Table 3.1 compares the models used in this thesis. In summary, we have covered the noteworthy families of neural networks in the context of 2D-to-3D pose lifters. Regarding model size, GraphMLP is the largest while SemGCN is the smallest. Model size can roughly indicate the relative training and inference time, with the latter being particularly significant for in-cabin driver pose estimation, which must operate in real-time. In the table, we also note in which chapter we use each model in the experiments.

Model	Architecture	Parameters	Size (MB)	Used in Chapters
SimpleBL [MHRL17]	MLP	4.3 M	17.105	4, 5
SemGCN [ZPT ⁺ 19]	GNN	434 K	1.739	4, 5
GraphMLP [LLG ⁺ 25]	GNN + MLP	9.5 M	37.931	4, 5
GraFormer [ZWT22]	GNN + Transformer	926 K	3.708	4, 5
JointFormer [LBG ⁺ 22]	Transformer	1.2 M	4.786	4, 5
RepNet [WR19]	MLP (trained with GAN)	8.4 M	33.65	6

Table 3.1: Summary of Model Characteristics.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

3D Pose Estimation Pipeline with 2D-to-3D Pose Lifters

In this chapter, we answer the first research question from Section 1.2, namely, how well 2D-to-3D pose lifters perform in the case of synthetic images. In Section 4.1, we describe our goal and the overall study approach. In Section 4.2, we detail the synthetic dataset used. Then, we lay out our setup for the experiment in Section 4.3. We also provide complete implementation details here. In the upcoming chapters, the pose estimation pipeline used in this chapter will be adapted to suit the experiments in those chapters, and thus we will only describe the changes made. Finally, the results are shown and discussed in Section 4.4 and Section 4.4, respectively.

4.1 Objective & Approach

We aim to answer the first research question regarding how well 3D poses can be estimated from monocular synthetic images using 2D-to-3D pose lifters. As discussed in earlier chapters, prior studies have shown that reasonable estimates can be produced from a single image. Here, we seek to evaluate how well these models perform on our synthetic data. The results will also clarify the accuracy trade-offs associated with adopting this approach. The machine learning pipeline implemented in this experiment serves as the foundation for the experiments in subsequent chapters.

To achieve this, we implement a pipeline that first performs 2D HPE on input images and then estimates 3D poses from the resulting 2D keypoints. The core objective is to train the selected 2D-to-3D pose lifters using a synthetic dataset. Since the data are synthetic, the exact coordinates of 2D and 3D keypoints are known, allowing us to use them as input and target, respectively. We then compare the accuracy of these models with the

results from [SSN⁺23], which use triangulation on the same data. Because triangulation utilizes images from multiple views along with their corresponding camera parameters, our single-view pose estimator is not expected to surpass those results. However, if the performance is comparable, it would demonstrate that the 2D-to-3D pose lifter offers a reasonable balance between accuracy and the simplicity of single-view estimation.

4.2 Dataset: SyntheticCabin IR

In this experiment, we make use of the SyntheticCabin IR dataset [SSN⁺23]. It is a synthetic dataset for driver pose estimation created by our project partner, emotion3D¹. It contains both RGB and simulated near-infrared (NIR) images of 3D human models sitting in a driver’s seat with a variety of body movements. The metadata, represented in JSON format, specifies the parameters used for generating the dataset, including the car interiors, the human models, and their pose. The exact positions of the body joints of the driver are given for all keypoints in both 2D and 3D. Synthetic datasets help address the scarcity of 3D ground truth pose annotations for driver monitoring.

The images were generated in Unity² with 3D human models from RenderPeople³, and 3D vehicles from Hum3D⁴. The human poses were generated randomly by an inverse kinematic workflow. For example, the positions of the hands were determined first, then the angles and positions of the elbows and arms were calculated afterward.

The SyntheticCabin IR dataset contains 50,000 infrared images captured from three fixed viewpoints: the driver-side A-Pillar, the codriver-side A-Pillar, and the rear mirror. The views are denoted as *A_Pillar_Driver*, *A_Pillar_Codriver*, and *Rear_Mirror*, respectively. There are 16 different human models, 8 male and 8 female, that differ in age and physical characteristics. Each model is represented by 3,125 images. Figure 4.1 illustrates representative examples from the dataset.

4.3 Experiment Setup

We set up an experiment according to the setup described in [SSN⁺23]. The SyntheticCabin IR dataset was divided into training, validation, and testing sets. The training set contains twelve human models, while the validation and testing sets each contain two models. In total, there are 37,500 images for training and 6,250 images for both the validation and testing sets. We trained the models using the training set and tracked their training progress using the validation set. The testing set was used to compare the final results of the models. The evaluation metrics for this experiment are PJPE and MPJPE. PJPE provides detailed results showing which joints are easier or more

¹<https://emotion3d.ai/>

²<https://unity.com/>

³<https://renderpeople.com/>

⁴<https://3dmodels.org/>

SyntheticCabin IR

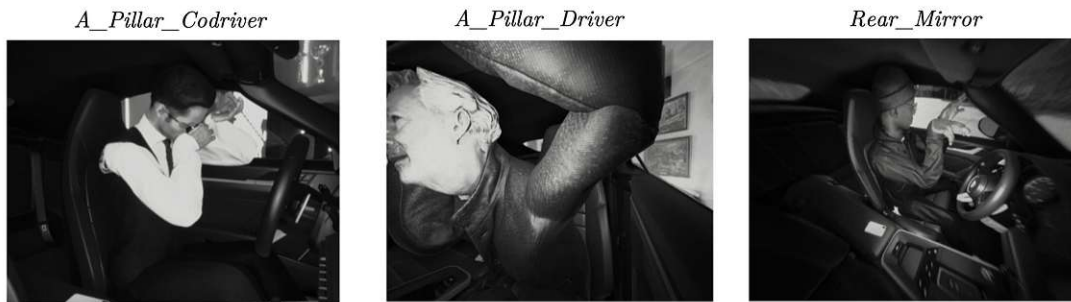


Figure 4.1: Example images from the SyntheticCabin IR dataset.

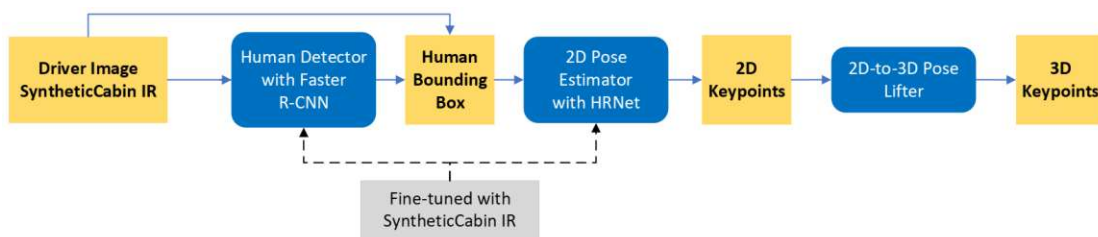


Figure 4.2: 3D pose estimation pipeline for the experiment in Chapter 4.

challenging to estimate, whereas MPJPE represents the mean of PJPE values, providing an overall performance metric.

The process was performed separately for each of the three views in SyntheticCabin IR: *A_Pillar_Driver*, *A_Pillar_Codriver*, and *Rear_Mirror*. The 2D HPE component was fixed by preprocessing the 2D poses. We then trained five 2D-to-3D pose lifters—namely, SimpleBL, SemGCN, GraphMLP, GraFormer, and JointFormer—on these resulting 2D poses and compared their results. Through this approach, we can identify the best viewing angle for camera installation and determine which deep learning architecture performs better. All results were compared against those from the multi-view triangulation method [SSN⁺23]. We expected the MPJPE from the 2D-to-3D pose lifters to be closely comparable.

For implementation, we constructed a pipeline for 3D pose estimation in three stages, as described in Section 3.1. Figure 4.2 illustrates the pipeline setup. The input driver images were taken from the SyntheticCabin IR dataset. Here, we used Faster R-CNN and HRNet for detecting the driver and performing 2D HPE, respectively. Both models are publicly available and were pre-trained on the COCO dataset. We then fine-tuned them on the SyntheticCabin IR dataset to adapt to the texture of the synthetic images. The estimated 2D poses were fed into one of the 2D-to-3D pose lifters listed above to learn from the annotated 3D poses of the synthetic dataset. We then evaluated the trained models on the test dataset with the same camera views as the training set and compared

the results.

The experiments were run on a workstation equipped with an AMD Ryzen 9 5900X 12-Core Processor, 32 GB of RAM, and an NVIDIA TITAN Xp GPU with 12 GB of VRAM. The operating system was Ubuntu 22.04.5 LTS. To ensure consistency and reproducibility, we ran the experiments inside a Docker container based on Ubuntu 20.04.5 LTS, using the official PyTorch image `pytorch/pytorch:2.0.1-cuda11.7-cudnn8-devel`. The experiments were implemented in Python using the deep learning library PyTorch [PGM⁺19] version 2.0.1, along with PyTorch Lightning [FT], a framework that facilitates setting up training pipelines in PyTorch.

4.3.1 Preparation of 2D Human Poses

We treat the first two stages of the pipeline as a fixed part of the experiment. Thus, we can preprocess the whole dataset and produce a 2D pose dataset for training the 2D-to-3D pose lifters. Here, we detail the steps we take to preprocess the dataset.

Preprocessing for SyntheticCabin IR

The SyntheticCabin IR dataset includes generated images and annotation files for each frame. We preprocess the dataset to facilitate its use as input to the pipeline. The images are rescaled to a resolution of 1280×1024 pixels. They are then organized into three folders—*train*, *val*, and *test*—according to the provided data split. Finally, we rename each image file using its corresponding frame ID.

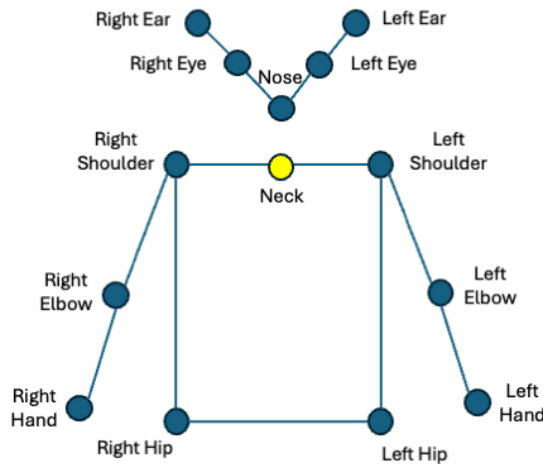


Figure 4.3: Skeleton pose representation showing keypoints used for upper-body driver pose estimation.

For the annotations, the dataset includes detailed locations of multiple body parts, which were used as parameters for generating the images. We extract only the 2D and 3D annotations corresponding to the 13 upper-body keypoints, following [SSN⁺23].

Figure 4.3 illustrates the selected keypoints, which are labeled as if the model were facing the observer. As shown in the dataset images in Section 4.2, in-cabin cameras typically capture only the driver’s upper body. The steering wheel often occludes the lower body, while those parts generally exhibit limited movement and therefore contribute less to driver behavior analysis. Due to these domain-specific characteristics, our skeleton pose annotation includes only upper-body keypoints.

Unlike the conventional approach of centering the keypoint coordinates at the pelvic bone, we use the neck—denoted by the yellow dot in Figure 4.3—as the reference point. This point is computed as the mean position of the left and right shoulders. Finally, we generate JSON description files in the COCO-style human keypoint annotation format, originally introduced in the COCO 2017 Keypoint Detection Task [LMB⁺15], for each data split.

Human Detector

In the first stage of the pipeline, we employ the Faster R-CNN to detect drivers in the input images. The implementation we use is one with a feature pyramid network as a feature extractor, `R-50-FPN-Caffe-3x`, from MMDet [CWP⁺19], a software toolkit for object detection. The selected version, `faster-rcnn_r50-caffe_fpn_ms-1x_coco-person`, was pre-trained in Caffe [JSD⁺14] specifically to detect the person class in the COCO dataset.

As there are significant differences in image texture between synthetic and real-world images, we need to fine-tune the model on the SyntheticCabin IR dataset. For this, we used MMEngine [MME22], which provides tools for creating training and evaluation pipelines. In the pipeline, we also perform random flipping and resizing for data augmentation. The training was conducted using the training set for one epoch. Table 4.1 shows the Average Precision (AP) — the area under the precision-recall curve for the person class — of the fine-tuning results for each camera view on the test set for three Intersection over Union (IoU) thresholds: $\text{IoU} \geq 50$ (AP_{50}), $\text{IoU} \geq 75$ (AP_{75}), and IoU threshold from 50 to 95 ($AP_{50:95}$). We can see that the performance is good; however, it lags behind that reported in [SSN⁺23] for the AP, especially for the *Rear_Mirror* view. The reduction in performance may have been influenced by the model occasionally generating multiple bounding boxes for the same person (false positives) or boxes that were excessively large. We did not further optimize the detector, as the results shown in Table 4.2 were adequate for the 2D pose estimation in the subsequent stage.

After obtaining a fine-tuned human detector, we perform inference on all the images in SyntheticCabin IR and save the results for further steps. During inference, the detector may produce multiple bounding boxes. Because the SyntheticCabin IR dataset always contains a single person, the driver, in the scene, we simply select the bounding box with the highest confidence score. Note that the bounding box convention of MMDet is in `xyxy` format, which specifies a bounding box with the coordinates of the top-left and

the bottom-right corners. This format is different from the COCO dataset, which uses the xywh format, the top-left coordinate along with the width and the height of the box.

SyntheticCabin IR (Test Set)				
Evaluation result for fine-tuned human detector (Faster R-CNN)				
Model	Metrics (%)	<i>A_Pillar_Driver</i>	<i>A_Pillar_Codriver</i>	<i>Rear_Mirror</i>
[SSN ⁺ 23]	<i>AP</i> _{50:95}	98.0	98.3	89.9
	<i>AP</i> ₅₀	100.0	100.0	100.0
	<i>AP</i> ₇₅	99.0	99.0	99.0
Ours	<i>AP</i> _{50:95}	95.9	96.8	61.3
	<i>AP</i> ₅₀	99.0	99.0	99.0
	<i>AP</i> ₇₅	99.0	99.0	85.3

Table 4.1: Evaluation result for fine-tuned human detector (Faster R-CNN)

2D Pose Estimator

The second stage of the pipeline is the 2D pose estimator. We use the HRNet model. Similar to the human detector, we employ the one implemented by MMPose [MMP20], a software toolkit for human pose estimators. The selected pre-trained model is `td-hm_hrnet-w48_udp-8xb32-210e_coco-384x288`. This model is built with an HRNet architecture with a width of 48 channels in the highest resolution branch. The model was trained with a batch size of 256 on 384x288 pixel images from the COCO dataset for 210 epochs. During training, multiple data augmentations were used, such as random flip, bounding box transformation, and affine transformation. The output of the model is heatmaps, one for each keypoint. We can use the coordinates of the highest value in the heatmap for the predicted keypoint.

Because the HRNet takes a synthetic image and a bounding box as input, we also need to fine-tune the model as with the human detector. Again, we use MMEngine for fine-tuning with a batch size of 32 images for five epochs. We train the model using both ground truth bounding boxes and 2D poses from the SyntheticCabin IR dataset. The training pipeline also utilizes the same data augmentations.

As with the human detector, we evaluate the fine-tuned model using the testing set. We test the model under two setups: supplying the model with the ground truth bounding boxes or the predicted results from the human detector. Using the ground truth tells us how good the trained HRNet is, while using the predicted ones indicates the combined performance of the first two stages. The evaluation metric is the AP as well, with the same levels of thresholds: 50%, 75%, and 50% to 95%. However, for 2D HPE, the similarity between the predicted and the ground truth keypoints is measured with a metric called Object Keypoint Similarity (OKS), which measures how close the overall detected poses are on a scale between 0 (lowest) and 1 (highest), accounting for differences

in scale. Table 4.2 contrasts the results with [SSN⁺23]. We see that for the $AP_{50:95}$ that our fine-tuned HRNet yields slightly better results. Similarly, the evaluation results for the bounding boxes from the Faster R-CNN are at a similar level. We can see that the poorer performance of Faster R-CNN does not affect the AP of the HRNet for the *Rear_Mirror* view.

Finally, we run inference for all the images in SyntheticCabin IR using the bounding boxes from Faster R-CNN. The estimated poses from the HRNet are 17 full body keypoints in image coordinates, including those occluded or outside of the image. As we discussed, our skeletal representation uses only 13 upper body keypoints. Thus, we discard the rest.

SyntheticCabin IR (Test Set)					
Evaluation result for fine-tuned 2D Pose Estimator (HRNet)					
Model	Bounding Box	Metrics (%)	<i>A_Pillar_Driver</i>	<i>A_Pillar_Codriver</i>	<i>Rear_Mirror</i>
[SSN ⁺ 23]	Ground Truth	$AP_{50:95}$	87.6	94.1	84.5
		AP_{50}	99.0	99.0	99.0
		AP_{75}	96.9	99.0	91.7
Faster R-CNN		$AP_{50:95}$	87.8	94.0	84.7
		AP_{50}	99.0	99.0	99.0
		AP_{75}	96.9	99.0	91.7
Ours	Ground Truth	$AP_{50:95}$	90.6	96.7	91.7
		AP_{50}	99.0	100.0	99.0
		AP_{75}	94.9	99.0	98.0
Faster R-CNN		$AP_{50:95}$	93.8	90.3	86.4
		AP_{50}	100.0	99.0	99.0
		AP_{75}	94.8	99.0	95.9

Table 4.2: Evaluation result for fine-tuned 2D Pose Estimator (HRNet)

4.3.2 Training of 2D-to-3D Pose Lifters

The final stage of the pipeline is the 2D-to-3D pose lifter. Here, we train the model using SyntheticCabin IR as before, using ground-truth 2D poses as input and 3D poses as ground truth. At this stage, we select five models to represent various deep learning architectures. For the model implementation, we aim to source them from the official GitHub⁵ repositories of the models’ authors if a PyTorch implementation is available. If that is not the case, we review alternative sources on GitHub for reliable implementations. For SimpleBL, the original implementations is in TensorFlow, so we use an implementation by Motoki Kimura⁶. The official implementation for SemGCN⁷,

⁵<https://github.com/>

⁶<https://github.com/motokimura/3d-pose-baseline-pytorch>, accessed on 2023-06-06.

⁷<https://github.com/garyzhao/SemGCN>, accessed on 2023-08-09

GraphMLP⁸, GraFormer⁹, and JointFormer¹⁰ are available on GitHub.

For the training pipeline, we use PyTorch Lightning [FT] to help simplify the process. We wrap the model in a `LightningModule`. Then, we structure the input tensor according to each model’s specification. We normalize the size of the 2D poses by min-max scaling using the range of the coordinates with respect to the pose itself, as opposed to the global range of the dataset. Thus, the poses in the input will be on a similar scale, and they fit into a unit square.

During training, we use PyTorch Lightning’s `Trainer` to run the training loop with a random seed of 1234. The training was done in batches of 64 poses, with any incomplete final batch for each epoch being dropped. Adam was used as the optimizer. The learning rate was set using a step scheduler, starting at 0.001 and decaying by a factor of 0.96 each epoch for all models. Training continued for a maximum of 200 epochs with early stopping criteria applied. Every five epochs, the trainer evaluated the MPJPE of the validation set, and training stopped if the validation MPJPE failed to improve for five consecutive evaluations. This early stopping mechanism helped the models converge effectively given the training data, while the learning rate scheduler ensured gradual adjustment of the optimization process. Additionally, we employed gradient clipping at 1.0 to stabilize the learning progress [Mik12].

The loss function of 2D-to-3D pose lifter is the mean squared error (MSE) between annotated 3D pose, $J_i \in \mathbb{R}^{K \times 3}$, and the predicted 3D pose, $J_i^* \in \mathbb{R}^{K \times 3}$.

$$\mathcal{L} = \frac{1}{N} \sum_{i \in \mathcal{A}} \|J_i - J_i^*\|_2^2$$

Where N is the number of samples in each batch, K is the number of joints, \mathcal{L} denotes the loss, and \mathcal{A} is a set of annotated keypoints. $\|\cdot\|^2$ is the L^2 norm.

4.4 Results

During testing, we used 2D poses obtained from inference with Faster R-CNN and HRNet. We then compared the results of our models with the triangulation-based results from [SSN⁺23], as shown in Table 4.3.

Our results are presented in Table 4.4, Table 4.5, and Table 4.6. We report PJPE for all thirteen joints and MPJPE for each camera setup. We also include the average PJPE and MPJPE across all models, as there was no single best lifter model for every configuration. The bold numbers indicate the lowest PJPE and MPJPE for each joint and setup/model.

⁸<https://github.com/Vegetebird/GraphMLP>, accessed on 2023-12-20.

⁹<https://github.com/Graformer/GraFormer>, accessed on 2023-12-21.

¹⁰<https://github.com/seblutz/JointFormer>, accessed on 2023-12-26.

It should be noted that acronyms are used in the tables due to space constraints. Specifically, *Dri*, *CoDri*, and *Rear* denote the *A_Pillar_Driver*, *A_Pillar_Codriver*, and *Rear_Mirror* views, respectively. *All_three* refers to a three-image setup using images from all three views. Lowercase prefixes *l* and *r* correspond to the left and right sides, respectively.

Figure 4.4 shows three examples of estimated 3D poses produced by JointFormer for each view of SyntheticCabin IR. The green skeletal figures represent the ground-truth poses, while the black figures show the poses inferred by JointFormer. Each row presents three views of the same pose to provide a better visual perspective.

Views Setup	Per Joint Position Error (PJPE) [mm]												MPJPE [mm]	
	Nose	lEye	rEye	lEar	rEar	lShoulder	rShoulder	lElbow	rElbow	lWrist	rWrist	lHip		rHip
<i>Dri/CoDri</i>	19.2	17.2	16.9	18.0	15.6	24.0	23.0	35.4	28.5	30.5	25.7	17.10	22.4	22.6
<i>CoDri/Rear</i>	32.3	32.0	29.2	37.6	24.6	50.6	24.3	128.5	21.4	92.8	49.7	65.4	26.2	47.3
<i>Dri/Rear</i>	22.2	19.8	19.2	18.8	17.5	22.9	23.2	50.0	30.2	37.50	35.0	30.0	20.6	26.7
<i>All_three</i>	24.6	23.0	21.8	24.8	16.8	32.3	23.5	71.3	26.7	53.6	46.8	37.5	23.0	32.2

Table 4.3: Evaluation results on the test set using triangulation, reported in [SSN+23].

Models	Per Joint Position Error (PJPE) [mm]												MPJPE [mm]	
	Nose	lEye	rEye	lEar	rEar	lShoulder	rShoulder	lElbow	rElbow	lWrist	rWrist	lHip		rHip
SimpleBL	28.3	27.3	26.3	23.9	20.7	18.3	18.3	87.0	63.2	54.6	45.0	40.9	34.1	37.5
SemGCN	30.0	32.1	30.7	26.9	25.3	19.7	23.8	93.6	63.2	56.7	46.6	49.2	36.9	41.1
GraphMLP	27.3	28.7	28.7	25.7	22.0	21.7	17.3	85.8	66.5	56.9	43.0	41.1	40.7	38.9
GraFormer	31.2	29.2	28.4	27.5	24.1	17.9	17.1	85.7	62.6	55.6	47.6	37.3	36.0	38.5
JointFormer	29.9	29.2	27.4	26.3	21.5	18.1	17.7	87.2	64.5	53.5	42.2	39.0	33.8	37.7
Average	29.3	29.3	28.3	26.1	22.7	19.1	18.8	87.8	64.0	55.5	44.9	41.5	36.3	38.7

Table 4.4: Evaluation result on the test set for *A_Pillar_Codriver* pillar view.

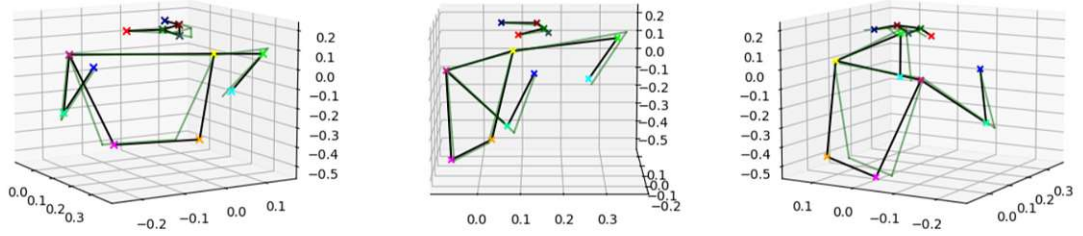
Models	Per Joint Position Error (PJPE) [mm]												MPJPE [mm]	
	Nose	lEye	rEye	lEar	rEar	lShoulder	rShoulder	lElbow	rElbow	lWrist	rWrist	lHip		rHip
SimpleBL	26.2	24.3	26.1	21.9	22.4	16.3	16.3	55.2	65.2	39.2	42.3	30.8	31.9	32.4
SemGCN	26.5	26.0	25.9	21.6	23.2	20.7	15.1	56.2	65.8	43.7	50.5	34.9	35.7	34.5
GraphMLP	23.8	22.5	24.3	21.3	22.8	19.4	17.9	57.5	64.1	39.7	40.0	41.2	39.9	33.7
GraFormer	25.8	23.1	24.2	23.0	22.7	17.1	19.2	63.2	69.8	39.3	45.3	34.0	33.7	34.1
JointFormer	24.8	22.9	24.2	21.2	22.0	15.5	15.5	53.3	64.5	38.2	45.1	33.3	31.1	31.9
Average	25.4	23.8	24.9	21.8	22.6	17.8	16.8	57.1	65.9	40.0	44.6	34.8	34.5	33.3

Table 4.5: Evaluation result on the test set for the *A_Pillar_Driver* view.

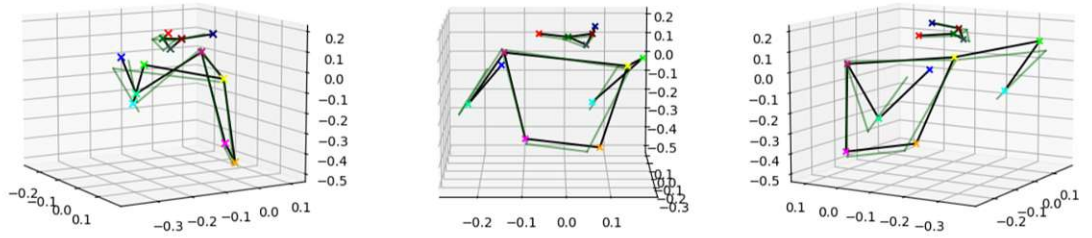
Models	Per Joint Position Error (PJPE) [mm]												MPJPE [mm]	
	Nose	lEye	rEye	lEar	rEar	lShoulder	rShoulder	lElbow	rElbow	lWrist	rWrist	lHip		rHip
SimpleBL	26.9	24.2	24.7	20.6	18.9	17.2	17.2	86.4	52.8	55.4	56.1	29.1	28.5	35.4
SemGCN	25.3	24.0	23.5	22.0	19.8	18.8	16.7	87.6	56.0	57.6	59.8	28.2	30.0	36.2
GraphMLP	27.3	25.4	25.1	23.9	19.6	18.7	19.1	85.8	48.2	53.1	50.7	25.7	29.0	34.9
GraFormer	29.1	25.4	25.3	25.4	19.9	18.2	17.8	88.7	54.9	57.5	65.1	30.6	31.3	37.8
JointFormer	24.8	23.4	23.5	21.3	18.5	15.8	15.8	88.2	53.6	55.6	53.3	28.5	26.9	34.7
Average	26.7	24.5	24.4	22.6	19.4	17.7	17.3	87.3	53.1	55.8	57.0	28.4	29.1	35.8

Table 4.6: Evaluation result on the test set for the *Rear_Mirror* view.

JointFormer (MPJPE=28.0 mm)
A_Pillar_Codriver



JointFormer (MPJPE=36.5 mm)
A_Pillar_Driver



JointFormer (MPJPE=21.9 mm)
Rear_Mirror

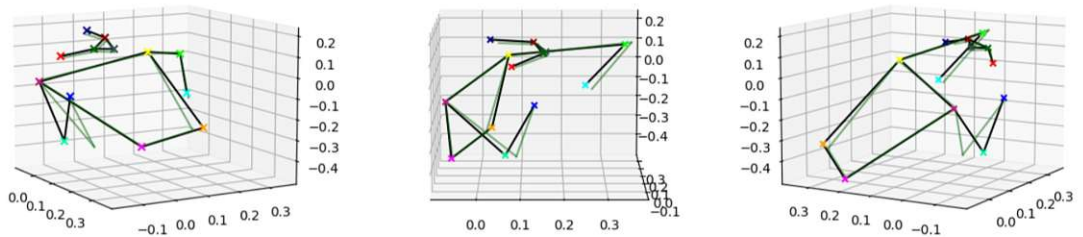


Figure 4.4: Example outputs for each image view of SyntheticCabin IR.

Comparing Table 4.3 with our results, it is evident that triangulation performs better overall, consistently yielding lower MPJPE scores than all 2D-to-3D pose lifter configurations. This outcome is expected, as the method from [SSN⁺23] utilized images from two vantage points—a clear advantage that cannot be matched by a single-image approach. Moreover, their method relied on the ground-truth camera parameters provided by the synthetic data generator, offering an additional performance advantage over our approach.

However, the results obtained from our 2D-to-3D pose lifter approach were comparatively close. All single-image models performed better than the *CoDri/Rear* setup. The best-performing view for a 2D-to-3D pose lifter was *A_Pillar_Driver*, which produced results comparable to the *All_three* triangulation setup, with an average difference in MPJPE of only 1.1 mm. Notably, the JointFormer model even delivered better results than the *All_three* configuration.

Regarding per-joint errors, the 2D-to-3D models were more accurate at predicting the locations of joints above the shoulders than those below, similar to the triangulation method. The prediction of the shoulders was the most accurate; however, this may be attributed to our use of the neck as the root joint.

In summary, the experiment in this chapter demonstrates that 2D-to-3D pose estimation can produce reasonably accurate 3D poses. Although triangulation methods with suitably chosen views outperform these models, the monocular approach is far simpler to deploy. It does not require the calibration of multiple cameras or precise camera parameters, and the equipment cost is substantially lower. This chapter establishes a foundation for the experiments in the following chapters, where synthetic data are utilized to improve 3D pose estimation on real-world data.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Transfer Learning from Synthetic Data

In this chapter, we explore transfer learning for 3D pose estimation using synthetic data. Our approach involves pre-training the 2D-to-3D pose lifter on ground-truth 3D poses from synthetic data and then fine-tuning the model with real-world data. The goal is to assess whether this strategy can reduce the reliance on real-world data and the need for 3D pose annotations. To evaluate this, we compare the performance of two models with identical 2D-to-3D pose lifter architectures: one pre-trained with synthetic data and one trained from scratch. If the results show that the pre-trained model achieves better performance with the same amount of real-world data, then pre-training can be considered beneficial for the lifting models.

We outline the objectives and overall approach in Section 5.1. The two datasets used in this experiment—the SyntheticCabin IR 1M and Drive&Act—are introduced in Section 5.2 as the synthetic and real-world datasets, respectively. The experimental setup and implementation details are described in Section 5.3. Finally, the results and corresponding discussion are presented in Section 5.4 and Section 5.5, respectively.

5.1 Objective & Approach

This chapter addresses the second research question: how effectively knowledge learned from synthetic data can be transferred to real-world data. Transfer learning refers to the practice of adapting a model trained on one domain or distribution to another. If the model learns useful representations from the source data, it can achieve strong performance on the target domain with fewer training samples and epochs. We previously observed the benefits of transfer learning when training the human detection and 2D

HPE models in the preceding chapter, where only a few fine-tuning epochs were required to obtain satisfactory results. A similar effect is expected for the case of 2D-to-3D pose lifters.

Because well-trained human detectors and 2D HPE models are widely available, we leverage them for the first two stages of the pipeline. Our approach pre-trains the 2D-to-3D pose lifters using annotated 2D and 3D poses from synthetic data. By focusing on skeletal keypoints, we can accelerate pre-training, as the number of parameters is substantially smaller than in image-based models. Moreover, synthetic and real 3D poses are generally more similar to each other than their corresponding image representations. Consequently, the 2D-to-3D pose lifters may be able to adapt more efficiently even when trained on a limited amount of real-world data.

To evaluate this hypothesis, we conducted an experiment comparing 2D-to-3D pose lifter models trained from scratch with those pre-trained on synthetic data. The experiment involved gradually increasing the amount of real-world data used for training and observing whether the pre-trained models achieved superior performance under equivalent data conditions.

5.2 Datasets

Our approach relies on both a synthetic dataset and a real-world dataset. The former is used for pre-training, while the latter is used for fine-tuning 2D-to-3D pose lifters and evaluation. Similar to the previous chapter, we also use synthetic data from the SyntheticCabin project; however, we employ an updated version of the dataset. For the real-world data, we use the Drive&Act dataset [MRH⁺19], which was originally developed for driver action recognition using multi-modal input data, including images and 3D poses. Figure 5.1 presents example images from both datasets.

5.2.1 SyntheticCabin

The synthetic dataset used in this experiment is an updated version of the SyntheticCabin dataset, called SyntheticCabin IR 1M, which was generated with several enhancements. This dataset contains a total of 1 million generated images—hence its name. The human models are identical to those in the original version. The movements were generated in a similar manner, with the addition that the models could now place their hands on the steering wheel. The lighting was simulated using ray tracing, and ten camera views are available. Furthermore, additional noise was added to the camera extrinsics, meaning that the camera angles and positions can vary slightly even within the same view.

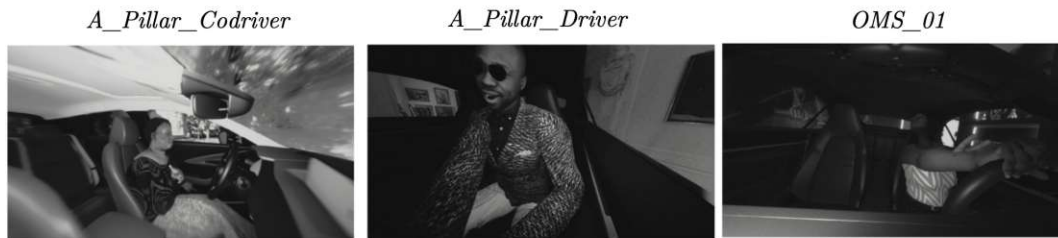
5.2.2 Drive&Act

As mentioned in Section 2.4.2, the Drive&Act dataset [MRH⁺19] is designed for DMS. To the best of our knowledge, it remains the only publicly available dataset with real camera images that provides 3D pose annotations in this domain. Although the annotations were

not obtained using motion capture, they represent the best available option. A follow-up study by the dataset authors reported that the automated annotations generated by OpenPose [CHS⁺19] differed from manually refined annotations by an average of 39.4 mm [MVS21].

The dataset contains 15 subjects (4 female and 11 male). The videos were recorded at a resolution of 1280×1024 pixels and 30 frames per second using five near-infrared cameras. In our experiments, we used only the inputs from three frontal camera views mounted on the A-Pillars (driver and co-driver sides) and the rearview mirror. Unlike the SyntheticCabin dataset, Drive&Act refers to these views as *a_column_driver*, *a_column_co_driver*, and *center_mirror*.

SyntheticCabin IR 1M



Drive&Act



Figure 5.1: Example images from SyntheticCabin IR 1M, and Drive&Act dataset [MRH⁺19].

5.3 Experiment Setup

We performed the experiment to determine whether using synthetic data for pre-training improves learning. Our hypothesis was that if two models are trained with the same amount of real-world data and the one pre-trained with synthetic data produces better results, then the use of synthetic data is beneficial.

We conducted the experiment using the following data split. Among the 15 subjects in Drive&Act, subject number 10 was reserved for validation, and subjects numbered 11–14 were used for testing. The remaining subjects were allocated for training. We experimented by gradually increasing the amount of training data from Drive&Act. For smaller subsets, the training data consisted of a fraction of a subject’s frames; subsequently, we increased the subset size to include multiple subjects. The models were first trained with fractions corresponding to 5%, 10%, 25%, 50%, and 100% of each subject’s frames. Training was repeated for all eight training subjects individually. The procedure is illustrated in Algorithm 5.1.

A sampling rate of 1 Hz was used, resulting in 2048 frames per subject. Thus, 5% corresponded to approximately 102 frames. Next, we trained the models with data from two and four subjects. Each configuration was repeated eight times with randomly selected subject combinations to obtain more reliable results, maintaining consistency with the eight runs used in the percentage-based experiments. The final configuration utilized all eight subjects, encompassing the entire training set, and was therefore executed only once. This procedure is illustrated in Algorithm 5.2.

During validation and testing, we used the complete validation and test sets to maintain consistent evaluation across all experiments. Note that the validation and test sets employed the same camera views as the training set, except when the model was trained with *All Views*. This design enabled us to evaluate model performance across varying amounts of real-world training data for both pre-trained and from-scratch models.

The evaluation metric used for all settings was MPJPE. However, Drive&Act annotated only joints visible in at least two cameras, since the 3D pose annotations were generated by triangulation. Therefore, the MPJPE was computed only on the annotated joints.

For each training data size, we calculated two sets of metrics: (1) the average MPJPE for each experiment configuration, and (2) the average improvement in MPJPE resulting from pre-training. For the average MPJPE, we computed the mean and standard error across subsets: across subjects for Algorithm 5.1, and across subject combinations for Algorithm 5.2. These calculations were performed separately for models with and without pre-training, as shown in line 12 of both algorithms.

To quantify the improvement from pre-training, we paired the MPJPE values from the same training subset—one from the pre-trained model and one from the baseline model trained from scratch. We then computed the difference in MPJPE for each pair, representing the gain from pre-training. This pairing step was implemented at line 10 in the algorithms. Finally, we calculated the mean and standard error of these differences across all pairs, as shown at line 13. The standard error of the mean (SE) was computed using SciPy’s `scipy.stats.sem` function [VGO⁺20]. Note that the SE could not be calculated for the configuration using all subjects, since this case involved only a single training combination.

Algorithm 5.1: Transfer learning experiments using subset frames of each subject.

```

1 Initialize:  $S = [5\%, 10\%, 25\%, 50\%, 100\%]$ 
2 for each subset size  $s_i$  in  $S$  do
3   for each subject  $a_i$  in  $\{a_1, \dots, a_8\}$  do
4     Initialize:  $M = \{\text{Model with pre-training, Model without pre-training}\}$ 
5     Sample  $s_i$  frames from subject  $a_i$  without replacement.
6     for each model  $m$  in  $M$  do
7       Train  $m$  using the sampled data with early stopping based on
       validation loss.
8       Evaluate  $m$  on the test set and record MPJPE.
9     end
10    Compute the difference in MPJPE between models with and without
    pre-training.
11  end
12  Compute the mean and standard error of MPJPE separately for models with
  and without pre-training.
13  Compute the mean and standard error of the MPJPE differences.
14 end

```

Algorithm 5.2: Transfer learning experiments using combinations of subjects.

```

1 Initialize:  $N = \{2, 4, 8\}$  subjects per experiment
2 for each combination size  $n$  in  $N$  do
3   repeat
4     Sample  $n$  subjects from the 8 available subjects  $\{a_1, \dots, a_8\}$  without
     replacement.
5     Initialize:  $M = \{\text{Model with pre-training, Model without pre-training}\}$ 
6     for each model  $m$  in  $M$  do
7       Train  $m$  using the sampled data with early stopping based on
       validation loss.
8       Evaluate  $m$  on the test set and record MPJPE.
9     end
10    Compute the difference in MPJPE between models with and without
    pre-training.
11  until 8 times;
12  Compute the mean and standard error of MPJPE separately for models with
  and without pre-training.
13  Compute the mean and standard error of the MPJPE differences.
14 end

```

5.3.1 Implementation

In this experiment, we implemented a three-stage 3D HPE pipeline following the same procedure as described in the previous chapter. The implementation details outlined there remain applicable here. The main differences concern the selection of datasets and the choice of models appropriate for each dataset. Therefore, this section focuses on the modifications introduced to the pipeline rather than repeating descriptions already provided.

The implementation comprises three components, as depicted in Figure 5.2. First, we pre-trained the 2D-to-3D pose lifters using synthetic 2D and 3D poses. Then, we prepared the 2D poses from the real-world data by performing 2D HPE. One advantage of using synthetic data is that it provides exact 2D projections of 3D poses, even in cases of occlusion. Thus, these annotations can help the model infer the locations of hidden joints even after fine-tuning. Finally, we fine-tuned the lifter with the estimated 2D poses and the annotated 3D poses.

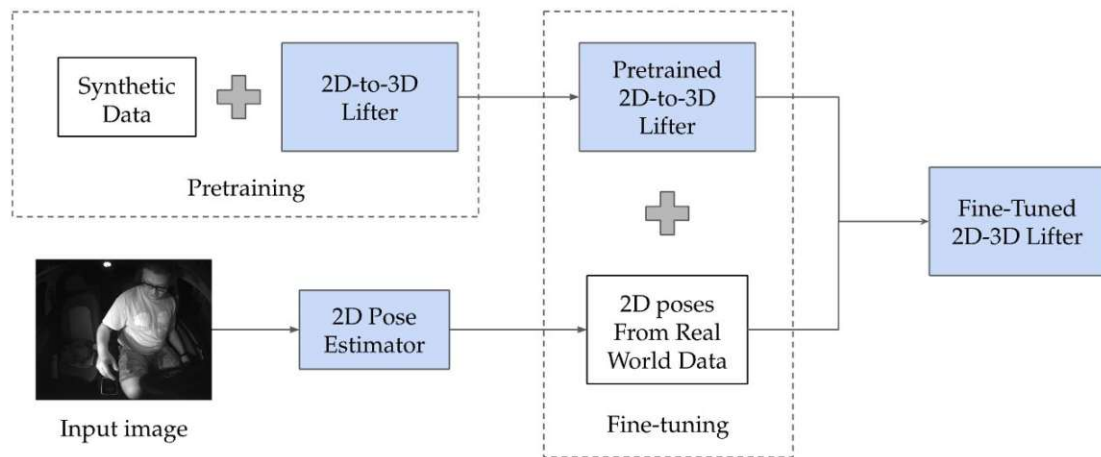


Figure 5.2: Setup of the 3D pose estimation training pipeline with transfer learning.

2D Pose Estimation

The 2D HPE module comprises the Faster R-CNN and the HRNet, both of which were pre-trained on the COCO dataset. We did not fine-tune these models further, as in Chapter 4, because they were already trained on real images and performed sufficiently well on Drive&Act. Moreover, Drive&Act does not provide its bounding box or 2D pose annotations to the public¹.

First, we prepared individual frame images from the Drive&Act videos. We extracted frames at a sampling rate of 1 Hz, assigning an identifier number to each frame and

¹<https://driveandact.com/>, accessed on 2023-05-03.

recording the corresponding subject. This metadata was then formatted into the COCO dataset JSON structure, along with the annotated upper-body 3D pose keypoints.

Next, we fed the frames into the pre-trained human detector, Faster R-CNN, to obtain the bounding box coordinates of the driver. Because Drive&Act contained at most one person—the driver—in each scene, we simply retained the bounding box with the highest confidence score when multiple boxes were predicted. Frames with no detected driver were discarded. The output was stored in COCO-style JSON files.

In the second stage of the pipeline, we provided the video frames and their bounding boxes as input to HRNet. The 2D HPE model produced 17 keypoints per frame, of which we retained only the 13 upper-body keypoints, as in the previous chapter. This output was also saved in JSON format, consistent with the bounding box files. Using the frame identifier numbers, the estimated 2D poses and annotated 3D poses could then be easily matched.

Pre-training of 2D-to-3D Pose Lifters

We used the SyntheticCabin IR 1M dataset for pre-training. This dataset offers a larger number of samples and a greater variety of views than the original SyntheticCabin IR. Recall that the improved synthetic dataset, SyntheticCabin IR 1M, contains images captured from ten camera views. However, there is still no exact match for the views in Drive&Act. As one might expect, installing a camera in a car allows virtually unlimited configurations; even a slight tilt or shift in position can result in different camera parameters.

Our approach was to use images from a set of specific views that were most similar to the target view. Table 5.1 lists all camera positions used to pre-train the models for each Drive&Act view, and Figure 5.3 shows example images of those views. In addition, we pre-trained another version of each model using the full SyntheticCabin IR 1M dataset, referred to as *All_Views*. We subsequently used these models to evaluate whether training with a larger but more generic dataset would lead to improved performance.

For pre-training, each 2D-to-3D pose lifter was trained using the annotated 2D keypoints of each synthetic image as input, with the corresponding 3D keypoints serving as ground truth for loss computation. The training procedure followed that described in Section 4.3.2.

5.3.2 Fine-tuning of 2D-to-3D Pose Lifters

After extracting the 2D poses, we proceeded to fine-tune the 2D-to-3D pose lifters using the 2D poses and the annotated 3D keypoints. A caveat of the annotated 3D poses from Drive&Act is that they are derived from triangulation. As we know, triangulation requires visibility from at least two different views, which is not always the case for every joint at every frame. Self-occlusion is the primary source of this issue. One can easily imagine a situation in which a driver twists their body to reach for an item in the back;

Target Drive&Act Views	Selected SyntheticCabin IR 1M Views
<i>a_column_co_driver</i>	<i>A_Pillar_Codriver, Front_Left, Front_TopLeft, Rear_Mirror</i>
<i>a_column_driver</i>	<i>A_Pillar_Driver, Front_Right, Front_TopRight</i>
<i>center_mirror</i>	<i>OMS_01, Dashboard, Front</i>
<i>a_column_co_driver,</i> <i>a_column_driver,</i> and <i>center_mirror</i>	<i>All Views</i> (<i>A_Pillar_Codriver, A_Pillar_Driver, Front_Left,</i> <i>Front_TopLeft, Rear_Mirror, Front_Right,</i> <i>Front_TopRight, OMS_01, Dashboard, Front</i>)

Table 5.1: List of camera positions in SyntheticCabin IR-1M targeting each Drive&Act view. The last row shows an additional setup used to pre-train models on the full synthetic dataset.

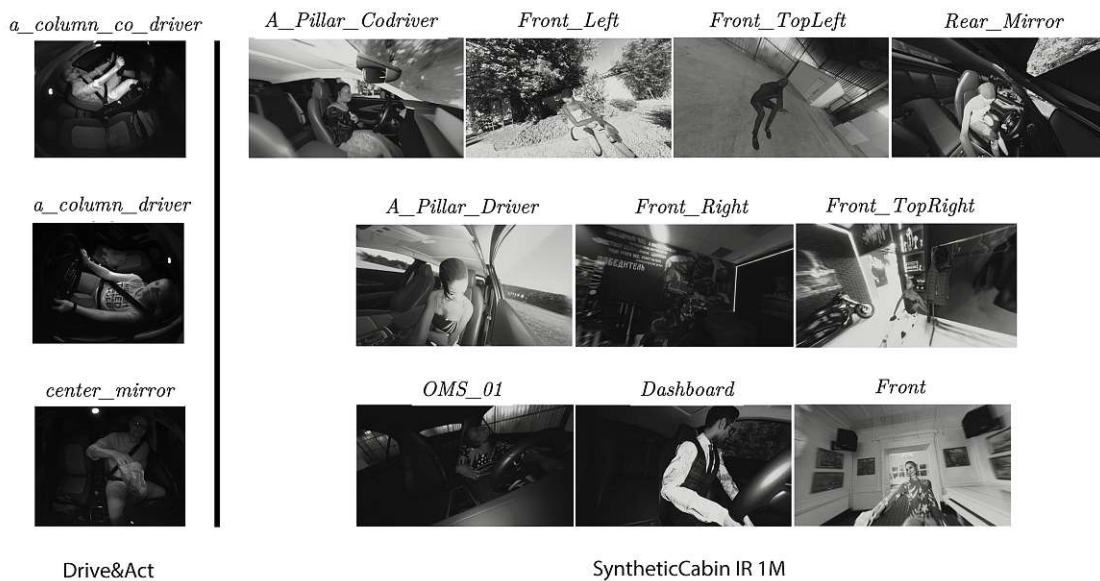


Figure 5.3: Example images for the camera positions of SyntheticCabin IR 1M targeting each Drive&Act view.

one side of the body may obscure the other from the camera’s view. Consequently, those joints cannot be annotated and are left empty in the dataset.

To handle the missing annotations, we performed the forward pass as usual, but when computing the loss for backpropagation, we included only the joints with valid annotations and masked out the loss contributions from the missing ones. The remaining training process followed that described in Section 4.3.2.

5.4 Results

In this section, we present the results in two formats. The tables show the test set MPJPE values for each Drive&Act view, along with their standard errors, whereas the figures compare the test results between the from-scratch and pre-trained 2D-to-3D pose lifters, highlighting the improvements achieved through pre-training. The key findings are summarized in this chapter, while the complete experimental results across all configurations are presented in Appendix A.

Table 5.2, Table 5.3, and Table 5.4 present the test results from the GraFormer model trained from scratch, fine-tuned with selected views, and fine-tuned with *All_Views*, respectively. Each row reports the MPJPE value and its standard error for each amount of Drive&Act training data, ranging from 5% to 50% of a subject, as well as the full data of 1, 2, 4, and 8 subjects, corresponding to 100%, 200%, 400%, and 800%, respectively. The values represent the concatenated results of Algorithm 5.1 and Algorithm 5.2. We highlighted the model with the lowest MPJPE for each data level in bold, and the overall lowest result in each table is further underlined.

Similarly, Figure 5.4 illustrates the overall MPJPE trends, comparing the results of the from-scratch and pre-trained models. In each row, the figure on the left visualizes the results summarized in the tables. The vertical axis represents the MPJPE, while the horizontal axis, shown on a logarithmic scale, indicates the amount of training data, ranging from 5% to 50% of a subject and up to the full datasets of 1, 2, 4, and 8 subjects. The error trends are plotted with dashed lines for the from-scratch models and solid lines for the pre-trained models. The y-axis range is restricted to 0-150 mm, with data points beyond this range excluded for clarity.

The figures in the right column illustrate the gains from pre-training by plotting the MPJPE improvement, calculated as the difference between the errors of the from-scratch and pre-trained models. Positive values therefore indicate an error reduction. The y-axis in these plots is truncated at a lower bound of -5 mm. Standard errors are represented by error bars.

5.5 Discussion

The main question addressed in this experiment was whether there is clear evidence supporting the benefit of pre-training. Looking at Figure 5.4, which shows the results

5. TRANSFER LEARNING FROM SYNTHETIC DATA

Drive&Act Test Set Result: <i>center_mirror</i> view								
No pre-training, Average MPJPE[mm]								
Model	Amount of Training data							
	5% of a subject	10% of a subject	25% of a subject	50% of a subject	single subject	two subjects	four subjects	all subjects
SimpleBL	404.3± 6.6	477.8±16.8	251.8±15.6	117.3± 7.5	75.2± 2.4	62.5± 1.8	55.2± 1.4	48.6
SemGCN	115.9± 5.3	93.9± 5.2	82.2± 4.2	81.1± 5.1	79.1± 4.5	69.3± 3.4	60.6± 2.2	54.1
GraphMLP	82.4± 2.9	77.2± 2.7	73.3± 3.7	69.7± 4.7	65.7± 5.0	60.8± 3.1	57.0± 2.1	53.7
GraFormer	90.0± 3.0	78.3± 3.0	72.9± 3.7	72.0± 4.2	66.6± 4.0	58.5± 1.3	57.1± 2.1	48.1
JointFormer	109.9± 3.4	105.3± 2.4	99.1± 2.6	91.7± 2.9	85.4± 3.1	65.9± 1.8	54.2± 1.7	50.2

Table 5.2: Test results of models without pre-training in *center_mirror* view. Bold values denote the best model for each quantity of training data.

Drive&Act Test Set Result: <i>center_mirror</i> view								
Pre-trained with selected views, Average MPJPE[mm]								
Model	Amount of Training data							
	5% of a subject	10% of a subject	25% of a subject	50% of a subject	single subject	two subjects	four subjects	all subjects
SimpleBL	88.0± 3.8	76.7± 3.3	72.6± 4.1	72.9± 5.1	73.4± 6.3	66.7± 4.9	52.9± 1.3	46.0
SemGCN	83.5± 4.9	74.8± 5.0	69.7± 4.6	69.2± 4.3	67.6± 4.4	58.4± 2.1	54.2± 1.9	48.6
GraphMLP	67.4± 3.8	69.6± 4.6	64.9± 4.2	61.9± 4.2	61.0± 3.7	55.3± 2.3	57.9± 2.8	57.2
GraFormer	70.9± 4.1	65.1± 4.5	66.2± 6.0	59.2± 4.1	59.2± 4.2	54.9± 2.1	52.8± 1.9	48.1
JointFormer	69.6± 3.3	66.5± 4.1	61.3± 3.6	66.3± 5.3	66.0± 6.5	55.1± 3.4	50.0± 1.3	46.0

Table 5.3: Test results of selected-view pre-trained models in *center_mirror* view.

Drive&Act Test Set Result: <i>center_mirror</i> view								
Pre-trained with <i>All_Views</i> , Average MPJPE[mm]								
Model	Amount of Training data							
	5% of a subject	10% of a subject	25% of a subject	50% of a subject	single subject	two subjects	four subjects	all subjects
SimpleBL	85.6± 2.6	79.4± 5.7	71.0± 3.8	69.8± 3.7	71.9± 5.7	63.9± 3.6	51.9± 1.0	48.6
SemGCN	77.1± 4.1	75.3± 5.8	73.2± 5.2	69.1± 4.5	72.9± 8.1	65.2± 4.4	54.9± 1.3	51.9
GraphMLP	67.5± 3.5	69.9± 4.7	66.2± 4.9	61.7± 4.2	61.7± 4.8	56.8± 2.6	55.5± 2.5	52.8
GraFormer	68.6± 3.5	62.3± 3.3	61.0± 3.7	60.9± 3.4	58.6± 3.7	54.2± 2.2	51.4± 2.0	49.9
JointFormer	70.3± 4.9	64.5± 5.3	62.0± 4.6	66.5± 8.7	60.6± 5.6	57.7± 3.5	48.2± 1.2	45.4

Table 5.4: Test results of *All_Views* pre-trained models in *center_mirror* view.

from GraFormer pre-trained with selected views and *All_Views* (Table 5.1), we can clearly see that the lines representing the pre-trained models consistently lie below those of the from-scratch models. This strongly suggests the advantage of pre-training. However, the benefits diminish as more real-world data are included. Visually, these two lines gradually converge in the left-column plots. The from-scratch models improve more rapidly and usually approach the pre-trained line when four or eight subjects are used, with some exceptional cases where the from-scratch line even drops below the pre-trained line. These patterns are consistently observed across all configurations shown in Appendix A.

When comparing models trained with selected views and *All_Views*, we find no consistent improvement from using the full data during pre-training. For instance, the overall smallest MPJPE of 45.4 mm was achieved by the JointFormer fine-tuned with all subjects from the *center_mirror* view in Table 5.4, which was only slightly lower than the

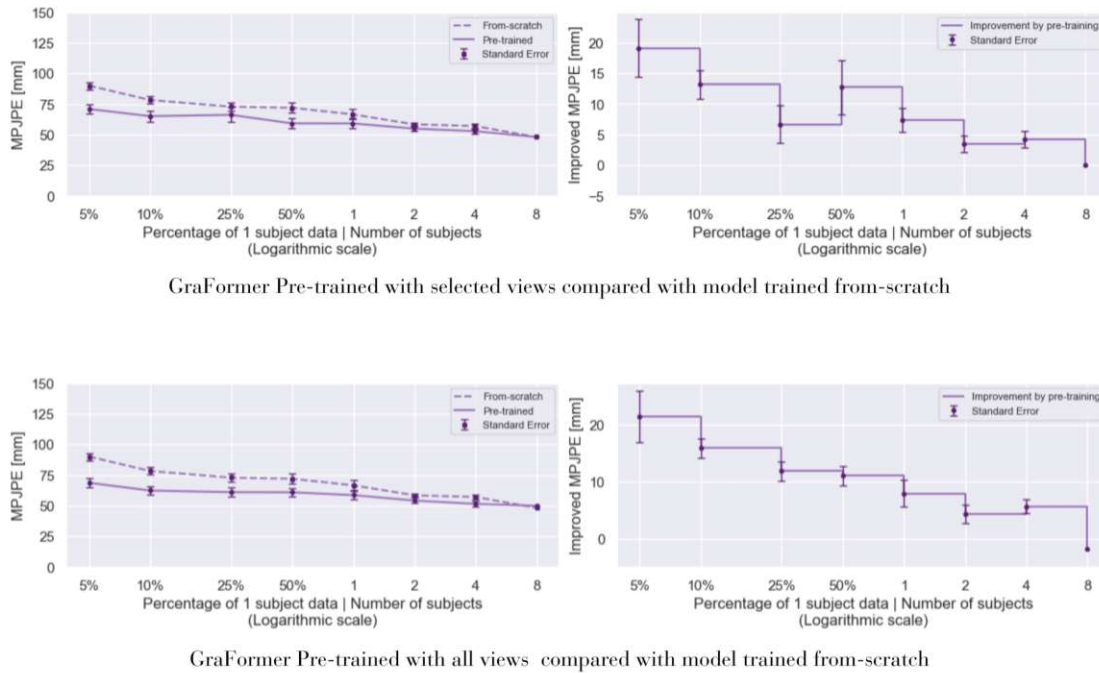


Figure 5.4: Evaluation results for GraFormer on the testing set of Drive&Act (*center_mirror*).

MPJPE of 46.0 mm obtained by the models in Table 5.3. However, the *All_Views*-pre-trained JointFormer performed worse (57.7 mm) than the selected-view one (55.1 mm) when two subjects were used. Therefore, pre-training with selected views appears generally preferable, as it avoids additional computational overhead without sacrificing performance. Nevertheless, having a generic model pre-trained with *All_Views* could still be advantageous when fine-tuning for multiple views.

In terms of camera views, the results indicate that models trained with images from the *center_mirror* view produced the smallest MPJPE, followed by the *a_column_co_driver* and *a_column_driver* views, respectively. For the *center_mirror* view, the JointFormer achieved the best performance at 45.4 mm for the *All_Views*-pre-trained model and 46.0 mm for the selected-view model. In the *a_column_co_driver* view, the GraFormer produced the lowest errors of 53.9 mm and 54.0 mm for the *All_Views* and selected-view models, respectively. Lastly, for the *a_column_driver* view, the best *All_Views* model was again the GraFormer (61.9 mm), while the best selected-view model was the JointFormer (61.6 mm).

We can also see from the tables that each model performs differently depending on the amount of fine-tuning data. When using only a small subset of a subject (less than 50%), GraphMLP often delivered the best or second-best results. In fact, it achieved the best performance in eight out of nine cases when only 5% of the data from a subject

were used. As the amount of training data increased to four subjects, GraFormer more frequently produced lower errors. Both GraphMLP and GraFormer are hybrid models that combine GNNs with other deep learning architectures, which may explain their strong performance. GNNs capture local dependencies, while MLP and transformer layers model global relationships. However, when all data were used, JointFormer yielded the best overall results.

In conclusion, the results clearly demonstrate the benefit of using synthetic data to pre-train 2D-to-3D pose lifters. The improvement was consistently observed across architectures, although the magnitude varied. We found that using synthetic data from *All_Views* did not provide a noticeable advantage over using selected views. The lowest MPJPE values were obtained from models trained with data from the *center_mirror* view. GraphMLP performed best when fine-tuned with smaller training samples, while GraFormer often achieved the lowest errors when more data were available. Finally, when trained on the full Drive&Act dataset, JointFormer produced the overall best performance.

Leveraging Synthetic Data for Unlabeled Datasets

In this chapter, we considered the scenario in which 3D pose annotations for real-world datasets are unavailable. This is often the case, as annotating 3D poses requires multi-view imaging, making it impractical to label individual images captured in the wild. It would be interesting to explore how effectively models can be trained without using any 3D annotations from the target dataset.

In the following sections, we describe two experimental strategies and report their results on the Drive&Act dataset (Section 2.4.2). Section 6.1 outlines the overall objectives of the study, while Section 6.2 details the experimental setup common to both strategies. Each method is then presented in Section 6.3 and Section 6.4, respectively, together with its specific setup, results, and discussion. Finally, an overall discussion is provided in Section 6.5.

6.1 Objective & Approach

In this final experiment, we aimed to answer the last research question concerning the usefulness of synthetic data for estimating 3D poses when 3D annotations of the target dataset were unavailable. Unlike 2D poses, which can be annotated manually by marking the location of each joint in any given image, 3D pose annotation is considerably more challenging. There are multiple obstacles to obtaining 3D pose annotations. The most common method involves using triangulation on multiple images captured with a calibrated multi-camera setup. The calibration must be precise in both camera parameters and synchronization between cameras, as even slight errors in these factors can result in substantial inaccuracies. After acquiring the images, accurate 2D annotation remains

a labor-intensive task. The use of complex equipment setups also presents difficulties when capturing movements outside controlled studio environments. Occlusion can further prevent certain keypoints from being visible from multiple viewpoints. Because of all these factors, 3D pose datasets are scarce, making it often impossible to obtain 3D annotations for custom datasets. Therefore, it is worthwhile to investigate whether models can be trained by leveraging synthetic data for target domains lacking 3D labels.

We explored two approaches. The first was to perform inference directly using the pre-trained model from Chapter 5. This approach demonstrates how effectively models trained solely on synthetic data can generalize to real-world data. The second approach involved training a 2D-to-3D pose lifter using only the 2D poses from the target dataset, together with paired 2D–3D pose examples from synthetic data. This approach corresponds to a semi-supervised learning setup, in which a model is trained on an unlabeled dataset with assistance from another labeled dataset.

For this experiment, we assumed that the Drive&Act dataset (Section 5.2.2) served as our target dataset, where 3D pose annotations were unavailable and only the images were accessible. We applied both approaches using SyntheticCabin IR 1M (Section 5.2.1) as the synthetic dataset. We then compared the results of these approaches with those of models trained with access to 3D pose annotations. This comparison illustrates the potential performance gap that could be reduced through improved synthetic data generation or enhanced learning strategies.

6.2 General Setup

For this experiment, we had full access to the synthetic data from SyntheticCabin IR 1M but only to the images from the Drive&Act dataset. We assumed that the annotated 3D poses of Drive&Act were unavailable and used them solely for evaluation. Subject number 10 was reserved as the validation set, and subjects 11, 12, 13, and 14 were used as the test set. The remaining subjects were used for training. Similar to the previous chapter (Table 5.1), we employed both the selected views from SyntheticCabin IR 1M corresponding to each target view of Drive&Act and the *All Views* configuration. This setup allowed us to investigate whether matching viewing angles yields better performance or whether using a larger, more diverse dataset is preferable.

Because the models were not trained with annotated 3D poses from the target dataset in this setup, the lifters could not infer the scale and orientation of the poses in the same coordinate system as the annotated 3D poses of Drive&Act. Therefore, MPJPE is not an appropriate measure of model accuracy. Instead, we used an alternative version of the metric, PA-MPJPE. To compute PA-MPJPE, we first performed Procrustes alignment to match the scale and rotation of the predicted 3D poses to the annotated ones before calculating MPJPE. This metric allows us to assess similarity based solely on the shape of the poses. The alignment was implemented using the code provided by Zhao et al. in

their SemGCN repository¹ [ZPT⁺19].

We tested the pre-trained models using each set of selected views from SyntheticCabin IR 1M described in Table 5.1 on the test set of the corresponding target view from Drive&Act. In addition, we evaluated the models trained with *All_Views* on all three views of Drive&Act.

6.3 Approach 1: Synthetic-to-real Generalization

The first approach is straightforward. We applied the same pose estimation pipeline from the previous chapter but used only the pre-trained 2D-to-3D pose lifter on the target dataset without fine-tuning. This experiment essentially tested the model’s ability to generalize from synthetic to real-world data. The closer the synthetic data distribution is to real-world data, the better the estimation performance is expected to be.

6.3.1 Experiment Setup

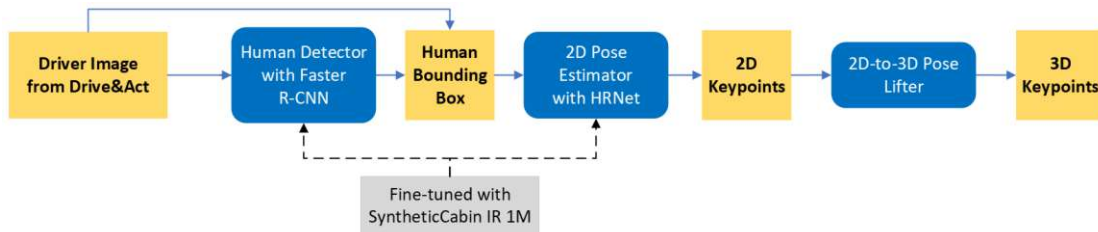


Figure 6.1: Pose estimation pipeline for Approach 1. The 2D-to-3D pose lifter was pre-trained with SyntheticCabin IR 1M but was not fine-tuned with real-world data, unlike in the previous chapter.

In this approach, we investigated how well the models generalized from the domain of synthetic data to the real world. We directly used the pre-trained models from Section 5.3.1. The pipeline is illustrated in Figure 6.1. Specifically, the 2D-to-3D pose lifters were pre-trained with SyntheticCabin IR 1M without any fine-tuning on Drive&Act. As in the previous chapter (Table 5.1), the models were trained with selected views corresponding to each target view in the Drive&Act dataset, and an additional model was trained using *All_Views*. We obtained four pre-trained lifters for each architecture.

The 2D HPE stages were performed in the same manner as in Chapter 5. We used a pre-trained human detector from MMDet and a pre-trained 2D pose estimator from MMPose. These two models processed images from Drive&Act to extract 2D poses, which were then passed to the pre-trained lifters to estimate 3D poses.

For comparison, we trained a reference model using the 2D and 3D poses from the Drive&Act dataset. This model served as a baseline for assessing how models trained

¹<https://github.com/garyzhao/SemGCN>, accessed on 2023-08-09.

solely on synthetic data differed from those trained directly on the target dataset. The performance gap observed between the reference model and the synthetic-data models indicates the potential for improvement achievable through better synthetic datasets or transfer strategies.

6.3.2 Results & Discussion

Table 6.1 and Table 6.2 present the results for models trained with selected views and *All_Views*, respectively. For comparison, Table 6.3 shows the results for models trained directly on Drive&Act. These tables display the Drive&Act test set PA-MPJPE for each model across different views. The numbers in bold indicate the lowest PA-MPJPE in each view, with the overall best result in each setting further underlined for emphasis.

Table 6.2 shows that the PA-MPJPE values were lowest for the *center_mirror* view, followed by the *a_column_co_driver* and *a_column_driver* views, respectively. We also observed that the selected-view pre-trained models generally achieved lower PA-MPJPE values than the *All_Views* pre-trained models. As mentioned previously, the selected-view pre-trained SemGCN achieved the best performance, with a PA-MPJPE of 53.78 mm. Among the *All_Views* pre-trained models, GraFormer obtained the lowest error at 59.95 mm for the *a_column_co_driver* view.

Table 6.3 presents the PA-MPJPE results for models trained exclusively on Drive&Act data. The best performance was achieved by SemGCN and GraFormer in the *center_mirror* view, both producing a PA-MPJPE of 28.9 mm—approximately half of our best zero-shot result. This difference highlights the substantial room for improvement in the realism and diversity of synthetic data generation for future studies.

The top two rows of Figure 6.3 show example outputs from GraFormer and SemGCN trained with selected views for the target view, *center_mirror*. The green skeletal figures represent the ground-truth poses, while the black figures show the poses inferred by the models. Each row presents three views of the same pose to provide a better visual perspective.

6.4 Approach 2: Semi-Supervised Learning

In the second approach, we utilized additional information from the target dataset. Specifically, we employed a 2D-to-3D pose lifter called RepNet, as described in Section 3.3.6. Unlike the supervised training approach used for the lifters in previous chapters, we trained this model without assuming a direct correspondence between 2D and 3D poses. Rather than comparing the predicted 3D poses with the ground truth, the GAN-based training process relied on a critic network and a 2D reprojection of the predicted poses to update the lifter.

In contrast to the original RepNet, which trains the model using a single dataset, we extended the training to include 2D poses from the real-world dataset as well. This

Drive&Act Test Set Result			
Pre-trained with selected views			
Model	<i>a_column_co_driver</i> PA-MPJPE[mm]	<i>a_column_driver</i> PA-MPJPE[mm]	<i>center_mirror</i> PA-MPJPE[mm]
SimpleBL	117.89	122.85	59.08
SemGCN	65.11	108.06	53.78
GraphMLP	69.57	92.14	60.58
GraFormer	65.62	90.33	57.31
JointFormer	83.09	105.87	56.39

Table 6.1: Test set results for models, each trained with selected views of SyntheticCabin IR 1M corresponding to each target view of the Drive&Act dataset.

Drive&Act Test Set Result			
Pre-trained with <i>All_views</i>			
Model	<i>a_column_co_driver</i> PA-MPJPE[mm]	<i>a_column_driver</i> PA-MPJPE[mm]	<i>center_mirror</i> PA-MPJPE[mm]
SimpleBL	85.13	136.38	60.47
SemGCN	70.25	92.73	59.95
GraphMLP	69.97	108.75	60.17
GraFormer	59.46	77.66	65.18
JointFormer	84.02	110.83	62.52

Table 6.2: Test set result for the model trained with *All_views* of SyntheticCabin IR 1M for each target view of Drive&Act.

Drive&Act Test Set Result			
Models trained with Drive&Act			
Model	<i>a_column_co_driver</i> PA-MPJPE[mm]	<i>a_column_driver</i> PA-MPJPE[mm]	<i>center_mirror</i> PA-MPJPE[mm]
SimpleBL	41.3	41.3	29.4
SemGCN	39.4	40.6	28.9
GraphMLP	37.5	45.1	30.3
GraFormer	37.8	40.4	28.9
JointFormer	35.2	40.3	31.9

Table 6.3: Test set results for models trained on each target view of Drive&Act, provided for comparison.

setup effectively bridges the model between synthetic and real domains. The use of fully labeled synthetic data to leverage unlabeled real-world data constitutes a semi-supervised learning framework.

6.4.1 Experiment Setup

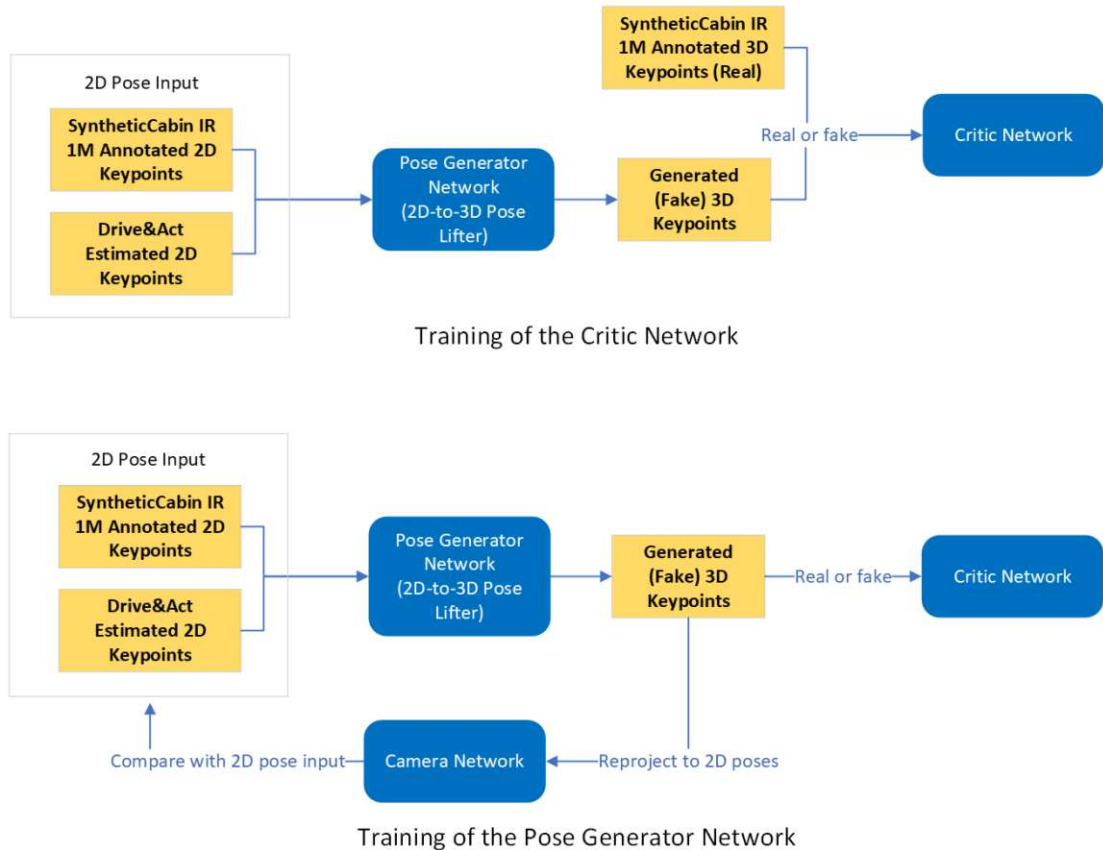


Figure 6.2: GAN training of RepNet was conducted by interleaving epochs of training the critic network and the pose generator network.

The GAN training process of RepNet comprised three models: the critic, the camera network, and the pose generator network (the 2D-to-3D pose lifter). The lifter learned to generate 3D poses from the given 2D poses by optimizing two losses: the critic loss and the reprojection loss, which was calculated by reprojecting the generated 3D poses to 2D using the camera network and comparing them with the input poses. The training required two data inputs: 2D and 3D poses. These pairs did not need to correspond or even originate from the same dataset; however, they had to be realistic, as the critic learned to distinguish real 3D poses from generated ones.

Naturally, we used the synthetic data for both 2D and 3D poses. Similar to the first

approach, we trained the lifter on the synthetic data and evaluated its synthetic-to-real generalization on Drive&Act. However, we also included 2D poses from Drive&Act during training. Incorporating 2D poses from the target dataset helped the model adapt to the distribution of poses commonly observed in real-world driving scenarios.

For the implementation of RepNet, we reimplemented the model in PyTorch, following the original authors' implementation². For each target Drive&Act view, the model was trained using the corresponding selected view from SyntheticCabin IR 1M, as described in Section 5.3.1. In addition, we provided the estimated 2D poses from Drive&Act of the same view.

During GAN training, we interleaved one epoch of training the pose generator network with one epoch of training the critic network. To train the generator, we fed 2D poses from either SyntheticCabin IR 1M or Drive&Act and computed the total loss as the sum of the critic score, camera network loss, and 2D reprojection loss. For the critic network, we used batches containing real 3D poses from SyntheticCabin IR 1M annotations and fake 3D poses generated by the lifter. The loss for the critic was computed as the adversarial loss with a gradient penalty term [GAA⁺17].

During validation, we used only the pose generator network (the 2D-to-3D pose lifter). The generator was fed with 2D poses, and the PA-MPJPE was computed against the corresponding 3D poses from SyntheticCabin IR 1M. For comparison, we also trained a reference RepNet using both 2D and 3D poses exclusively from Drive&Act. This model represents the scenario where RepNet is applied directly to the Drive&Act dataset. Note that the pose annotations in Drive&Act are incomplete, as occluded keypoints may be missing from the annotation.

6.4.2 Results & Discussion

Table 6.4 presents the test set results obtained from Drive&Act using RepNet. Similar to the validation phase, we evaluated the model's performance based on the 3D poses generated by the pose generator. The top row shows the results when RepNet was trained on SyntheticCabin IR 1M and applied using the predicted 2D poses from the Drive&Act dataset, while the bottom row shows the results when both the 2D and 3D poses from Drive&Act were used directly, without involving any synthetic data.

For this second approach, we observed that the RepNet model trained with synthetic data outperformed the model trained directly on Drive&Act, despite the latter using annotated 3D poses. The best result for the former setup was achieved in the *a_column_co_driver* view, with a PA-MPJPE of 106.36 mm—lower than the best result from the latter setup, which was 154.84 mm for the same view.

There are several possible explanations for why the model trained with synthetic data achieved better performance. First, the 3D annotations in the synthetic dataset were complete and consistent, providing both 2D and 3D coordinates for every keypoint. In

²<https://github.com/bastianwandt/RepNet> , accessed on 2023-11-07.

contrast, the Drive&Act dataset contained occluded or missing keypoints that were not annotated. Second, the synthetic dataset was substantially larger, which likely allowed the model to learn more robust and generalizable representations. Lastly, the cleaner and noise-free nature of synthetic data may have contributed to more stable training dynamics compared to real-world data.

The last row of Figure 6.3 shows example outputs from RepNet trained on SyntheticCabin IR 1M using predicted 2D poses from Drive&Act.

Drive&Act Test Set Result			
RepNet			
Training Dataset	<i>a_column_co_driver</i> PA-MPJPE[mm]	<i>a_column_driver</i> PA-MPJPE[mm]	<i>center_mirror</i> PA-MPJPE[mm]
SyntheticCabin IR 1M with predicted 2D Poses from Drive&Act	<u>106.36</u>	203.95	130.29
Predicted 2D Poses and annotated 3D poses from Drive&Act	154.84	235.11	204.25

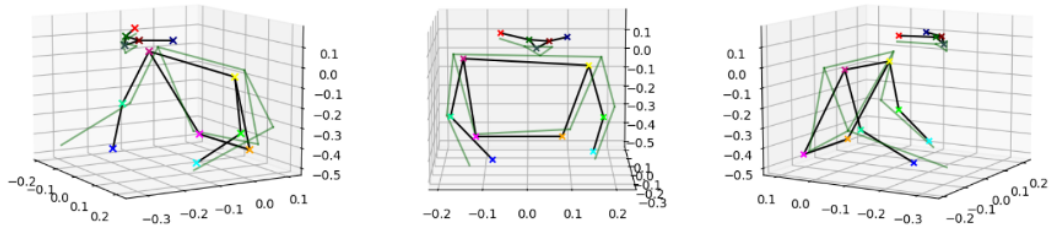
Table 6.4: Evaluation result on the test set for RepNet. The top row was trained with Synthetic IR 1M and predicted 2D poses from Drive&Act. For comparison, the second row used predicted 2D poses and annotated 3D poses of Drive&Act.

6.5 Discussion

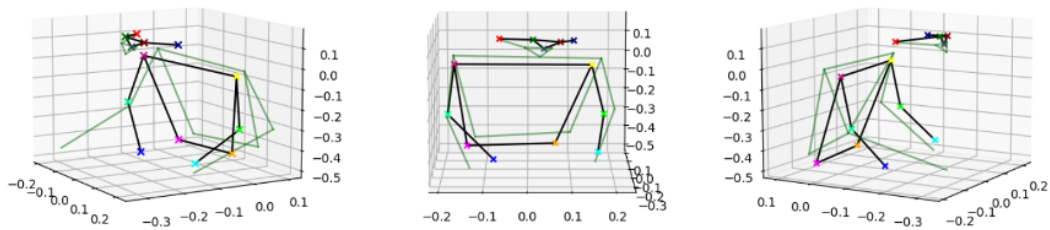
The results clearly favored Approach 1, in which we used pre-trained models for direct inference. The PA-MPJPE values from this approach were consistently and significantly lower than those obtained from Approach 2 across all corresponding views. The lowest PA-MPJPE achieved by the first approach was 53.78 mm in the *center_mirror* view, obtained by the SemGCN model trained with the selected views of SyntheticCabin IR 1M (Table 6.1). In contrast, the best result from the second approach was 106.36 mm, produced by the RepNet model in the *a_column_co_driver* view.

The supervised training used in the first approach, which relied on explicitly paired input-label data, proved to be more effective than the semi-supervised learning strategy employed by RepNet. Although predicted 2D poses from the Drive&Act dataset were included as additional input during the training of the second approach, they were insufficient to close the performance gap.

Approach 1: GraFormer pre-trained with Dashboard, Front, and OMS_01 view in SyntheticCabin IR 1M (PA-MPJPE=48.2 mm)



Approach 1: SemGCN pre-trained with Dashboard, Front, and OMS_01 view in SyntheticCabin IR 1M (PA-MPJPE=58.8 mm)



Approach 2: RepNet trained with SyntheticCabin IR 1M & predicted 2D poses from Drive&Act (PA-MPJPE=90 mm)

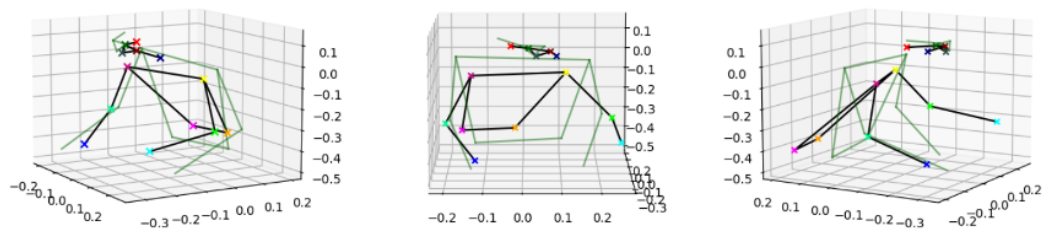


Figure 6.3: Example outputs for each model. The green skeletal figures are the ground truth poses. The black figures are the inferred poses from the models.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Conclusion and Future Works

7.1 Conclusion

In summary, this thesis explored the benefits of using synthetic data for 3D HPE. Specifically, we investigated whether synthetic data could effectively train 2D-to-3D pose lifters to achieve accuracy comparable to traditional methods. We hypothesized that it could help reduce the amount of real-world data required for training deep learning models. We then narrowed the scope of our study to focus on the final stage of a multi-stage monocular 3D HPE pipeline—the 2D-to-3D pose lifter. Six different models were employed to investigate the robustness and generalizability of this approach. First, to address our research question regarding the extent to which 3D poses can be estimated from monocular synthetic data, we showed that the method performed well by comparing the test accuracy in MPJPE between triangulation and the 2D-to-3D pose lifters on the SyntheticCabin IR dataset. The predictions from the lifters were within approximately 10 mm of those obtained via triangulation. It is worth noting that triangulation relied on ground-truth camera parameters, which are often difficult to obtain in real-world settings.

Next, to address our second research question regarding the extent to which knowledge of 3D HPE can be transferred from synthetic to real-world data, we investigated whether models pre-trained with synthetic data could alleviate the need for costly real-world data. We conducted an experiment comparing models trained from scratch with those pre-trained on synthetic data. The SyntheticCabin IR 1M dataset was used for pre-training, and the Drive&Act dataset for fine-tuning. We incrementally increased the amount of real-world data used for training and compared the test results of both approaches.

The empirical evidence clearly demonstrated the benefit of pre-training: pre-trained models consistently achieved better results compared to from-scratch models when trained with the same amount of real-world data. Hybrid models such as GraphMLP and GraFormer tended to perform particularly well when real-world data were limited, while transformer-based models such as JointFormer outperformed the others only when sufficient fine-tuning data were available. As an example, for *center_mirror* view, GraphMLP pre-trained with selected views and fine-tuned with 5% of a subject’s data produced the best test results MPJPE at 67.4 mm, compared to the from-scratch version which achieved 82.4 mm. Similarly, GraFormer pre-trained with selected views and fine-tuned with 50% of a subject’s data achieved an MPJPE of 59.2 mm, compared to 72.0 mm for the from-scratch version. However, the benefit diminished as more real-world data became available, and in some cases, the pre-trained models showed no visible improvement. For instance, GraFormer pre-trained with selected views and fine-tuned on the full Drive&Act dataset achieved the same MPJPE as the model trained from scratch, at 48.1 mm. Thus, we conclude that synthetic data are most beneficial when access to large-scale real-world datasets is limited, which is consistent with the observations reported by [SSN⁺23] in the context of 2D HPE.

Finally, to address our last research question, we examined the extent to which 3D poses can be estimated by leveraging synthetic data when 3D pose annotations for real-world data are unavailable. Two approaches were evaluated: (1) using a model trained solely on synthetic data, and (2) employing a semi-supervised learning strategy. For the latter, RepNet was used as a representative model. This method utilized a generative adversarial network with a 2D-to-3D pose lifter as the generator, which learned to produce accurate 3D poses through a re-projection loss. We trained the model using 2D pose annotations from the target dataset, Drive&Act, and 3D poses from the synthetic dataset, SyntheticCabin IR 1M. The results from the first approach were promising, achieving the best PA-MPJPE of approximately 85 mm, whereas the PA-MPJPE from RepNet was roughly twice as high. We also demonstrated that a substantial accuracy gap remains between models trained exclusively on synthetic data and those trained directly on real-world data. This highlights the ongoing need to develop more realistic synthetic datasets to further close the performance gap.

In conclusion, we have shown that utilizing synthetic data can significantly reduce the reliance on real-world data for 3D HPE in DMS. By leveraging synthetic data in the context of driver pose estimation, researchers can substantially lower the cost of data collection, simulate a broader range of human poses under diverse environmental conditions, and still retain a large portion of model accuracy.

7.2 Future Work

Our experimental results confirmed the benefits of using synthetic data to aid in pre-training 2D-to-3D pose lifters. We have also identified several opportunities to enhance and extend this work, as well as open questions that merit further investigation. This

section highlights promising directions for future research.

7.2.1 Synthetic Pose Data Generation and Augmentation

We have shown that using synthetic pose data improves the learning process of 2D-to-3D pose lifters. However, existing synthetic image datasets tend to emphasize generating visually realistic images of similar poses under varying textures and lighting conditions. As a result, the diversity and quantity of available synthetic pose data remain limited. Expanding both the scale and variation of synthetic poses could therefore yield further benefits.

Future studies could explore more advanced synthetic pose generation and augmentation techniques. Because skeletal poses are simpler to manipulate than full synthetic images, such approaches are computationally feasible and can even be applied dynamically during training. Large numbers of pose samples could be generated on the fly or in real time. Augmentation strategies might include horizontal flipping, pose rotation to simulate different camera viewpoints, or the injection of noise into existing real-world poses. The augmented 3D poses could then be projected into 2D to create additional training pairs for the lifter. These methods could substantially enlarge training datasets and improve model generalization. For instance, [GZF21] introduced a differentiable pose-augmentation module that can be trained jointly with a lifter network—an approach that future work could build upon.

7.2.2 Leveraging Synthetic Data for Driver Action Recognition

While 3D pose estimation has many applications—including computer animation, human–computer interaction, and motion tracking—it can also act as an intermediate representation for higher-level tasks such as action recognition. Accurate 3D poses produced by a complete pose estimation pipeline—comprising a human detector, a 2D pose estimator, and a 2D-to-3D pose lifter—could serve as valuable input for downstream analysis.

Future research could investigate how pre-trained lifters can help reduce the amount of data required for driver action recognition. Because actions are inherently temporal sequences of poses, the current pipeline would need to be extended to model temporal dynamics. This could be achieved by integrating sequence-processing architectures such as temporal convolutional networks or recurrent models, as proposed in [LFV⁺17] and [MAS⁺22]. The extended system could then be evaluated on the Drive&Act dataset, which was originally designed for driver action recognition, to quantify the benefits of incorporating pre-trained lifters into temporal action models.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Experimental Results from Chapter 5

This appendix presents the detailed results from the experiments described in Chapter 5. Refer to Section 5.4 and Section 5.5 for the corresponding analysis and discussion. The tables in Appendix A.1 report the test set MPJPE values for each Drive&Act view, along with their associated standard errors. The figures in Appendix A.2 illustrate the comparisons between 2D-to-3D pose lifters trained with and without pre-training, highlighting the performance improvements achieved through pre-trained networks. Finally, Appendix A.3 provides qualitative examples of 3D pose outputs generated by different GraFormer models trained under varying conditions for the same input pose.

A.1 Detailed Test Results

The tables in this section present the results for each Drive&Act view, organized into sets of three. Each set corresponds to: (1) models trained from scratch, (2) models fine-tuned with selected views, and (3) models fine-tuned with *All Views*. Each row reports the MPJPE value and its standard error for different amounts of Drive&Act training data, ranging from 5% to 50% of a subject, as well as the full datasets of 1, 2, 4, and 8 subjects—corresponding to 100%, 200%, 400%, and 800%, respectively. The reported values represent the combined outputs from Algorithm 5.1 and Algorithm 5.2. Models achieving the lowest MPJPE at each data level are highlighted in bold, while the overall lowest result in each table is further underlined.

Drive&Act view: *a_column_co_driver*

Drive&Act Test Set Result: <i>a_column_co_driver</i> view								
No pre-training, Average MPJPE[mm]								
Model	Amount of Training data							
	5% of a subject	10% of a subject	25% of a subject	50% of a subject	single subject	two subjects	four subjects	all subjects
SimpleBL	390.4±16.4	494.4±13.1	231.9± 9.2	113.5± 7.9	78.1± 4.6	68.7± 2.1	62.4± 1.4	60.6
SemGCN	284.4±51.0	99.5± 4.4	91.6± 5.2	84.0± 4.5	95.7± 9.2	76.2± 3.8	67.6± 2.2	65.2
GraphMLP	89.0± 2.5	85.3± 3.3	79.2± 3.3	77.9± 4.3	71.3± 3.1	66.1± 2.1	66.2± 2.0	60.8
GraFormer	92.9± 4.0	84.6± 3.7	78.1± 4.0	74.9± 4.0	71.4± 3.5	64.3± 1.6	60.9± 1.7	58.9
JointFormer	111.5± 8.7	104.9± 4.5	100.0± 3.6	92.7± 2.5	85.2± 2.6	75.5± 2.5	65.3± 1.8	54.4

Table A.1: Test result of model without pre-training in *a_column_co_driver* view.

Drive&Act Test Set Result: <i>a_column_co_driver</i> view								
Pre-trained with selected views, Average MPJPE[mm]								
Model	Amount of Training data							
	5% of a subject	10% of a subject	25% of a subject	50% of a subject	single subject	two subjects	four subjects	all subjects
SimpleBL	112.0± 3.1	92.1± 3.6	79.8± 3.2	75.8± 3.1	73.6± 3.1	65.6± 1.6	59.8± 1.2	56.7
SemGCN	99.6± 3.9	93.1± 4.9	89.7± 5.7	82.8± 5.2	79.7± 4.3	71.0± 3.5	63.8± 1.4	58.9
GraphMLP	86.2± 3.2	81.4± 4.9	75.1± 4.2	72.7± 3.5	71.5± 4.2	63.8± 1.3	61.9± 1.6	60.0
GraFormer	87.2± 5.1	82.2± 3.4	78.3± 5.2	73.7± 5.4	70.7± 4.1	62.2± 1.6	57.3± 1.2	54.0
JointFormer	95.0± 2.6	79.0± 3.2	75.7± 5.1	71.9± 4.7	70.5± 4.4	62.7± 1.9	59.4± 1.3	57.2

Table A.2: Test results of selected-view pre-trained models in *a_column_co_driver* view.

Drive&Act Test Set Result: <i>a_column_co_driver</i> view								
Pre-trained with <i>All_Views</i> , Average MPJPE[mm]								
Model	Amount of Training data							
	5% of a subject	10% of a subject	25% of a subject	50% of a subject	single subject	two subjects	four subjects	all subjects
SimpleBL	107.2± 2.2	94.0± 3.1	80.3± 2.6	76.1± 2.9	73.9± 2.9	65.9± 1.3	60.3± 1.3	58.5
SemGCN	94.9± 2.9	89.0± 5.3	86.3± 5.7	80.1± 4.6	78.0± 4.6	69.0± 2.7	62.4± 1.3	57.7
GraphMLP	87.1± 3.2	80.5± 5.6	75.1± 5.1	73.0± 4.1	71.2± 4.1	64.4± 1.8	63.5± 1.8	60.6
GraFormer	83.2± 4.2	80.2± 3.0	77.2± 6.4	69.1± 3.9	70.9± 4.2	59.7± 1.2	57.7± 1.3	53.9
JointFormer	91.2± 2.1	81.8± 3.2	76.5± 4.2	71.9± 3.9	71.7± 3.9	62.4± 1.9	58.9± 0.6	54.6

Table A.3: Test results of *All_Views* pre-trained models in *a_column_co_driver* view.

Drive&Act view: *a_column_driver*

Drive&Act Test Set Result: <i>a_column_driver</i> view								
No Pre-training, Average MPJPE[mm]								
Model	Amount of Training data							
	5% of a subject	10% of a subject	25% of a subject	50% of a subject	single subject	two subjects	four subjects	all subjects
SimpleBL	438.5±12.0	503.3±16.8	265.8±14.9	129.6± 5.6	85.6± 2.2	75.2± 1.9	68.8± 1.0	63.6
SemGCN	271.9±55.5	106.9± 4.5	98.9± 5.1	89.7± 4.1	91.0± 4.7	78.1± 2.0	73.3± 1.2	66.1
GraphMLP	99.6± 4.6	91.1± 2.8	89.3± 4.7	83.9± 3.7	78.5± 3.3	74.4± 1.7	69.7± 1.8	69.2
GraFormer	110.3± 6.3	91.2± 2.5	86.5± 3.7	81.5± 4.0	79.9± 3.0	71.6± 1.5	68.8± 1.1	64.1
JointFormer	112.9± 4.9	103.2± 4.0	100.3± 4.1	93.5± 2.7	88.5± 2.0	77.8± 1.8	67.6± 1.2	61.6

Table A.4: Test results of models without pre-training in *a_column_driver* view.

Drive&Act Test Set Result: <i>a_column_driver</i> view								
Pre-trained with selected views, Average MPJPE[mm]								
Model	Amount of Training data							
	5% of a subject	10% of a subject	25% of a subject	50% of a subject	single subject	two subjects	four subjects	all subjects
SimpleBL	110.9± 1.8	96.4± 3.4	88.3± 4.5	82.3± 3.7	80.1± 2.8	71.6± 1.4	67.0± 0.9	62.2
SemGCN	99.4± 3.7	94.7± 3.6	86.1± 2.5	85.2± 3.8	83.2± 4.1	73.5± 1.5	70.2± 1.3	64.7
GraphMLP	91.3± 3.1	87.8± 2.6	83.3± 3.1	79.3± 2.7	79.2± 4.2	71.2± 1.6	70.0± 2.1	67.4
GraFormer	94.5± 3.4	84.4± 3.1	79.2± 3.4	77.1± 4.2	73.5± 2.4	69.9± 2.0	66.6± 1.2	63.2
JointFormer	92.6± 4.3	83.4± 2.4	79.8± 3.3	76.3± 3.3	74.5± 3.0	68.2± 1.4	64.2± 1.1	61.6

Table A.5: Test results of selected-view pre-trained models in *a_column_driver* view.

Drive&Act Test Set Result: <i>a_column_driver</i> view								
Pre-trained with <i>All_Views</i> , Average MPJPE[mm]								
Model	Amount of Training data							
	5% of a subject	10% of a subject	25% of a subject	50% of a subject	single subject	two subjects	four subjects	all subjects
SimpleBL	113.0± 1.7	97.7± 3.0	88.3± 4.6	82.3± 3.4	79.4± 3.1	70.7± 1.4	66.8± 1.0	62.0
SemGCN	112.1± 4.4	104.0± 5.4	93.6± 4.1	91.7± 4.3	85.4± 3.0	77.2± 1.8	71.1± 1.1	63.7
GraphMLP	92.6± 4.1	86.8± 3.3	83.0± 3.8	80.0± 4.0	78.0± 3.6	71.3± 1.8	69.1± 1.7	66.4
GraFormer	98.7± 3.1	88.3± 2.9	79.4± 3.3	76.5± 3.2	74.2± 2.8	69.0± 1.6	65.1± 1.1	61.9
JointFormer	101.0± 3.7	86.9± 3.2	81.0± 4.3	76.5± 2.8	74.8± 2.7	68.6± 1.4	66.7± 1.4	63.0

Table A.6: Test results of *All_Views* pre-trained models in *a_column_driver* view.

Drive&Act view: *center_mirror*

Drive&Act Test Set Result: <i>center_mirror</i> view								
No pre-training, Average MPJPE[mm]								
Model	Amount of Training data							
	5% of a subject	10% of a subject	25% of a subject	50% of a subject	single subject	two subjects	four subjects	all subjects
SimpleBL	404.3± 6.6	477.8±16.8	251.8±15.6	117.3± 7.5	75.2± 2.4	62.5± 1.8	55.2± 1.4	48.6
SemGCN	115.9± 5.3	93.9± 5.2	82.2± 4.2	81.1± 5.1	79.1± 4.5	69.3± 3.4	60.6± 2.2	54.1
GraphMLP	82.4± 2.9	77.2± 2.7	73.3± 3.7	69.7± 4.7	65.7± 5.0	60.8± 3.1	57.0± 2.1	53.7
GraFormer	90.0± 3.0	78.3± 3.0	72.9± 3.7	72.0± 4.2	66.6± 4.0	58.5± 1.3	57.1± 2.1	48.1
JointFormer	109.9± 3.4	105.3± 2.4	99.1± 2.6	91.7± 2.9	85.4± 3.1	65.9± 1.8	54.2± 1.7	50.2

Table A.7: Test results of models without pre-training in *center_mirror* view.

Drive&Act Test Set Result: <i>center_mirror</i> view								
Pre-trained with selected views, Average MPJPE[mm]								
Model	Amount of Training data							
	5% of a subject	10% of a subject	25% of a subject	50% of a subject	single subject	two subjects	four subjects	all subjects
SimpleBL	88.0± 3.8	76.7± 3.3	72.6± 4.1	72.9± 5.1	73.4± 6.3	66.7± 4.9	52.9± 1.3	46.0
SemGCN	83.5± 4.9	74.8± 5.0	69.7± 4.6	69.2± 4.3	67.6± 4.4	58.4± 2.1	54.2± 1.9	48.6
GraphMLP	67.4± 3.8	69.6± 4.6	64.9± 4.2	61.9± 4.2	61.0± 3.7	55.3± 2.3	57.9± 2.8	57.2
GraFormer	70.9± 4.1	65.1± 4.5	66.2± 6.0	59.2± 4.1	59.2± 4.2	54.9± 2.1	52.8± 1.9	48.1
JointFormer	69.6± 3.3	66.5± 4.1	61.3± 3.6	66.3± 5.3	66.0± 6.5	55.1± 3.4	50.0± 1.3	46.0

Table A.8: Test results of selected-view pre-trained models in *center_mirror* view.

Drive&Act Test Set Result: <i>center_mirror</i> view								
Pre-trained with <i>All_Views</i> , Average MPJPE[mm]								
Model	Amount of Training data							
	5% of a subject	10% of a subject	25% of a subject	50% of a subject	single subject	two subjects	four subjects	all subjects
SimpleBL	85.6± 2.6	79.4± 5.7	71.0± 3.8	69.8± 3.7	71.9± 5.7	63.9± 3.6	51.9± 1.0	48.6
SemGCN	77.1± 4.1	75.3± 5.8	73.2± 5.2	69.1± 4.5	72.9± 8.1	65.2± 4.4	54.9± 1.3	51.9
GraphMLP	67.5± 3.5	69.9± 4.7	66.2± 4.9	61.7± 4.2	61.7± 4.8	56.8± 2.6	55.5± 2.5	52.8
GraFormer	68.6± 3.5	62.3± 3.3	61.0± 3.7	60.9± 3.4	58.6± 3.7	54.2± 2.2	51.4± 2.0	49.9
JointFormer	70.3± 4.9	64.5± 5.3	62.0± 4.6	66.5± 8.7	60.6± 5.6	57.7± 3.5	48.2± 1.2	45.4

Table A.9: Test results of *All_Views* pre-trained models in *center_mirror* view.

A.2 Comparison of Models With and Without Pre-training

The figures in this section illustrate the MPJPE trends, comparing the results between models trained from scratch and those pre-trained with synthetic data. The six figure groups correspond to the combinations of Drive&Act views and pre-training datasets. In each group, the figures in the left column provide a visual summary of the quantitative results reported in the tables. The horizontal axis, shown on a logarithmic scale, represents the amount of training data—ranging from 5%, 10%, and 50% of a subject to the full datasets of 1, 2, 4, and 8 subjects. The vertical axis denotes the MPJPE. Dashed lines indicate the from-scratch models, while solid lines represent the pre-trained models. The y-axis range is restricted to 0-150 mm, with data points beyond this range omitted for clarity.

The figures in the right column show the performance gains achieved through pre-training, plotted as the MPJPE improvement computed by subtracting the pre-trained model's error from that of the from-scratch model. Positive values therefore indicate a gain in accuracy. The lower bound of the y-axis in these plots is set to -5 mm. Standard errors are represented by error bars.

A. EXPERIMENTAL RESULTS FROM CHAPTER 5

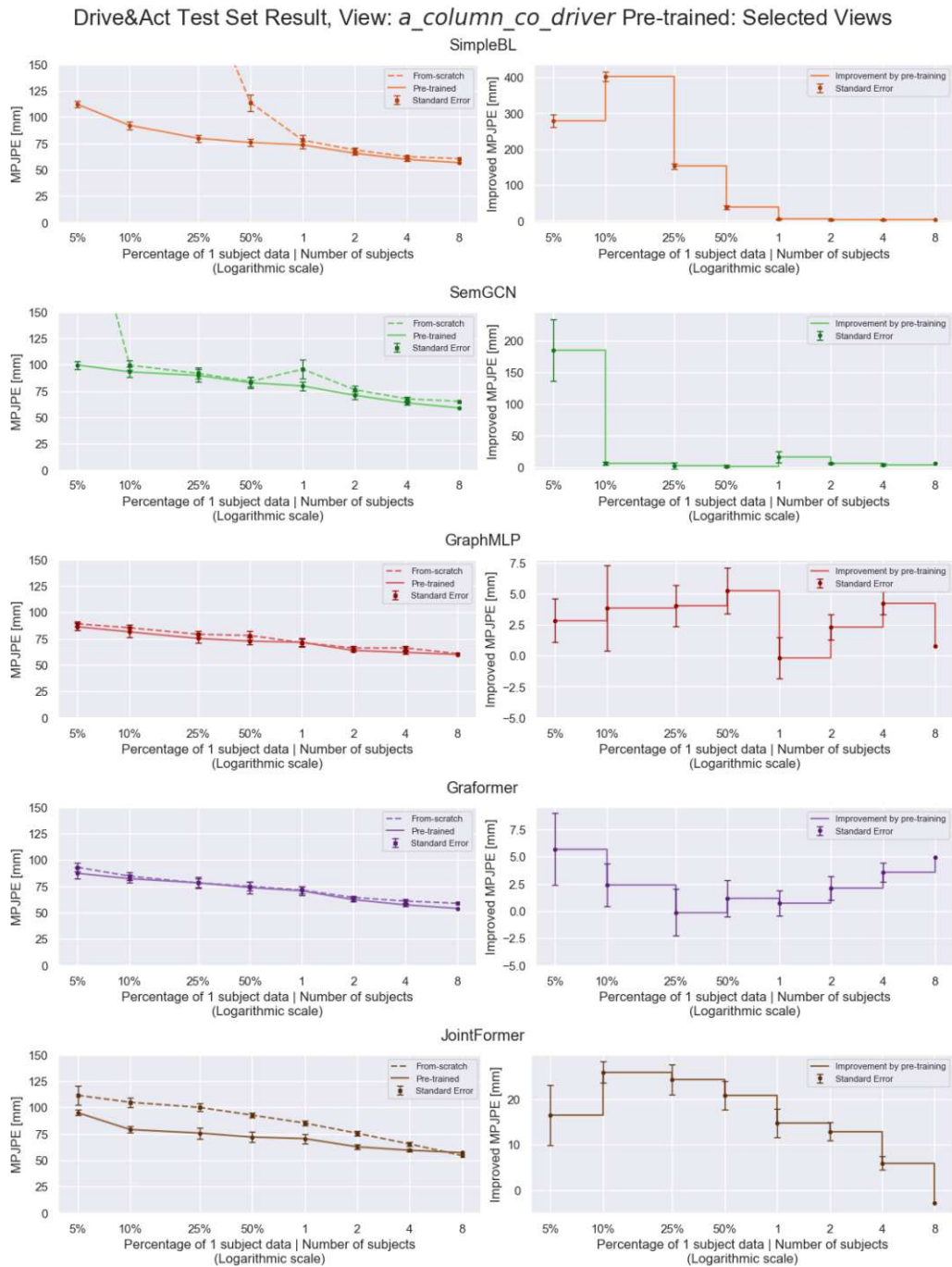


Figure A.1: Comparing evaluation result from *a_column_co_driver* view for model pre-trained with selected views and model without pre-training.

A.2. Comparison of Models With and Without Pre-training

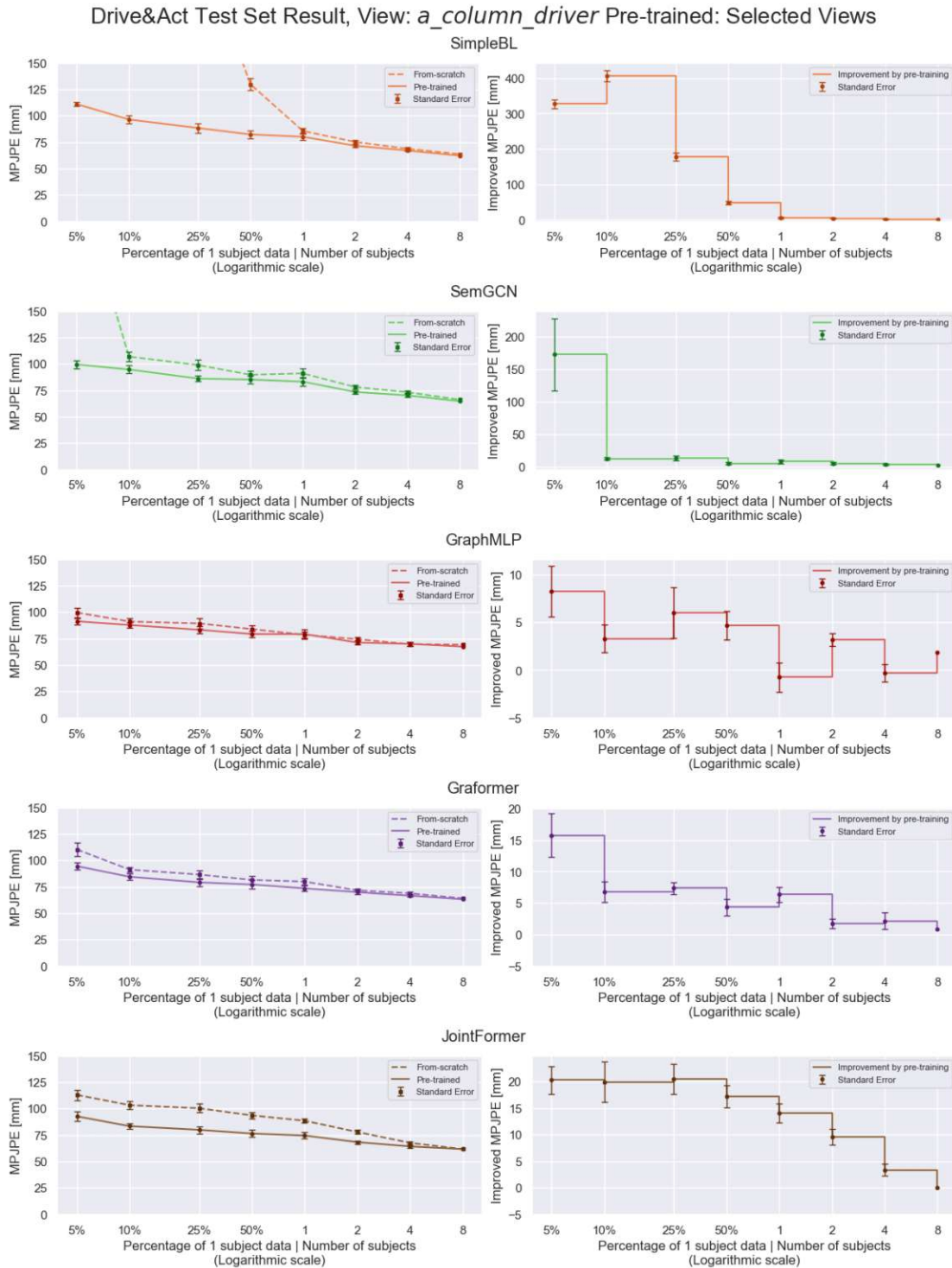


Figure A.2: Comparing evaluation result from *a_column_driver* view for model pre-trained with selected views and model without pre-training.

Drive&Act Test Set Result, View: *center_mirror* Pre-trained: Selected Views

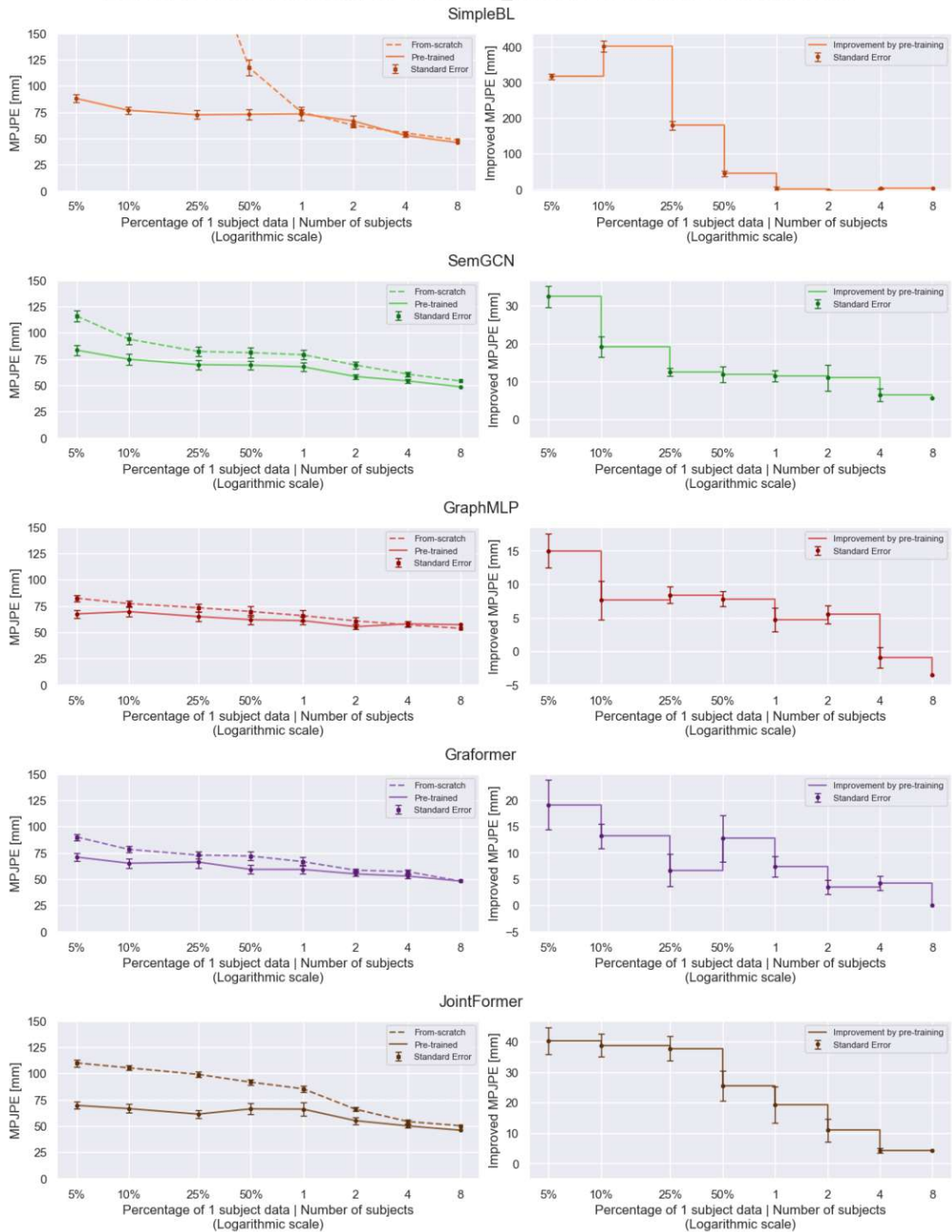


Figure A.3: Comparing evaluation result from *center_mirror* view for model pre-trained with selected views and model without pre-training.

A.2. Comparison of Models With and Without Pre-training

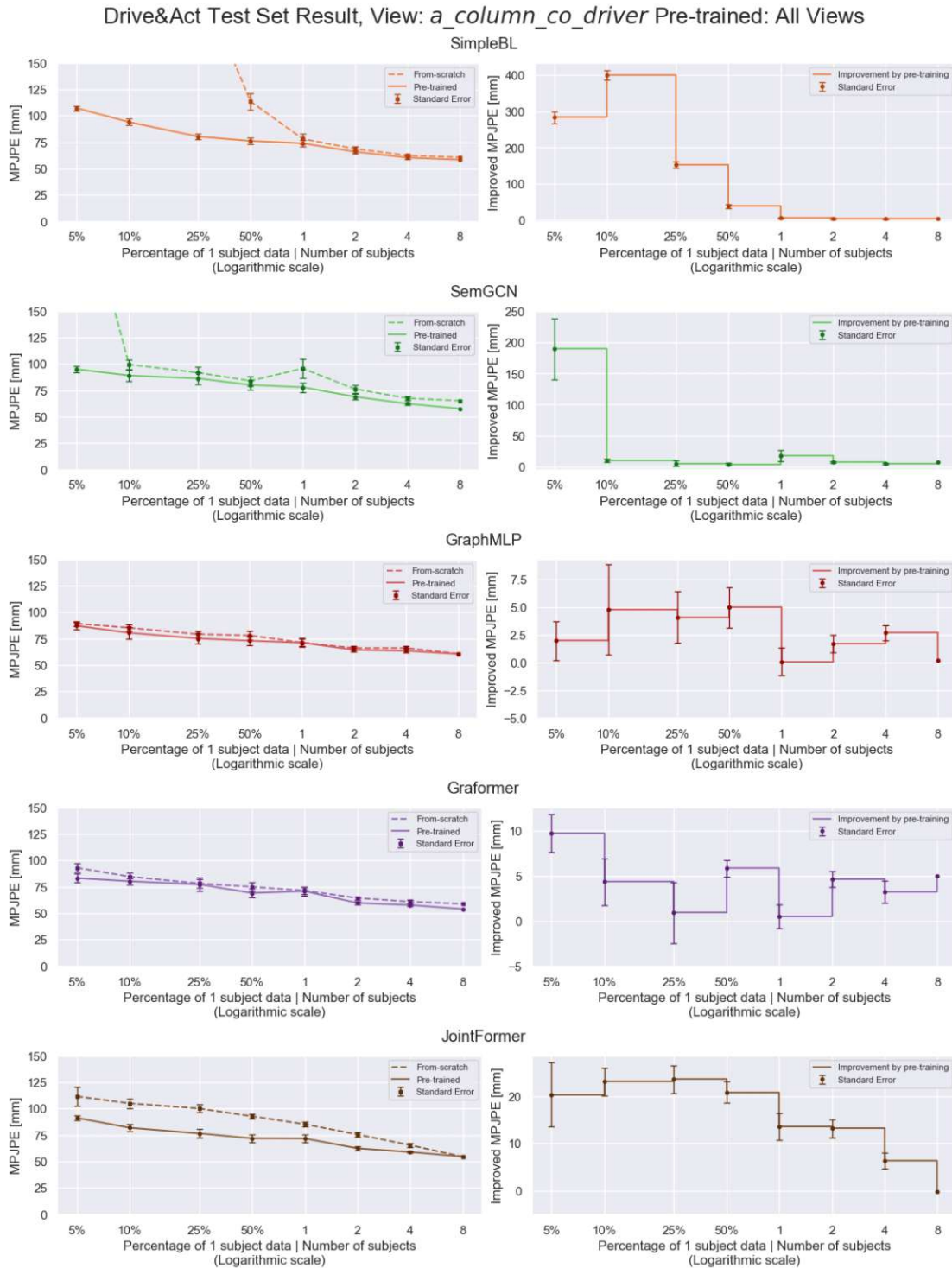


Figure A.4: Comparing evaluation result from *a_column_co_driver* view for model pre-trained with *All_Views* and model without pre-training.

Drive&Act Test Set Result, View: *a_column_driver* Pre-trained: All Views

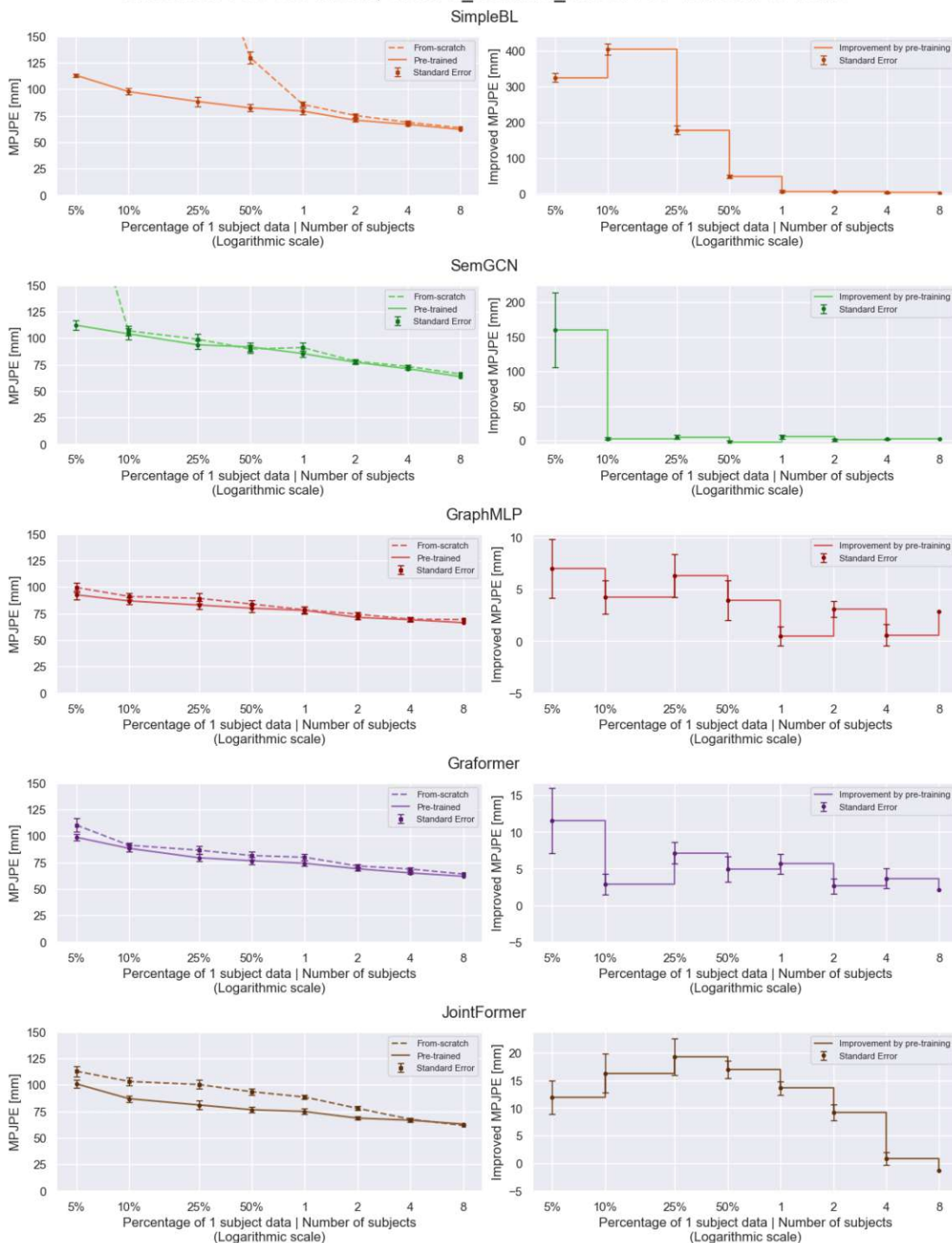


Figure A.5: Comparing evaluation result from *a_column_driver* view for model pre-trained with *All_Views* and model without pre-training.

A.2. Comparison of Models With and Without Pre-training

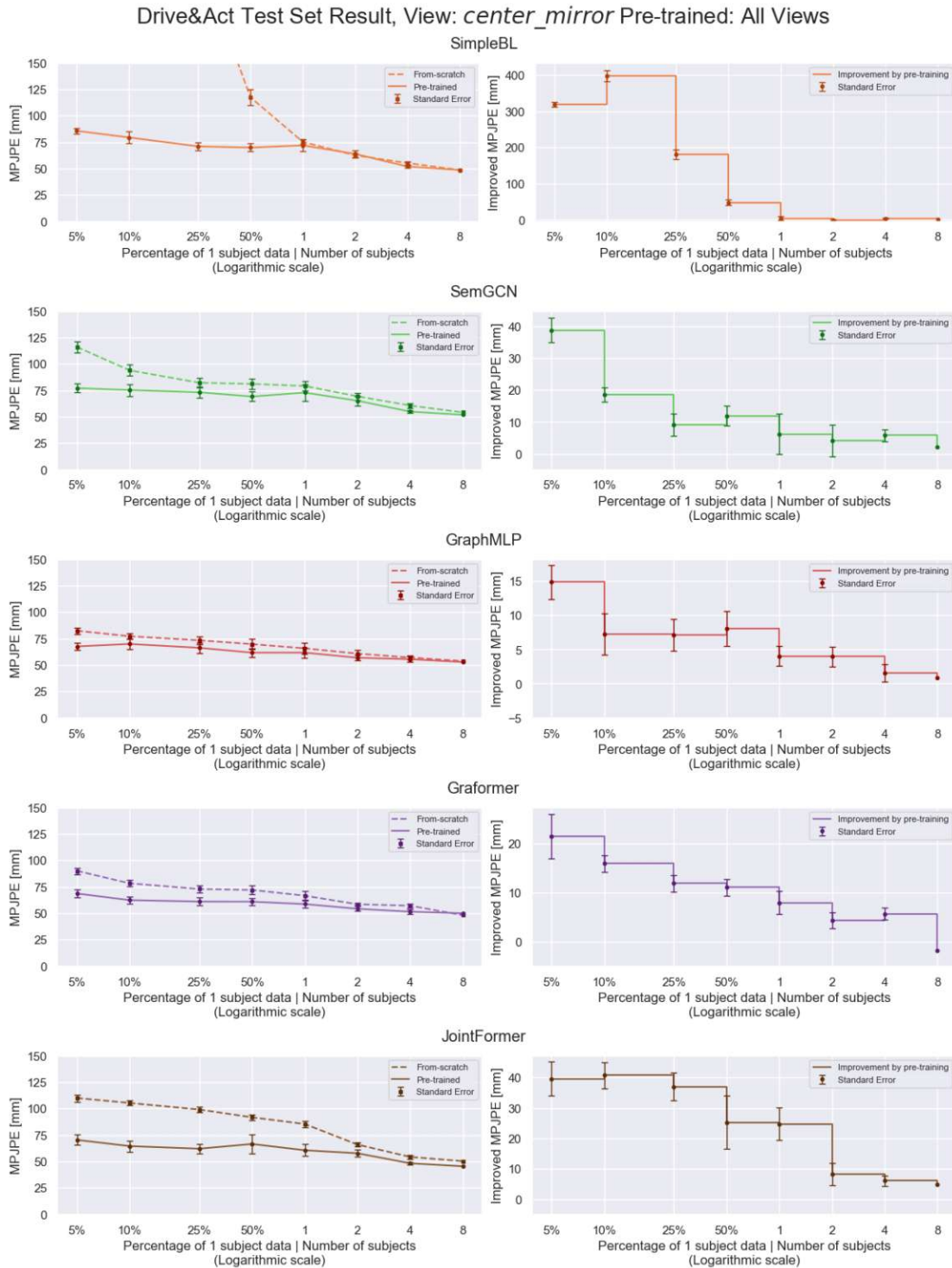


Figure A.6: Comparing evaluation result from *center_mirror* view for model pre-trained with *All_Views* and model without pre-training.

A.3 Example 3D Pose Outputs

This section compares the estimated 3D poses produced by different GraFormer models: one trained from scratch and two pre-trained using the selected-view and *All_Views* configurations. All three models were trained (or fine-tuned) with progressively larger subsets of the Drive&Act dataset, following the experimental setup described in Chapter 5. The same input image, shown in Figure A.7, was used for all models. This frame corresponds to Subject 12, captured from the *center_mirror* camera of the Drive&Act dataset.

The results show that the estimated poses become progressively closer to the ground-truth pose as the models are trained with more data. Moreover, the pre-trained models produce more accurate pose estimations than the from-scratch model when trained on the same amount of Drive&Act data. In the visualization, the green skeletal figures represent the ground-truth poses, while the black figures show the poses inferred by the models. Each row presents three views of the same pose to provide a clearer visual perspective.

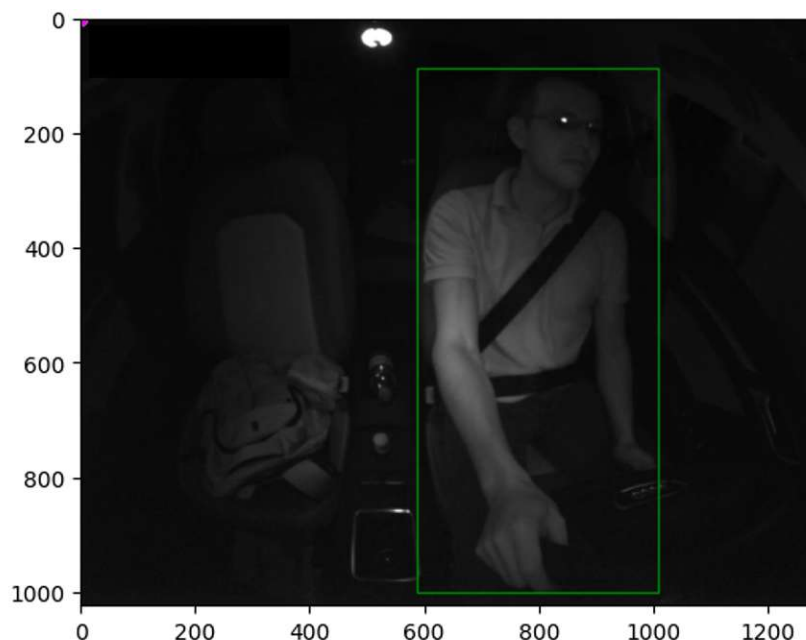
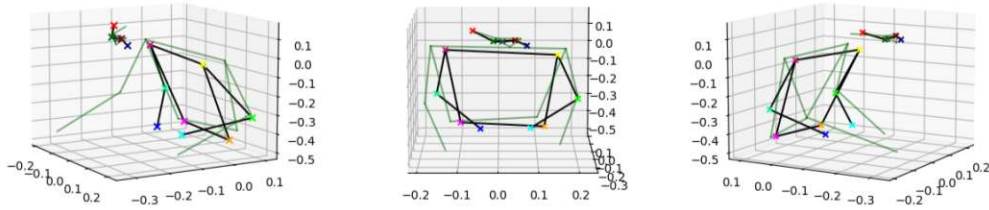
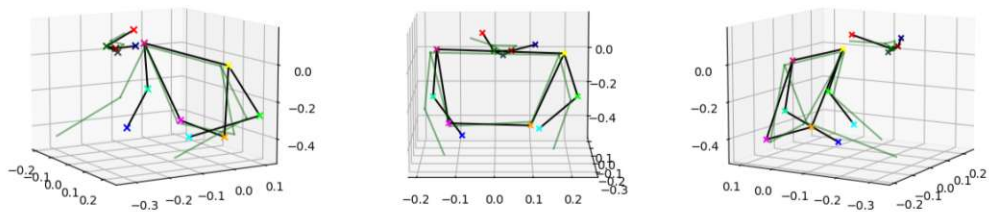


Figure A.7: The frame from Drive&Act that we used as input to the GraFormer model for pose estimation. The green box indicates the bounding box detected by Faster R-CNN.

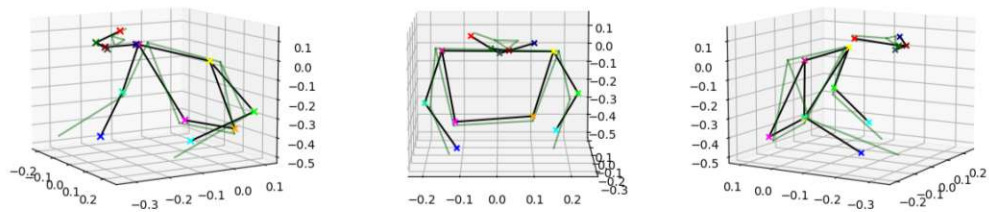
GraFormer with No Pre-training & Trained with 5% of a subject (MPJPE=65.3 mm)



GraFormer with No Pre-training & Trained with 10% of a subject (MPJPE=52.0 mm)



GraFormer with No Pre-training & Trained with 25% of a subject (MPJPE=44.1 mm)



GraFormer with No Pre-training & Trained with 50% of a subject (MPJPE=35.4 mm)

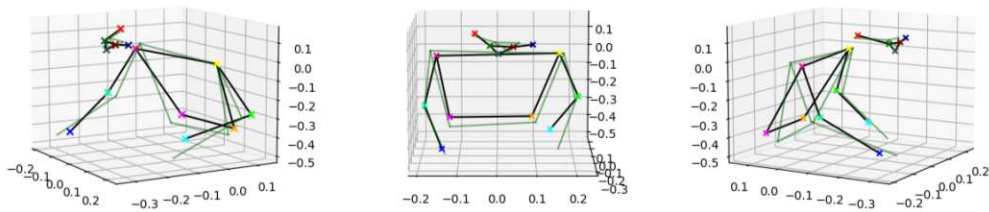
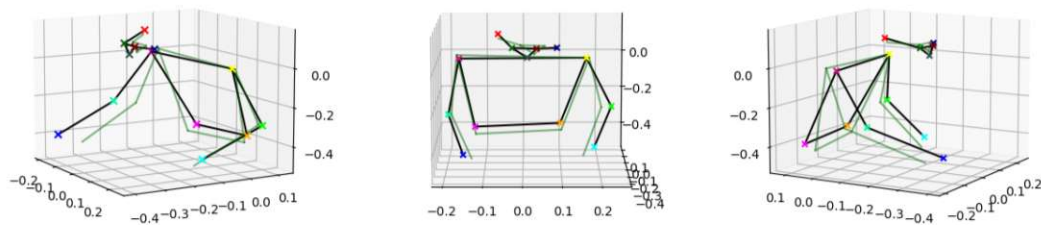


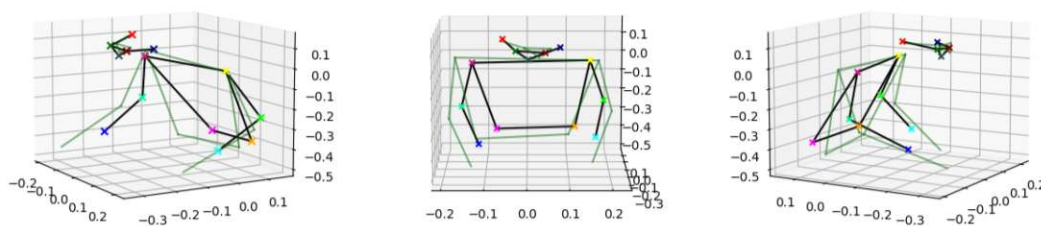
Figure A.8: Example pose predicted from GraFormer with no pre-training for the model fine-tuned with 5% to 50% of a subject. The green skeletal figures are the ground-truth poses. The black figures are the inferred poses from the model.

A. EXPERIMENTAL RESULTS FROM CHAPTER 5

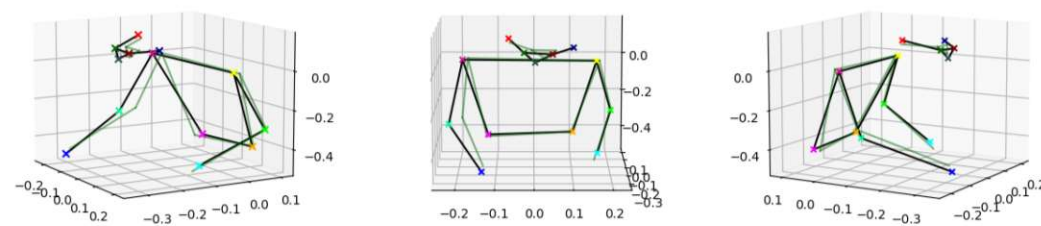
GraFormer with No Pre-training & Trained with 1 subject (MPJPE=37.5 mm)



GraFormer with No Pre-training & Trained with 2 subjects (MPJPE=57.2 mm)



GraFormer with No Pre-training & Trained with 4 subjects (MPJPE=23.7 mm)



GraFormer with No Pre-training & Trained with all subjects (MPJPE=26.8 mm)

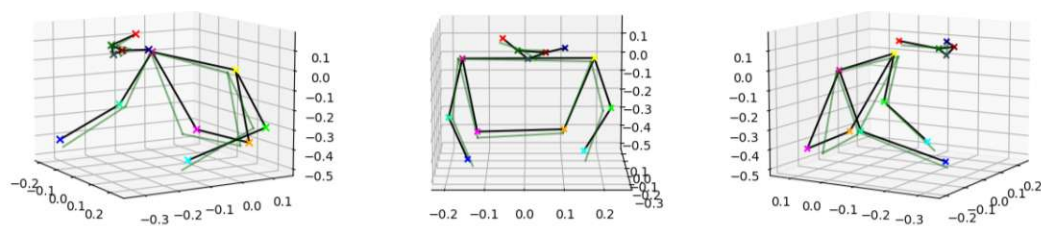
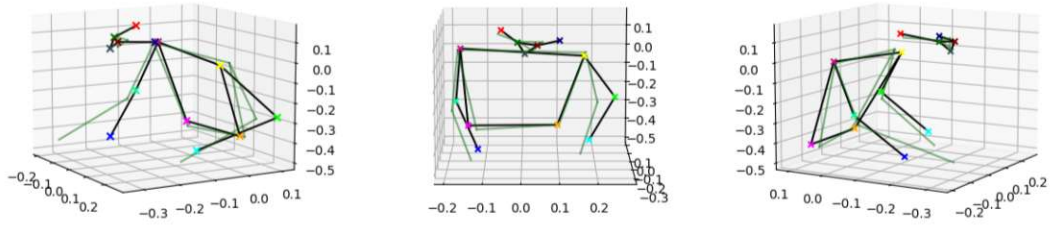
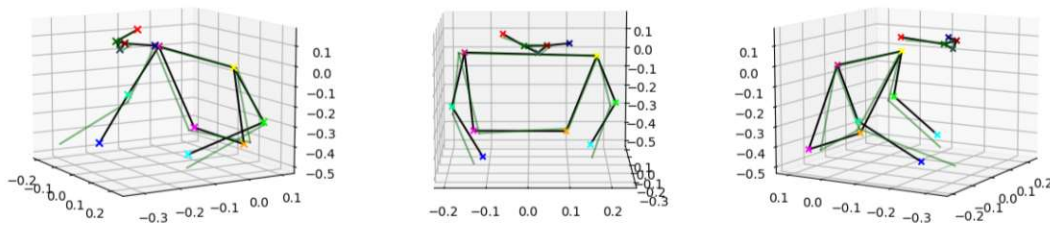


Figure A.9: Example pose predicted from GraFormer pre-trained with no pre-training for the model fine-tuned with 1 to 8 subjects. The green skeletal figures are the ground-truth poses. The black figures are the inferred poses from the model.

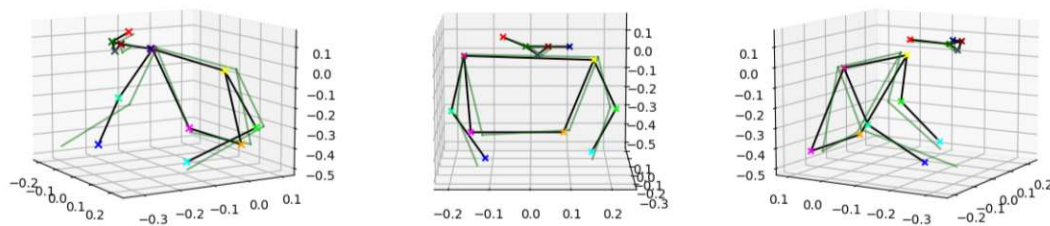
GraFormer Pre-trained with Selected Views & Fine-tuned with 5% of a subject (MPJPE=38.0 mm)



GraFormer Pre-trained with Selected Views & Fine-tuned with 10% of a subject (MPJPE=27.8 mm)



GraFormer Pre-trained with Selected Views & Fine-tuned with 25% of a subject (MPJPE=31.5 mm)



GraFormer Pre-trained with Selected Views & Fine-tuned with 50% of a subject (MPJPE=27.9 mm)

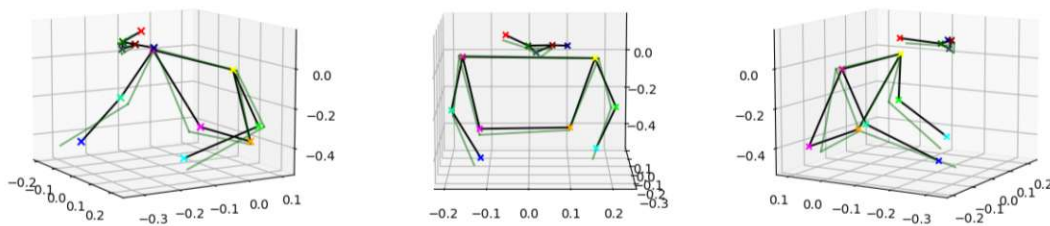
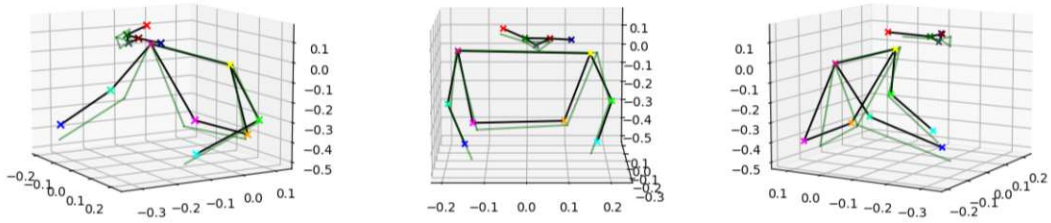


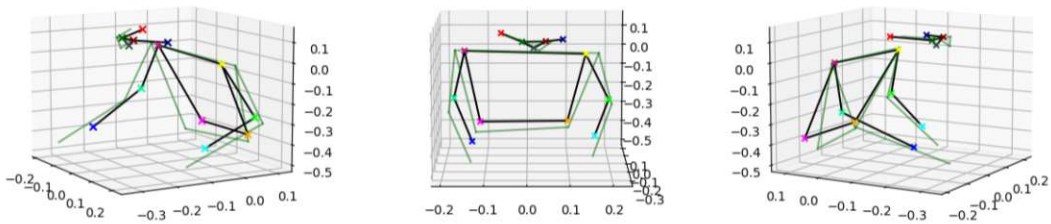
Figure A.10: Example pose predicted from GraFormer pre-trained with selected views for the model fine-tuned with 5% to 50% of a subject. The green skeletal figures are the ground-truth poses. The black figures are the inferred poses from the model.

A. EXPERIMENTAL RESULTS FROM CHAPTER 5

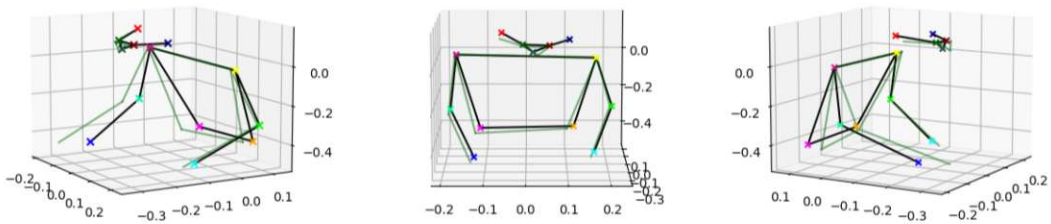
GraFormer Pre-trained with Selected Views & Fine-tuned with 1 Subject (MPJPE=34.6 mm)



GraFormer Pre-trained with Selected Views & Fine-tuned with 2 Subjects (MPJPE=45.7 mm)



GraFormer Pre-trained with Selected Views & Fine-tuned with 4 Subjects (MPJPE=29.3 mm)



GraFormer Pre-trained with Selected Views & Fine-tuned with all Subjects (MPJPE=22.2 mm)

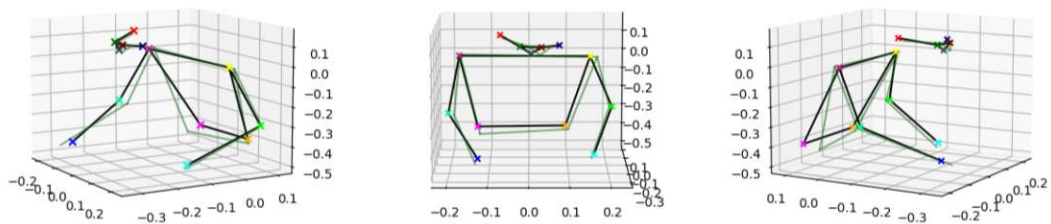


Figure A.11: Example pose predicted from GraFormer pre-trained with selected views for the model fine-tuned with 1 to 8 subjects. The green skeletal figures are the ground-truth poses. The black figures are the inferred poses from the model.

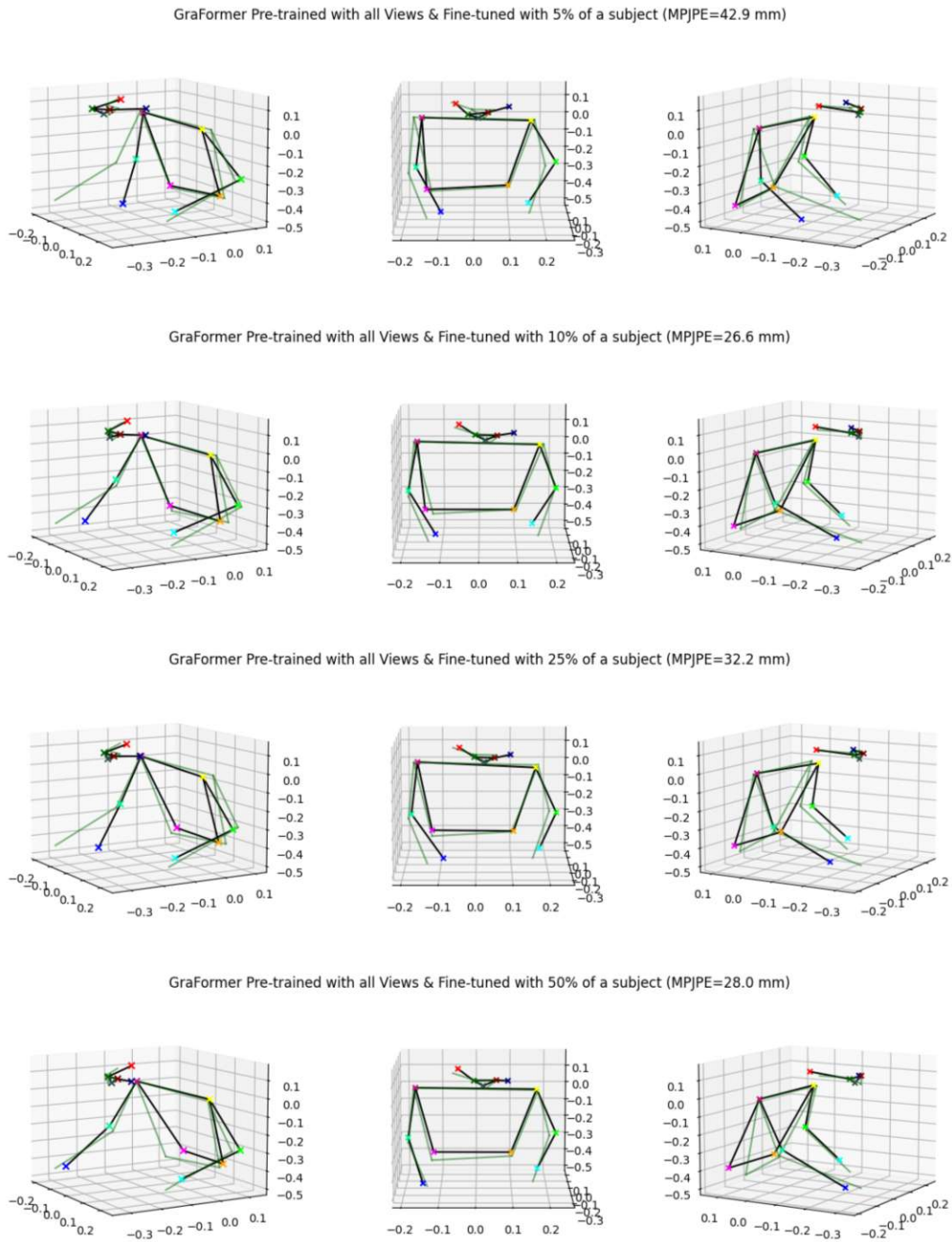


Figure A.12: Example pose predicted from GraFormer pre-trained with *All Views* for the model fine-tuned with 5% to 50% of a subject. The green skeletal figures are the ground-truth poses. The black figures are the inferred poses from the model.

A. EXPERIMENTAL RESULTS FROM CHAPTER 5

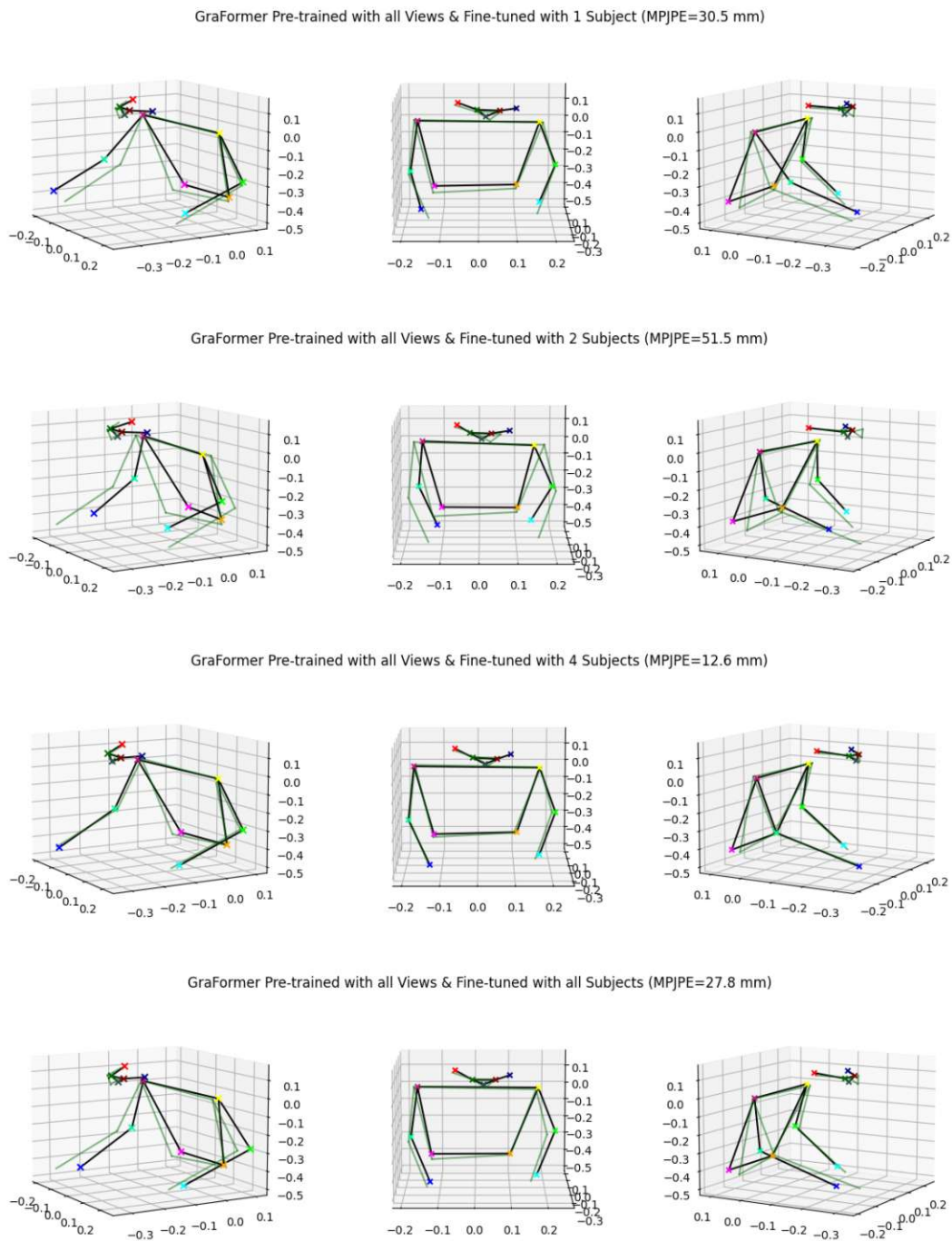


Figure A.13: Example pose predicted from GraFormer pre-trained with *All Views* for the model fine-tuned with 1 to 8 subjects. The green skeletal figures are the ground-truth poses. The black figures are the inferred poses from the model.

Overview of Generative AI Tools Used

This thesis employed AI tools exclusively for grammar checking and proofreading purposes. The original text was written entirely by the author. As a non-native English speaker, the author found it beneficial to use such tools to ensure grammatical accuracy and improve the overall flow of the language.

No part of this thesis was generated or written by generative AI beyond language correction or stylistic proofreading.

The following is a comprehensive list of the generative AI tools used for proofreading in this work.

- Grammarly¹ - Grammarly Pro was used in the early editing (from January to May 2024) especially in Chapter 1 for grammar checking and word suggestions. However, the tool was quite opinionated about certain writing styles, e.g., it strongly discouraged the use of passive voice. Thus, we didn't continue using it.
- Claude² (Claude 3.5 Sonnet, Claude 3.7 Sonnet, and Claude 4.5) and OpenAI³ ChatGPT 5, ChatGPT 5.1 and ChatGPT 5.2 were used for proofreading the content of the remaining chapters. The prompts used for proofreading were in the following format:

Proof read the following text.
"<The original text is inserted here.>"

¹<https://www.grammarly.com/>

²<https://claude.ai/>

³<https://chatgpt.com/>

As an example, Claude 3.7 Sonnet would produce output text in the following format.

Here's the proofread version:

"<Edited Text>"

Changes made:

- <Change #1 and its reason for the change, e.g., adding a comma, incorrect word used, incorrect tense, or suggesting usage of sentence conjunctions.>
- <Change #2 and its reason.>- ...

Here is an example of the usage from Claude.

Prompt:

Proof read the following text.

"The empirical evidence clearly demonstrated the benefit of pre-training: pre-trained models consistently achieved better results compared to from-scratch model when trained with the same amount of real-world data. Hybrid models such as GraphMLP and GraFormer tended to perform particularly well when real-world data were limited, while transformer-based models such as JointFormer outperformed the others only when sufficient fine-tuning data were available. For `center_mirror` view, GraphMLP pre-trained with selected views and fine-tuned with 5% of a subject produced the best test results MPJPE at 67.4 mm, compared to the from-scratch version which achieve 82.4 mm. Similarly, GraFormer pre-trained with selected views and fine-tuned with 50% of a subject gave an MPJPE of 59.2 mm, while it was 72.0 mm for the from-scratch one."

Output:

Here's the proofread version:

"The empirical evidence clearly demonstrated the benefit of pre-training: pre-trained models consistently achieved better results compared to from-scratch models when trained with the same amount of real-world data. Hybrid models such as GraphMLP and GraFormer tended to perform particularly well when real-world data were limited, while transformer-based models such as JointFormer outperformed the others only when sufficient fine-tuning data were available. For `center_mirror` view, GraphMLP pre-trained with selected views and fine-tuned with 5% of a subject's data produced the best test results MPJPE at 67.4 mm, compared to the from-scratch version which achieved 82.4 mm. Similarly, GraFormer pre-trained with selected views and fine-tuned with 50% of a subject's data achieved an MPJPE of 59.2 mm, compared to 72.0 mm for the from-scratch version."

Changes made:

- "from-scratch model" → "from-scratch models" Subject-verb agreement: "models" should be plural to match "pre-trained models" in the comparison.
- "which achieve" → "which achieved" Tense consistency: should be past tense to match "produced" earlier in the sentence.
- "5% of a subject" and "50% of a subject" → "5% of the subject's data" and "50% of the subject's data" (or similar clarification) Clarity: "5% of a subject" is ambiguous—it sounds like a portion of a person rather than a portion of the data from one subject.
- "gave an MPJPE of 59.2 mm, while it was 72.0 mm" → "achieved an MPJPE of 59.2 mm, compared to 72.0 mm" Consistency and clarity: "gave" is less formal than "achieved/produced" used elsewhere; "while it was" has an ambiguous pronoun reference.

Note that different AI models and their versions may produce slight variations in details and formatting.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Figures

2.1	Perceptron Models. Figure obtained from [BB24].	6
2.2	Convolution Mechanism of Convolutional Neural Network. Figure obtained from [Pri23].	8
2.3	Architecture of the transformer model. Figure obtained from [VSP ⁺ 17].	10
2.4	Example images from 2D HPE datasets.	19
2.5	Example images from 3D HPE datasets.	21
2.6	Example images from synthetic HPE datasets.	22
3.1	General structure of three-stage 3D Human Pose Estimation.	27
3.2	Architecture of Faster R-CNN. Figure from [RHGS15].	29
3.3	Architecture of HRNet. Figure from [SXLW19].	30
3.4	Architecture of the Simple Baseline Model. Figure from [MHRL17].	31
3.5	Architecture of the SemGCN. Figure from [ZPT ⁺ 19].	32
3.6	Architecture of the GraphMLP. Figures from [LLG ⁺ 25].	33
3.7	Architecture of the GraFormer. Figure from [ZWT22].	34
3.8	Architecture of the JointFormer. Figure from [LBG ⁺ 22].	35
3.9	Training of the RepNet. Figure from [WR19].	36
4.1	Example images from the SyntheticCabin IR dataset.	41
4.2	3D pose estimation pipeline for the experiment in Chapter 4.	41
4.3	Skeleton pose representation showing keypoints used for upper-body driver pose estimation.	42
4.4	Example outputs for each image view of SyntheticCabin IR.	48
5.1	Example images from SyntheticCabin IR 1M, and Drive&Act dataset [MRH ⁺ 19].	53
5.2	Setup of the 3D pose estimation training pipeline with transfer learning.	56
5.3	Example images for the camera positions of SyntheticCabin IR 1M targeting each Drive&Act view.	58
5.4	Evaluation results for GraFormer on the testing set of Drive&Act (<i>center_mirror</i>).	61
6.1	Pose estimation pipeline for Approach 1. The 2D-to-3D pose lifter was pre-trained with SyntheticCabin IR 1M but was not fine-tuned with real-world data, unlike in the previous chapter.	65
		99

6.2	GAN training of RepNet was conducted by interleaving epochs of training the critic network and the pose generator network.	68
6.3	Example outputs for each model. The green skeletal figures are the ground truth poses. The black figures are the inferred poses from the models. . .	71
A.1	Comparing evaluation result from <i>a_column_co_driver</i> view for model pre-trained with selected views and model without pre-training.	82
A.2	Comparing evaluation result from <i>a_column_driver</i> view for model pre-trained with selected views and model without pre-training.	83
A.3	Comparing evaluation result from <i>center_mirror</i> view for model pre-trained with selected views and model without pre-training.	84
A.4	Comparing evaluation result from <i>a_column_co_driver</i> view for model pre-trained with <i>All_Views</i> and model without pre-training.	85
A.5	Comparing evaluation result from <i>a_column_driver</i> view for model pre-trained with <i>All_Views</i> and model without pre-training.	86
A.6	Comparing evaluation result from <i>center_mirror</i> view for model pre-trained with <i>All_Views</i> and model without pre-training.	87
A.7	The frame from Drive&Act that we used as input to the GraFormer model for pose estimation. The green box indicates the bounding box detected by Faster R-CNN.	88
A.8	Example pose predicted from GraFormer with no pre-training for the model fine-tuned with 5% to 50% of a subject. The green skeletal figures are the ground-truth poses. The black figures are the inferred poses from the model.	89
A.9	Example pose predicted from GraFormer pre-trained with no pre-training for the model fine-tuned with 1 to 8 subjects. The green skeletal figures are the ground-truth poses. The black figures are the inferred poses from the model.	90
A.10	Example pose predicted from GraFormer pre-trained with selected views for the model fine-tuned with 5% to 50% of a subject. The green skeletal figures are the ground-truth poses. The black figures are the inferred poses from the model.	91
A.11	Example pose predicted from GraFormer pre-trained with selected views for the model fine-tuned with 1 to 8 subjects. The green skeletal figures are the ground-truth poses. The black figures are the inferred poses from the model.	92
A.12	Example pose predicted from GraFormer pre-trained with <i>All_Views</i> for the model fine-tuned with 5% to 50% of a subject. The green skeletal figures are the ground-truth poses. The black figures are the inferred poses from the model.	93
A.13	Example pose predicted from GraFormer pre-trained with <i>All_Views</i> for the model fine-tuned with 1 to 8 subjects. The green skeletal figures are the ground-truth poses. The black figures are the inferred poses from the model.	94

List of Tables

2.1	Summary of Training Paradigms.	14
3.1	Summary of Model Characteristics.	37
4.1	Evaluation result for fine-tuned human detector (Faster R-CNN)	44
4.2	Evaluation result for fine-tuned 2D Pose Estimator (HRNet)	45
4.3	Evaluation results on the test set using triangulation, reported in [SSN ⁺ 23].	47
4.4	Evaluation result on the test set for <i>A_Pillar_Codriver</i> pillar view.	47
4.5	Evaluation result on the test set for the <i>A_Pillar_Driver</i> view.	47
4.6	Evaluation result on the test set for the <i>Rear_Mirror</i> view.	47
5.1	List of camera positions in SyntheticCabin IR-1M targeting each Drive&Act view. The last row shows an additional setup used to pre-train models on the full synthetic dataset.	58
5.2	Test results of models without pre-training in <i>center_mirror view</i> . Bold values denote the best model for each quantity of training data.	60
5.3	Test results of selected-view pre-trained models in <i>center_mirror view</i>	60
5.4	Test results of <i>All_Views</i> pre-trained models in <i>center_mirror view</i>	60
6.1	Test set results for models, each trained with selected views of SyntheticCabin IR 1M corresponding to each target view of the Drive&Act dataset.	67
6.2	Test set result for the model trained with <i>All_views</i> of SyntheticCabin IR 1M for each target view of Drive&Act.	67
6.3	Test set results for models trained on each target view of Drive&Act, provided for comparison.	67
6.4	Evaluation result on the test set for RepNet. The top row was trained with Synthetic IR 1M and predicted 2D poses from Drive&Act. For comparison, the second row used predicted 2D poses and annotated 3D poses of Drive&Act.	70
A.1	Test result of model without pre-training in <i>a_column_co_driver view</i>	78
A.2	Test results of selected-view pre-trained models in <i>a_column_co_driver view</i>	78
A.3	Test results of <i>All_Views</i> pre-trained models in <i>a_column_co_driver view</i>	78
A.4	Test results of models without pre-training in <i>a_column_driver view</i>	79
A.5	Test results of selected-view pre-trained models in <i>a_column_driver view</i>	79
		101

A.6	Test results of <i>All_Views</i> pre-trained models in <i>a_column_driver</i> view. . .	79
A.7	Test results of models without pre-training in <i>center_mirror</i> view.	80
A.8	Test results of selected-view pre-trained models in <i>center_mirror</i> view. . .	80
A.9	Test results of <i>All_Views</i> pre-trained models in <i>center_mirror</i> view. . . .	80

Glossary

2D-to-3D pose lifter A deep learning models which take 2D anatomical key points as an input and estimate 3D key points as an output.. 3–5, 24, 27, 28, 30, 31, 35–37, 39, 41, 42, 45, 46, 49, 51, 52, 56, 57, 59, 62, 64–66, 68, 69, 73–75, 77

SyntheticCabin A project aims to develop a simulation system that generates synthetic images of vehicle interiors, enabling the efficient development of driver/occupant monitoring systems. View more at <https://projekte.ffg.at/projekt/3992951>. 3, 4, 52

UNISCOPE-3D A project aimed at enabling robust monocular 3D human body, pose, and action analysis using computer vision and deep learning, with a primary focus on vehicle occupant monitoring applications. View more at <https://projekte.ffg.at/projekt/5134999>. 3



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acronyms

- 2D HPE** 2D human pose estimation. 14, 16–20, 23–25, 27, 28, 30, 39, 41, 44, 51, 52, 56, 57, 65, 74
- 3D HPE** 3D human pose estimation. 2–4, 14, 16, 17, 19–25, 27, 31, 56, 73, 74
- AP** Average Precision. 43–45
- CNN** convolutional neural network. 7, 8, 23
- COCO** Common Objects in Context. 18, 19, 28, 41, 43, 44, 56, 57
- DMS** driver monitoring systems. 1, 2, 23, 52, 74
- Euro NCAP** European New Car Assessment Programme. 2
- Faster R-CNN** Faster Region-based Convolutional Neural Network. 28, 29, 41, 43, 45, 46, 56, 57
- FFG** Austrian Research Promotion Agency. 3
- GAN** generative adversarial network. 11, 35, 66, 68
- GCN** graph convolutional network. 9, 33, 34
- GNN** graph neural network. 8, 9, 31–33
- GraFormer** Graph Convolution Transformer for 3D Pose Estimation. 33, 34, 41, 46, 59–62, 66, 74, 77, 88, 96, 100
- GraphMLP** Graph MLP-Like Architecture for 3D Human Pose Estimation. 32, 33, 37, 41, 46, 61, 62, 74, 96
- HPE** human pose estimation. 2, 5, 14–17, 28
- HRNet** High-Resolution Network. 28, 30, 41, 44–46, 56, 57, 99

IoU Intersection over Union. 43

JointFormer Single-Frame Lifting Transformer with Error Prediction and Refinement for 3D Human Pose Estimation. 34, 35, 41, 46, 47, 49, 60–62, 74, 96

MLP multi-layer perceptron. 6, 7, 23, 24, 33

MoCap motion capture. 14, 19–22

MPJPE mean per-joint position error. 16, 17, 24, 40, 41, 46, 49, 54, 55, 59–62, 64, 73, 74, 77, 81, 96, 97

NLP natural language processing. 5, 9

PA-MPJPE Procrustes Aligned mean per-joint position error. 17, 64, 66, 69, 70, 74

PJPE per-joint position error. 17, 40, 41, 46

RepNet Reprojection Network. 35, 36, 66, 68–70, 74, 101

RNN recurrent neural network. 9

SemGCN Semantic Graph Convolutional Networks. 31, 32, 37, 41, 45, 65, 66, 70

SimpleBL Simple Baseline Model for 2D-to-3D Pose Estimation. 31, 36, 41, 45

Bibliography

- [ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, page 214–223, 2017.
- [AHMJ⁺14] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(10):1533–1545, 2014.
- [AMA⁺23] Sirwan K. Ahmed, Mona G. Mohammed, Salar O. Abdulqadir, Rabab G. Abd El-Kader, Nahed A. El-Shall, Deepak Chandran, Mohammad E. Ur Rehman, and Kuldeep Dhama. Road traffic accidental injuries and deaths: A neglected global health issue. *Health Science Reports*, 6(5):e1240, 2023.
- [APGS14] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2D human pose estimation: New benchmark and state of the art analysis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, page 3686–3693, 2014.
- [AYKY24] Hyojung Ahn, Jeongmin Yu, Jonghan Ko, and Jong-Min Yeom. Enhanced short-term prediction of solar radiation using HRNet model with geostationary satellite data. *IEEE Geoscience and Remote Sensing Letters*, 21:1–5, 2024.
- [BB24] Christopher M Bishop and Hugh Bishop. *Deep learning: foundations and concepts*. Springer, 2024.
- [BBO17] Anastasia Borovykh, Sander Bohte, and Kees Oosterlee. Conditional time series forecasting with convolutional neural networks. In *International Conference on Artificial Neural Networks*, 2017.
- [BCB15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, 2015.

- [CHL24] Xiao Cao, Wei Hu, and Hui Liu. Integrated driver pose estimation for autonomous driving. In *International Joint Conference on Biomedical Engineering Systems and Technologies*, page 695–702, 2024.
- [CHS⁺19] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. OpenPose: Realtime multi-person 2D pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1302–1310, 2019.
- [CTG⁺19] Yujin Chen, Zhigang Tu, Liuhaog Ge, Dejun Zhang, Ruizhi Chen, and Junsong Yuan. SO-HandNet: Self-organizing network for 3D hand pose estimation with semi-supervised learning. In *IEEE/CVF International Conference on Computer Vision*, page 6960–6969, 2019.
- [CTH20] Yucheng Chen, Yingli Tian, and Mingyi He. Monocular human pose estimation: A survey of deep learning-based methods. *Computer Vision and Image Understanding*, 192(102897), 2020.
- [CWP⁺18] Yilun Chen, Zhicheng Wang, Yuxiang Peng, Zhiqiang Zhang, Gang Yu, and Jian Sun. Cascaded pyramid network for multi-person pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, page 7103–7112, 2018.
- [CWP⁺19] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open MMLab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.
- [DBK⁺21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.
- [DBV16] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, volume 29, pages 3844–3852, 2016.
- [DCWB⁺20] Steve Dias Da Cruz, Oliver Wasenmuller, Hans-Peter Beise, Thomas Stifter, and Didier Stricker. SVIRO: Synthetic vehicle interior rear seat occupancy dataset and benchmark. In *IEEE Winter Conference on Applications of Computer Vision*, page 962–971, 2020.

- [DDS⁺09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.
- [DGL⁺16] Thomas A. Dingus, Feng Guo, Suzie Lee, Jonathan F. Antin, Miguel Perez, Mindy Buchanan-King, and Jonathan Hankey. Driver crash risk factors and prevalence evaluation using naturalistic driving data. *Proceedings of the National Academy of Sciences*, 113(10):2636–2641, 2016.
- [DSWV16] Quentin De Smedt, Hazem Wannous, and Jean-Philippe Vandeborre. Skeleton-based dynamic hand gesture recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, page 1206–1214, 2016.
- [EKN⁺17] Andre Esteva, Brett Kuperl, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, 2017.
- [FT] William Falcon and The PyTorch Lightning team. PyTorch lightning. <https://github.com/Lightning-AI/lightning>. Accessed: 2023-07-01.
- [GAA⁺17] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein GANs. In *Advances in Neural Information Processing Systems*, volume 30, pages 5769–5779, 2017.
- [GAG⁺17] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*, page 1243–1252, 2017.
- [GBB11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, page 315–323, 2011.
- [GCJT21] Romain Guesdon, Carlos Crispim-Junior, and Laure Tougne. DriPE: A dataset for human pose estimation in real-world driving settings. In *IEEE/CVF International Conference on Computer Vision Workshops*, page 2865–2874, 2021.
- [GDDM14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

- [GGJY20] Chenjie Ge, Irene Yu-Hua Gu, Asgeir Store Jakola, and Jie Yang. Deep semi-supervised learning for brain tumor classification. *BMC Medical Imaging*, 20(1), 2020.
- [Gir15] Ross Girshick. Fast R-CNN. In *IEEE/CVF International Conference on Computer Vision*, pages 1440–1448, 2015.
- [GMS25] Priyanka Goyal, Adway Mitra, and Manjira Sinha. Sphrnet: Generating high-resolution crop maps from remote sensing imagery using hrnet with separable convolution. In *International Conference on Data Science and Management of Data*, page 27–34. Association for Computing Machinery, 2025.
- [GPAM⁺14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.
- [GSR⁺17] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, volume 70, pages 1263–1272, 2017.
- [GZF21] Kehong Gong, Jianfeng Zhang, and Jiashi Feng. PoseAug: A differentiable pose augmentation framework for 3D human pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, page 8575–8584, 2021.
- [HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *IEEE/CVF International Conference on Computer Vision*, pages 2980–2988, 2017.
- [HLVDMW17] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, page 2261–2269, 2017.
- [Hod] Jessica Hodgins. CMU Mocap Database. <http://mocap.cs.cmu.edu/>. Accessed: 2025-04-30.
- [HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
- [HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

- [HVU⁺18] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Curtis Hawthorne, Andrew M Dai, Matthew D Hoffman, and Douglas Eck. Music Transformer: Generating music with long-term structure. In *International Conference on Learning Representations*, 2018.
- [HYL17] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1025–1035, 2017.
- [HZG⁺23] Rongtian Huo, Ye Zhang, Yulan Guo, Zhaojie Ju, and Qing Gao. GT-Former: 3D driver body pose estimation in video with graph convolution network and transformer. *IEEE Transactions on Intelligent Vehicles*, page 1–12, 2023.
- [HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, page 770–778, 2016.
- [IPOS14] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6M: Large scale datasets and predictive methods for 3D human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339, 2014.
- [JAFF16] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, page 694–711, 2016.
- [JE10] Sam Johnson and Mark Everingham. Clustered pose and nonlinear appearance models for human pose estimation. In *British Machine Vision Conference*, pages 12.1–12.11, 2010.
- [JE11] Sam Johnson and Mark Everingham. Learning effective human pose estimation from inaccurate annotation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, page 1465–1472, 2011.
- [JEP⁺21] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, page 583–589, 2021.

- [JLT⁺15] Hanbyul Joo, Hao Liu, Lei Tan, Lin Gui, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social motion capture. In *IEEE/CVF International Conference on Computer Vision*, page 3334–3342, 2015.
- [JSD⁺14] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM international conference on Multimedia*, page 675–678, 2014.
- [KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 84 – 90, 2012.
- [KSZ15] Moritz Körber, Wolfgang Schneider, and Markus Zimmermann. Vigilance, boredom proneness and detection time of a malfunction in partially automated driving. In *International Conference on Collaboration Technologies and Systems*, pages 70–76, 2015.
- [KW16] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2016.
- [KYH⁺24] Kwang-Lim Ko, Jun-Sang Yoo, Chang-Woo Han, Jungyeop Kim, and Seung-Won Jung. Pose and shape estimation of humans in vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 25(1):402–416, 2024.
- [LBBH98] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [LBG⁺22] Sebastian Lutz, Richard Blythman, Koustav Ghostal, Matthew Moynihan, Ciaran Simms, and Aljosa Smolic. JointFormer: Single-frame lifting transformer with error prediction and refinement for 3D human pose estimation. In *International Conference on Pattern Recognition*, pages 1156–1163, 2022.
- [LBH15] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 2015.
- [LBOM12] Yann A. LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. *Efficient BackProp*, page 9–48. Springer Berlin Heidelberg, 2012.
- [LBSM23] Wu Liu, Qian Bao, Yu Sun, and Tao Mei. Recent advances of monocular 2D and 3D human pose estimation: A deep learning perspective. *ACM Computing Surveys*, 55(4):1–41, 2023.

- [LD15] Shuying Liu and Weihong Deng. Very deep convolutional neural network based image classification using small training sample size. In *IAPR Asian Conference on Pattern Recognition*, page 730–734, 2015.
- [LFV⁺17] Colin Lea, Michael D. Flynn, René Vidal, Austin Reiter, and Gregory D. Hager. Temporal convolutional networks for action segmentation and detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1003–1012, 2017.
- [LLG⁺25] Wenhao Li, Hong Liu, Tianyu Guo, Hao Tang, and Runwei Ding. GraphMLP: A graph MLP-Like architecture for 3D human pose estimation. *Pattern Recognition*, 158:110925, 2025.
- [LMB⁺15] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, page 740–755, 2015.
- [LMR⁺15] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Transactions on Graphics*, 34(6):248:1–248:16, 2015.
- [LXD⁺22] Xiaomei Li, Zhijiang Xie, Xiong Deng, Yanxue Wu, and Yangjun Pi. Traffic sign detection based on improved faster R-CNN for autonomous driving. *The Journal of Supercomputing*, 78(6):7982–8002, 2022.
- [LXT⁺18] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, volume 31, pages 6391–6401, 2018.
- [MAS⁺22] Vittorio Mazzia, Simone Angarano, Francesco Salvetti, Federico Angelini, and Marcello Chiaberge. Action Transformer: A self-attention model for short-time pose-based human action recognition. *Pattern Recognition*, 124:108487, 2022.
- [MBB⁺21] Dietrich Manstetten, Frank Beruscha, Hans-Joachim Bieg, Fanny Kobiela, Andreas Korthauer, Wolfgang Krautter, and Claus Marberger. The evolution of driver monitoring systems: A shortened story on past, current and future approaches how cars acquire knowledge about the driver’s state. In *International Conference on Human-Computer Interaction with Mobile Devices and Services*, page 1–6, 2021.
- [MCL19] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. Camera distance-aware top-down approach for 3D multi-person pose estimation from a single rgb image. In *IEEE/CVF International Conference on Computer Vision*, page 10132–10141, 2019.

- [MHRL17] Julieta Martinez, Rayat Hossain, Javier Romero, and James J. Little. A simple yet effective baseline for 3D human pose estimation. In *IEEE/CVF International Conference on Computer Vision*, pages 2659–2668, 2017.
- [Mik12] Tomáš Mikolov. *Statistical Language Models Based on Neural Networks*. PhD thesis, Brno University of Technology, 2012.
- [MKM22] Iman Mahdinia, Asad J. Khattak, and Antora Mohsena Haque. How effective are pedestrian crash prevention systems in improving pedestrian safety? harnessing large-scale experimental data. *Accident Analysis & Prevention*, 171:106669, 2022.
- [MME22] MMEngine Contributors. MMEngine: OpenMMLab foundational library for training deep learning models. <https://github.com/open-mmlab/mengine>, 2022. Accessed: 2023-07-04.
- [MMP20] MMPose Contributors. OpenMMLab pose estimation toolbox and benchmark. <https://github.com/open-mmlab/mmpose>, 2020. Accessed: 2023-07-18.
- [MO14] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.
- [MP69] Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969.
- [MRH⁺19] Manuel Martin, Alina Roitberg, Monica Haurilet, Matthias Horne, Simon Reiß, Michael Voit, and Rainer Stiefelhagen. Drive&Act: A multi-modal dataset for fine-grained driver behavior recognition in autonomous vehicles. In *IEEE/CVF International Conference on Computer Vision*, pages 2801–2810, 2019.
- [MVS21] Manuel Martin, Michael Voit, and Rainer Stiefelhagen. An evaluation of different methods for 3D-driver-body-pose estimation. In *IEEE International Intelligent Transportation Systems Conference*, page 1578–1584, 2021.
- [NPN⁺24] Akhil Nair, Varad Patil, Rohan Nair, Adithi Shetty, and Mimi Cherian. A review on recent driver safety systems and its emerging solutions. *International Journal of Computers and Applications*, 46(3):137–151, 2024.
- [NTN⁺21] Chi Cuong Nguyen, Giang Son Tran, Van Thi Nguyen, Jean-Christophe Burie, and Thi Phuong Nghiem. Pulmonary nodule detection based on faster R-CNN with adaptive anchor box. *IEEE Access*, 9:154740–154751, 2021.

- [NYD16] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, page 483–499, 2016.
- [ODZ⁺16] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.
- [PGM⁺19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: an imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8026–8037, 2019.
- [PHT⁺21] Priyanka Patel, Chun-Hao P. Huang, Joachim Tesch, David T. Hoffmann, Shashank Tripathi, and Michael J. Black. AGORA: Avatars in geography optimized for regression analysis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13463–13473, 2021.
- [Pri23] Simon J.D. Prince. *Understanding Deep Learning*. The MIT Press, 2023.
- [RDGF16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.
- [RF18] Joseph Redmon and Ali Farhadi. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.
- [RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.
- [RHW86] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.
- [RKH⁺21] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, page 8748–8763, 2021.

- [RLDL24] Bin Ren, Mengyuan Liu, Runwei Ding, and Hong Liu. A survey on 3D skeleton-based action recognition using learning method. *Cyborg and Bionic Systems*, 5(0100), 2024.
- [Ros63] Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. *American Journal of Psychology*, 76:705, 1963.
- [RTB17] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics*, 36(6), 2017.
- [SBB10] Leonid Sigal, Alexandru O. Balan, and Michael J. Black. HumanEva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International Journal of Computer Vision*, 87(1):4–27, 2010.
- [SCD⁺17] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *IEEE/CVF International Conference on Computer Vision*, pages 618–626, 2017.
- [SGT⁺09] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.
- [SSN⁺23] Daniel Sagmeister, Dominik Schörkhuber, Matej Nezveda, Fabian Stiedl, Maria Schimkowitzsch, and Margrit Gelautz. Transfer learning for driver pose estimation from synthetic data. In *IEEE Intelligent Vehicles Symposium*, pages 1–7, 2023.
- [SVI⁺16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, page 2818–2826, 2016.
- [SWS25] Huan Shi, Xiaopeng Wang, and Jia Shi. Fall detection algorithm using enhanced HRNet combined with YOLO. *Sensors*, 25(13):4128, 2025.
- [SXLW19] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5686–5696, 2019.
- [SXW⁺18] Xiao Sun, Bin Xiao, Fangyin Wei, Shuang Liang, and Yichen Wei. Integral human pose regression. In *European Conference on Computer Vision*, page 536–553, 2018.

- [TGA19] Hüseyin Temiz, Berk Gökberk, and Lale Akarun. Multi-view reconstruction of 3D human pose with procrustes analysis. In *International Conference on Image Processing Theory, Tools and Applications*, pages 1–5, 2019.
- [The23] The European Commission. Supplementing regulation (eu) 2019/2144 of the european parliament and of the council. <https://eur-lex.europa.eu/eli/reg/2019/2144/oj/eng>, 2023. Accessed: 2025-04-30.
- [THK⁺21] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. MLP-Mixer: An all-MLP architecture for vision. In *Advances in Neural Information Processing Systems*, volume 34, page 24261–24272, 2021.
- [TOF⁺19] Helena R. Torres, Bruno Oliveira, Jaime Fonseca, Sandro Queirós, João Borges, Nélon Rodrigues, Victor Coelho, Johannes Pallauf, José Brito, and José Mendes. Real-time human body pose estimation for in-car depth images. In *Technological Innovation for Industry and Service Systems*, page 169–182, 2019.
- [UTSS22] Ben Usman, Andrea Tagliasacchi, Kate Saenko, and Avneesh Sud. Meta-Pose: Fast 3D pose from multiple views without 3D supervision. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, page 6749–6760, 2022.
- [VCC⁺17] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio’, and Yoshua Bengio. Graph Attention Networks. In *International Conference on Learning Representations*, 2017.
- [VGO⁺20] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.
- [vMHB⁺18] Timo von Marcard, Roberto Henschel, Michael J. Black, Bodo Rosenhahn, and Gerard Pons-Moll. Recovering accurate 3D human pose in the wild using imus and a moving camera. In *European Conference on Computer Vision*, page 614–631, 2018.

- [VRM⁺17] Gul Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, page 4627–4635, 2017.
- [VSP⁺17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008, 2017.
- [WAR19] Bastian Wandt, Hanno Ackermann, and Bodo Rosenhahn. A kinematic chain space for monocular motion capture. In *European Conference on Computer Vision Workshops*, page 31–47, 2019.
- [WHG23] Fabian Windbacher, Michael Hödlmoser, and Margrit Gelautz. Single-stage 3d pose estimation of vulnerable road users using pseudo-labels. In *Image Analysis*, pages 401–417, 2023.
- [WR19] Bastian Wandt and Bodo Rosenhahn. RepNet: Weakly supervised training of an adversarial reprojection network for 3D human pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7774–7783, 2019.
- [WTZ⁺21] Jinbao Wang, Shujie Tan, Xiantong Zhen, Shuo Xu, Feng Zheng, Zhenyu He, and Ling Shao. Deep 3D human pose estimation: A review. *Computer Vision and Image Understanding*, 210(103225), 2021.
- [XLL⁺23] Wentian Xin, Ruyi Liu, Yi Liu, Yu Chen, Wenxin Yu, and Qiguang Miao. Transformer for skeleton-based action recognition: A review of recent advances. *Neurocomputing*, 537:164–186, 2023.
- [XSW⁺23] Liang Xu, Ziyang Song, Dongliang Wang, Jing Su, Zhicheng Fang, Chenjing Ding, Weihao Gan, Yichao Yan, Xin Jin, Xiaokang Yang, Wenjun Zeng, and Wei Wu. ActFormer: A gan-based transformer towards general action-conditioned 3D human motion generation. In *IEEE/CVF International Conference on Computer Vision*, pages 2228–2238, 2023.
- [YJvdS19] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 5508–5518, 2019.
- [YLJ⁺19] Zhijie Yao, Yazhou Liu, Zexuan Ji, Quansen Sun, Pongsak Lasang, and Shengmei Shen. 3D driver pose estimation based on joint 2D-3D network. In *IEEE International Conference on Image Processing*, page 2546–2550, 2019.

- [ZML⁺23] Lijuan Zhou, Xiang Meng, Zhihuan Liu, Mengqi Wu, Zhimin Gao, and Pichao Wang. Human pose-based estimation, tracking and action recognition with deep learning: A survey. *arXiv preprint arXiv:2310.13039*, 2023.
- [ZPT⁺19] Long Zhao, Xi Peng, Yu Tian, Mubbasir Kapadia, and Dimitris N. Metaxas. Semantic graph convolutional networks for 3D human pose regression. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3425–3435, 2019.
- [ZWC⁺23] Ce Zheng, Wenhan Wu, Chen Chen, Taojiannan Yang, Sijie Zhu, Ju Shen, Nasser Kehtarnavaz, and Mubarak Shah. Deep learning-based human pose estimation: A survey. *ACM Computing Surveys*, 56(1):1–37, 2023.
- [ZWT22] Weixi Zhao, Weiqiang Wang, and Yunjie Tian. GraFormer: Graph-oriented transformer for 3D pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20406–20415, 2022.
- [ZYMK16] Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. Transfer learning for low-resource neural machine translation. In *Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575, 2016.
- [ZZL⁺24] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2024.