Traffic Gesture Classification for Intelligent Vehicles

MSc. Thesis Report

Jelle Ammerlaan

TU Delft

Cover image from [64]

Traffic Gesture Classification for Intelligent Vehicles

MSc. Thesis Report

by

Jelle Ammerlaan

May 2020
to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday June 11, 2020 at 14:00.

Student number:	4311337
Project duration:	January 1, 2019 - June 11, 2020
Supervisors:	dr. F.B. Flohr
	dr. J.F.P. Kooij
Examination committee:	Prof. dr. D.M. Gavrila
	dr. ir. J.C.F. de Winter

TUDelft

Preface

Before you lies my Master thesis, the final stage of my studies in Mechanical Engineering and the accumulation of a lot of work. The final scope of this thesis is quite different from the typical mechanics and physics courses found in a Mechanical Engineering curriculum, but has allowed me to work on a very exciting and rapidly developing field that has definitely piqued my interest and captured my imagination.

Writing my final thesis proved to be one of the most challenging things of my academic career, but has taught me a lot on a personal and professional level. It was a process with ups and downs, frustrating at times when the results weren't as desired, to breakthrough moments when solutions finally worked. Looking back, I am proud of the results achieved and presented in this work.

This thesis would not have been possible without the excellent guidance and support of my supervisors, for which I would like to thank them extensively. I would like to thank Fabian Flohr especially, for all the meetings, guiding me through the research process and for always being willing to provide useful feedback on any work. I would like to thank Julian Kooij for all the guidance in academic writing and help in structuring my work in a coherent matter.

Finally, I also would like to thank my friends and family. My parents for their unconditional support, no matter the undertaking. My friends, for listening to me rambling about networks, training and pose estimation for a year.

Jelle Ammerlaan
Delft, May 2020

Abstract

Self-driving vehicles have shown rapid development in recent years and continue to move towards full autonomy. For high or full automation, self-driving vehicles will have to be able to address and solve a broad range of situations, one of which is interaction with traffic agents. For correct and save maneuvering through these situations, reliable detection of agents followed by an accurate classification of the traffic gestures used by agents is essential. This problem has received limited attention in literature to date.

The objective of this work is to establish and investigate a working traffic gesture pipeline by leveraging the latest developments in the fields of computer vision and machine learning. This work investigates and compares how well state-of-the-art methods translate to traffic gesture recognition and what application specific problems are encountered.

Multiple configurations based on skeletal features, estimated using OpenPose, and classified using recurrent neural networks (RNN) were investigated. Skeleton estimation using OpenPose and feature representations were evaluated using an action recognition dataset with motion capture ground-truth. Three RNN network architectures, varying in complexity and size, were evaluated on traffic gestures. The robustness of the developed system regarding viewpoint variation is explored, combined with the viability of transfer learning for traffic gestures. To train and validate these methods, a new traffic gesture dataset is introduced, on which an mAP of 0,70 is achieved.

The results show that the proposed methods are able to classify traffic gestures within reasonable computation time and illustrate the value of transfer learning for gesture recognition. These promising results validate the methodology used and show that this direction warrants further research.

Contents

1 Introduction

Where self-driving cars once seemed like a futuristic dream, the last couple of years they have been advancing closer to a reality every year. Breakthroughs in fields as machine learning, computer vision and artificial intelligence quickly brought automated driving of cars to such a level that independent driving on highways is a reality with the likes of Tesla's Autopilot and a broad scope of competitors alternatives.

The incentives for the development of self-driving cars are abundant. Self-diving vehicles have the potential to turn boring commutes into more useful or entertaining undertakings, by allowing the driver to work or do more entertaining activities. Furthermore, self-driving cars could solve infrastructural and environmental challenges with new usage models, such as on-demand vehicles or car sharing, which both can reduce congestion and emissions. A final and major incentive is the reduction of traffic accidents. According to the World Health Organization, yearly 1.35 million people die in traffic [61], making road traffic injuries the leading cause of death for children and young adults world-wide. Human error was estimated to account for 94% of traffic crashes in the United States [2] and 75% of crashes in the United Kingdom [57]. Clearly, removing human error from the equation would greatly reduce the loss of injuries, monetary loss and, most importantly, save human lives.

Many hurdles still have to be overcome in order to reach the highly desired full automation. For complete independent operation without a driver in the loop, the vehicle needs to be able to address all possible scenario's it encounters. One scenario that has not yet received major attention in literature and remains unsolved to date are encounters with traffic agents on worksites or intersections. An essential part of streamlined interaction between traffic agents and self-driving cars is accurate recognition of the gestures and directions of the agents. This work aims to provide a starting point for functional traffic gesture recognition for self-driving cars.

There is a broad spectrum of challenges accompanying this problem. Firstly, a common challenge for all automated driving systems is the wide range of conditions in which the system needs to function well. From bright daylight and harsh lighting to rainy, overcast conditions such as in Figure 1.1b, the system needs to be able to function in all conditions. A second challenge is the fuzzy definition of exact gestures. Gestures are very much something ingrained in cultures that humans typically classify more based on intuition than a specific rule set. The gestures that people use to interact with each other, such as 'come here', 'wave' or 'stop', are easily identified from a human perspective, but are numerically hard to define exactly from a classification point-of-view. Furthermore, the gestures used in traffic scenarios vary worldwide. Finally, and closely linked to the previous challenge, are the large variations in execution from actor to actor, as illustrated in Figure 1.1a. Gestures leave a lot a wiggle room in the spatial and temporal domain for the execution of gestures. Some agents can use short, quick movements, whereas others use large, slow movements, while both gestures convey the same message.

Fortunately the problem of traffic gesture recognition has large overlap with the existing fields of computer vision, machine learning, action and gesture recognition. Literature from these rapidly developing fields provide existing and established techniques which can be applied to tackle this problem. Much work has been done in the fields of action and gesture recognition on the classification of human movements, especially with skeleton data and depth images extracted using the Kinect camera from Microsoft. The estimation of human poses in images is now possible with works such as OpenPose from Cao et al. [10]. The field of machine learning has been growing especially fast and offers a wide range of methods to tackle classification problems.

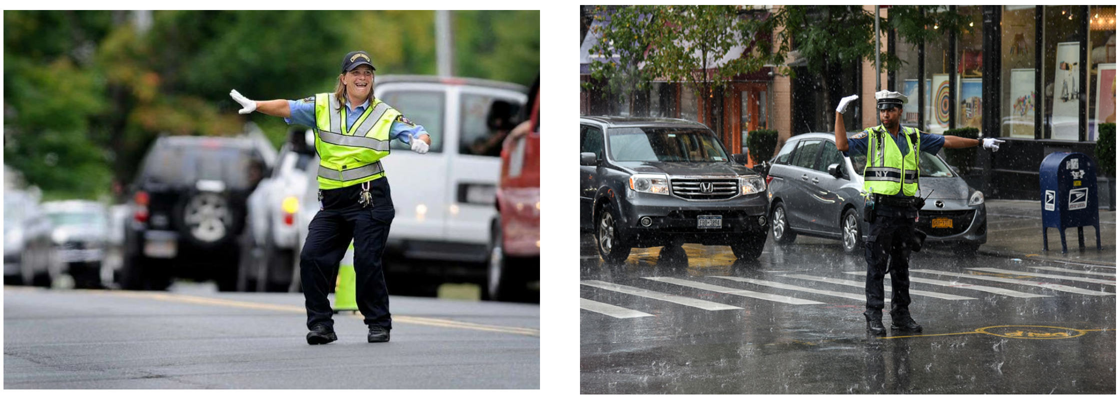

(a) (b)

Figure 1.1: Examples of challenging encounters. While the officer in (a) would brighten the day of most commuters, the cheerful gesture style makes classification more difficult. Poor weather conditions in (b) create detection challenges. From [15] [49]

Seemingly, the technologies to tackle traffic gesture recognition are now available. The goal of this work is to create the first exploratory research and investigate how well these existing methods transfer over to traffic gesture recognition. The research questions posed in this work are:

- How close can OpenPose based action recognition come in performance to accurate motion capture data? How much can skeletal feature representation improve the performance of OpenPose detections?
- How well do state-of-the-art action recognition methods perform on the problem of traffic gesture recognition?
- How well do the methods and features deal with viewpoint variations? Does transfer learning work for viewpoint variations?
- Is the performance of the entire end-to-end system fast enough for real-world performance?

Answering these research questions will require implementing and evaluating the entirety of the classification pipeline, from feature extraction to classification. This is done across several experiments, on both existing datasets from literature and a new traffic gesture specific dataset. The performance of 2D human pose estimation with OpenPose as feature for classification is evaluated against motion capture data using the Berkeley MHAD dataset of Ofli et al. [51]. Additionally, using the novel, multi-viewpoint traffic gesture dataset, called Greenscreen Police Gesture Dataset [3], classification performance and robustness are evaluated in an application specific setting. By answering these questions, the goal is to create a good starting point for further research into traffic gesture recognition and hopefully contribute to eventually fully automated self-driving cars.

This thesis will first provide further background information and a literature review in Chapter 2, including the contributions of this work with respect to the existing literature. Chapter 3 will outline the methods used in the experiments. Chapter 4 introduces the Greenscreen Police Gesture Dataset and describes its set-up and label design. Chapter 5 will evaluate the introduced methods using Greenscreen Police Gesture Dataset and Berkeley MHAD. The results will be discussed further in Chapter 6, followed by finally the conclusions and recommendations for further work in Chapter 7.

2 Background and literature

2.1. Previous work in traffic gesture recognition

As mentioned in the introduction, traffic gesture recognition has not received major attention in literature. Some work has been done using wearable sensors such as accelerometers to classify traffic gestures by Yuan and Wang [62, 70]. While these works show good results, they used rule-based classification which limits the flexibility of the classifier and evaluated using a small set of test gestures. Furthermore, this approach requires special hardware, which creates a new set of practical problems and limits functionality of the gesture recognition to those equipped with this special hardware. Ideally, classification works using purely car based sensors.

Vision based approaches have been explored, for example the works of Guo et al. [20] and Cai and Guo [8], which are based on the upper body limb angles of agents. They use foreground extraction and try to optimally cover the extracted body with limbs according to a body plan. Classification is done using joint angles and Gabor features (image filters used for texture analysis) with a nearest neighbour classifier. This approach has clear drawbacks: it is reliant on the colour of the visibility jacket for foreground extraction and the method used for pose estimation suffers greatly from viewpoint variations. Nevertheless, there are clear benefits to a vision based approach as it is not reliant on expensive hardware solutions and can use the cars onboard video cameras.

Advances in the fields of computer vision and machine learning have developed new state-of-the-art methods that can be used to address these issues with vision based gesture recognition. Applications of these methods in the domain of traffic gesture recognition has been limited. One of the first works that uses modern machine learning methods in this domain is the work of He et al. [26], which was published simultaneously during the development of this work. Their recognition pipeline is based on joint detection using a modified convolutional pose machine and skeletal features. Classification is done using a single layer LSTM recurrent neural network. Evaluation was done on a custom dataset consisting of in total 2 hours of police gestures.

While the overall approach of this work is very similar to the work of He et al. [26], there are some stark differences. This work provides more in-depth evaluations of the options in the pipeline considering feature representation and network architecture. Especially the networks evaluated in this work are much larger that the network used by He et al. [26]. Furthermore, while their dataset consists only of gesture aimed at the camera of 8 mutually exclusive classes, of which the definitions are defined in Figure 2.1, whereas the Greenscreen Police Gesture Dataset [3] introduced in Section 5.2.2 included viewpoint variations and a continuous flow of several gestures simultaneously.

The work of He et al. [26] shows promising results, both in accuracy and speed, and support the viability of the design choices made in this work. This chapter will describe how this work is positioned in the literature fields of pose estimation and skeleton based action and gesture recognition. Furthermore the definitions of traffic gestures and existing datasets are explored.

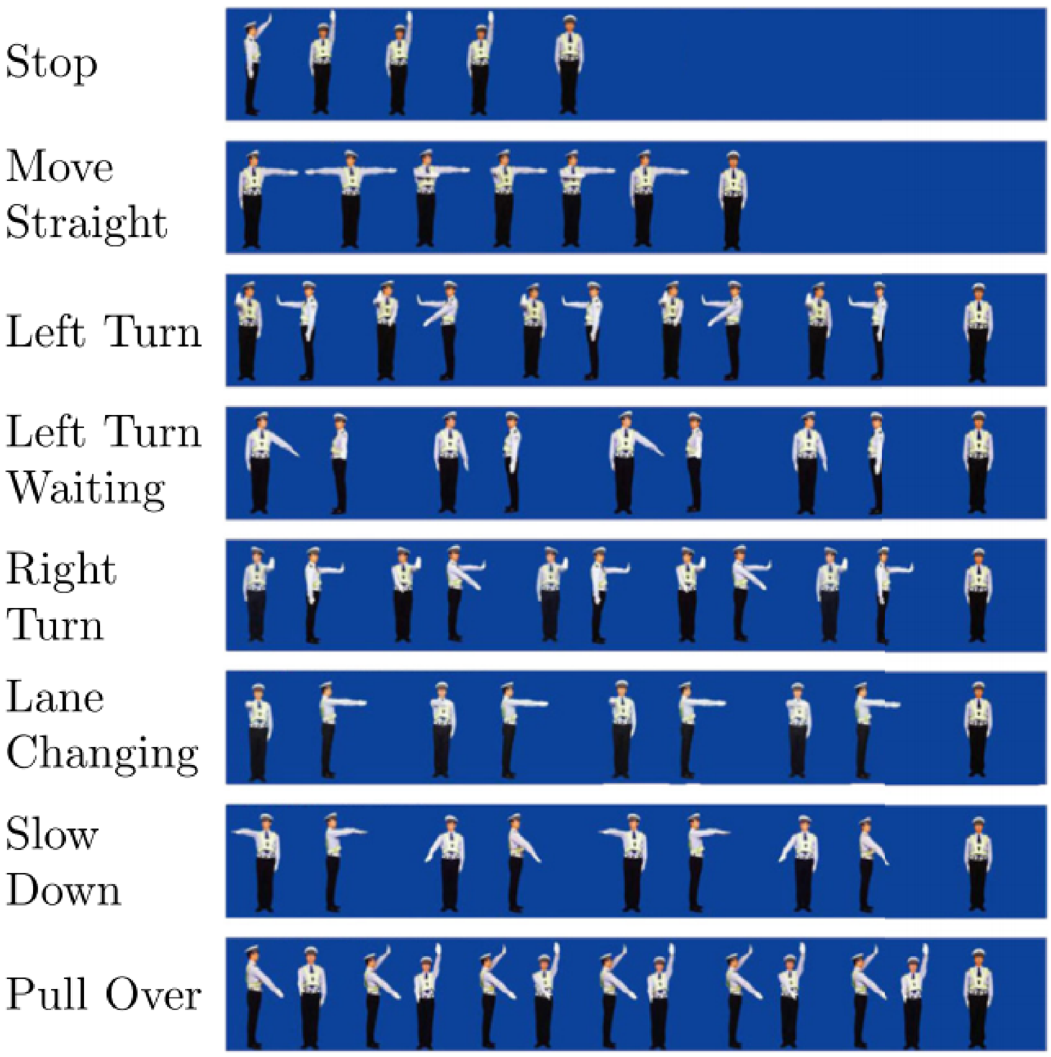


Figure 2.1: Chinese traffic gesture execution definition from He et al. [26].

2.2. Traffic gesture definitions

Internationally, clear differences in the execution of traffic gestures can be observed. Gupta et al. [21] tried to find an universal language in the gestures of traffic control officers, for the specific use-case of intelligent vehicles. Their approach was to create an overview of gestures used across the globe (USA, UK, Australia, Germany, India and China) and look for similarities or differences. By looking up legislative guidelines the expansive overview presented in Figure 2.2a and 2.2b was created. Their findings were that while there are large amounts of overlap in the gestures, there are no universal rules and differences exist. An example of differences in gesture definition is the stop command; generally this entails raising one of the arms up towards the vehicle and extending the other. However, the German gesture for stopping entails to extend both arms.

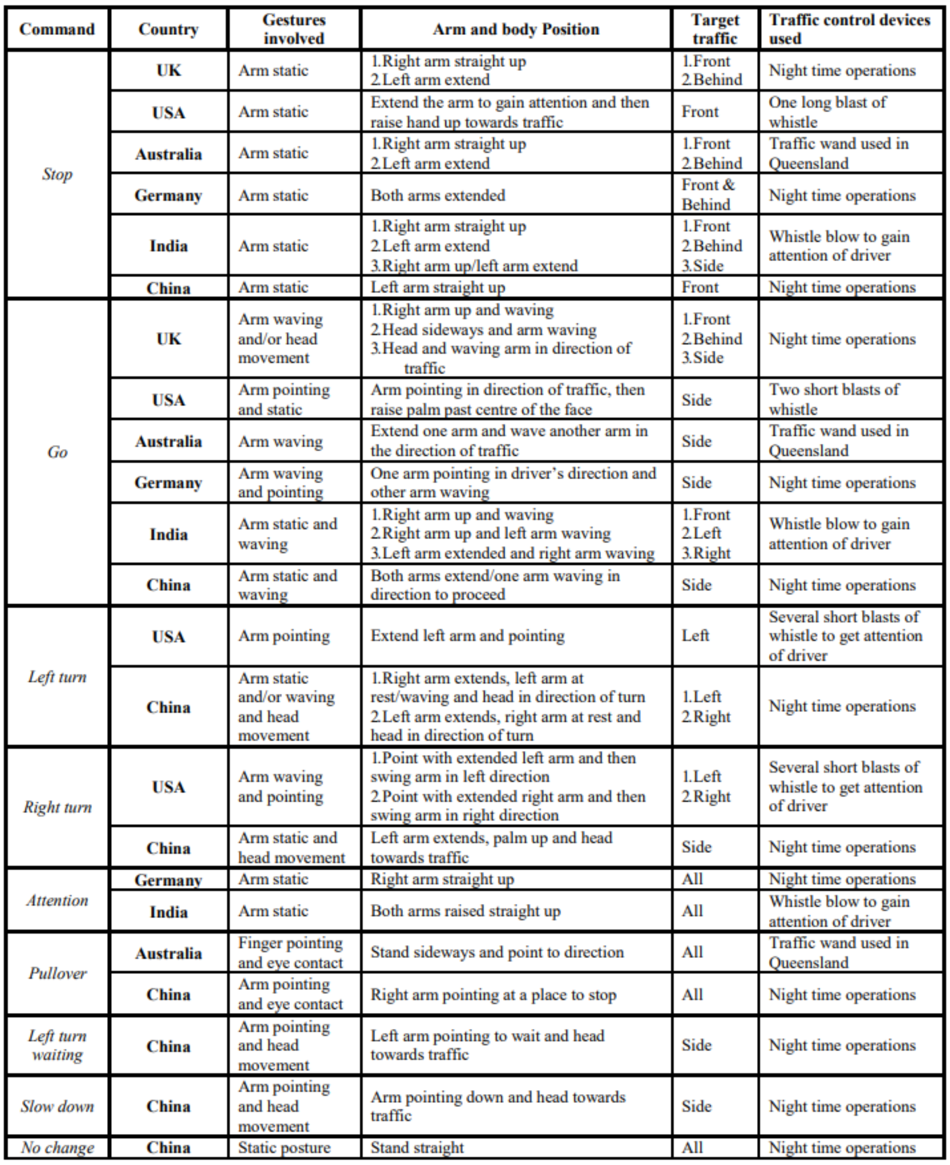
Looking at the gestures found in the overview from Gupta et al. in Figure 2.2, the following instructions can be summarized:

- Stop the vehicle
- Start the vehicle and/or move in a certain direction
- Make a turn (left or right)
- Slow down

with several other options available in some regions. The gestures listed are the main gestures used and thus most essential for autonomous driving. Further analysis of these gestures and their execution results in three other categories to distinguish gestures:

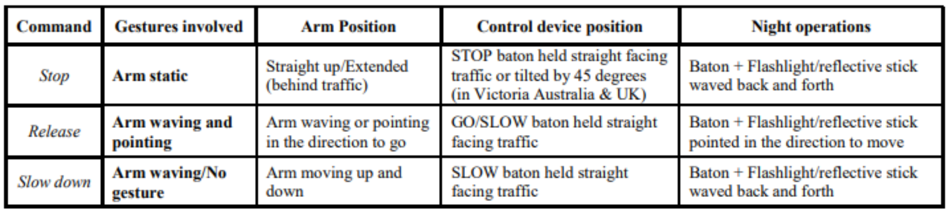
- Static gestures, the agent maintains a certain pose
- Dynamic gestures, the agent executes a certain movement
- Car-specific gestures, the agent gestures to a specific car (can be both static and dynamic movements)

Dynamic gestures are expected to be harder to classify due to the larger variation in temporal and spatial execution. Waving movements leave more room for agent to agent variation in speed and execution style compared to the static poses. Car-specific gestures also pose the challenge of linking an instruction to a specific vehicle, which requires accurate identification of the intended vehicle. Datasets used in this work lack car specific gestures and are limited to static and dynamic gestures.



Command	Country	Gestures involved	Arm and body Position	Target traffic	Traffic control devices used
Stop	**UK**	Arm static	1.Right arm straight up 2.Left arm extend	1.Front 2.Behind	Night time operations
	USA	Arm static	Extend the arm to gain attention and then raise hand up towards traffic	Front	One long blast of whistle
	Australia	Arm static	1.Right arm straight up 2.Left arm extend	1.Front 2.Behind	Traffic wand used in Queensland
	Germany	Arm static	Both arms extended	Front & Behind	Night time operations
	India	Arm static	1.Right arm straight up 2.Left arm extend 3.Right arm up/left arm extend	1.Front 2.Behind 3.Side	Whistle blow to gain attention of driver
	China	Arm static	Left arm straight up	Front	Night time operations
Go	**UK**	Arm waving and/or head movement	1.Right arm up and waving 2.Head sideways and arm waving 3.Head and waving arm in direction of traffic	1.Front 2.Behind 3.Side	Night time operations
	USA	Arm pointing and static	Arm pointing in direction of traffic, then raise palm past centre of the face	Side	Two short blasts of whistle
	Australia	Arm waving	Extend one arm and wave another arm in the direction of traffic	Side	Traffic wand used in Queensland
	Germany	Arm waving and pointing	One arm pointing in driver's direction and other arm waving	Side	Night time operations
	India	Arm static and waving	1.Right arm up and waving 2.Right arm up and left arm waving 3.Left arm extended and right arm waving	1.Front 2.Left 3.Right	Whistle blow to gain attention of driver
	China	Arm static and waving	Both arms extend/one arm waving in direction to proceed	Side	Night time operations
Left turn	**USA**	Arm pointing	Extend left arm and pointing	Left	Several short blasts of whistle to get attention of driver
	China	Arm static and/or waving and head movement	1.Right arm extends, left arm at rest/waving and head in direction of turn 2.Left arm extends, right arm at rest and head in direction of turn	1.Left 2.Right	Night time operations
Right turn	**USA**	Arm waving and pointing	1.Point with extended left arm and then swing arm in left direction 2.Point with extended right arm and then swing arm in right direction	1.Left 2.Right	Several short blasts of whistle to get attention of driver
	China	Arm static and head movement	Left arm extends, palm up and head towards traffic	Side	Night time operations
Attention	**Germany**	Arm static	Right arm straight up	All	Night time operations
	India	Arm static	Both arms raised straight up	All	Whistle blow to gain attention of driver
Pullover	**Australia**	Finger pointing and eye contact	Stand sideways and point to direction	All	Traffic wand used in Queensland
	China	Arm pointing and eye contact	Right arm pointing at a place to stop	All	Night time operations
Left turn waiting	**China**	Arm pointing and head movement	Left arm pointing to wait and head towards traffic	Side	Night time operations
Slow down	**China**	Arm pointing and head movement	Arm pointing down and head towards traffic	Side	Night time operations
No change	**China**	Static posture	Stand straight	All	Night time operations

(a)



Command	Gestures involved	Arm Position	Control device position	Night operations
Stop	**Arm static**	Straight up/Extended (behind traffic)	STOP baton held straight facing traffic or tilted by 45 degrees (in Victoria Australia & UK)	Baton + Flashlight/reflective stick waved back and forth
Release	**Arm waving and pointing**	Arm waving or pointing in the direction to go	GO/SLOW baton held straight facing traffic	Baton + Flashlight/reflective stick pointed in the direction to move
Slow down	**Arm waving/No gesture**	Arm moving up and down	SLOW baton held straight facing traffic	Baton + Flashlight/reflective stick waved back and forth

(b)

Figure 2.2: An overview of gestures found for intersection (2.2a) and worksite scenarios (2.2b) by Gupta et al. [21]

2.3. Skeleton based action and gesture recognition

Gesture recognition can be considered a subset of the larger field of action recognition. Early attempts used image based features to classify actions such as motion-energy images and motion-history images in the work of Bobick and Davis [7], the Space-Time volumes of Yilmaz and Shah [69] or the space-time interest points of Laptev [42]. However, the common shortcoming of all these feature selections is a lack of robustness to appearance variations and vulnerability to viewpoint changes. Skeletal features offer an attractive alternative.

Johansson [34] conducted the first research on skeleton based action recognition back in the 1970s, by showing test subjects human movements where only bright spots on limbs were visible. The locations of the points proved to be sufficient to accurately classify human actions. The use of skeletal features was validated again in the more recent work of Yao et al. [68], where they found that pose based classification outperformed appearance features substantially. Furthermore, fusion of these features proved to yield no advantage, suggesting that skeletal features succesfully capture the same information as appearance based features.

Historically, research into skeleton based action recognition has exploded due to the release of the Microsoft Kinect depth sensor, which allowed for the accessible creation of large depth and 3D skeleton based action datasets. Examples of which are the Chalearn gesture dataset [22], NTU-RGB+D of Liu et al. [45], UT Kinect of Xia et al. [65] and SBU Kinect Interaction of Yun et al. [71].

Figure 2.3: Sample frames from NTU-RGD+D [45]

Current state-of-the-art on these datasets is based on deep learning methods suchs as recurrent neural networks and convolutional neural networks. The latter approach usually relies on transforming a skeleton sequence to an image, such as the work of Kim and Reiter [36]. In contrast, recurrent neural networks are able to model temporal sequence by nature. Use of long short term memory (LSTM), introduced by Hochreiter and Schmidhuber [30], is considered standard, which helps substantially to stabilize training and model temporal relations over longer distances. Use of LSTM networks has been widespread in several fields that use sequence modelling such as natural language modeling and speech recognition. Gated Recurrent Units (GRU) of Cho et al. [12] were introduced more recently and aim to provide the same benefits as LSTM at a reduced model complexity. Wang et al. [63] showed in a broad comparison that there is no single approach that works best and performance varies per dataset. Generally, RNN based methods worked best for continuous motion recognition (such as real-time traffic gesture recognition), whereas CNN approaches seemed to have a slight edge on large datasets such as NTU-RGB+D [45]. Research in this work has been focused on recurrent neural network based classification, as they provide a simple yet versatile solution to classify gesture sequences, traditionally have shown good results and translate well to continuous gesture recognition.

A common trend in the works that use the recurrent approach is the question of how to model the relations between joints. Early works simply fed joint coordinates into the network, and relied on the network to learn the relations. However, it became apparent that explicitly describing the relations of the limbs and joints to the network could yield increases in accuracy. Du et al. [16] introduced hierarchical recurrent neural networks, in which limbs are fed separately to subnets which are later merged, as can be seen in Figure 2.4. However, this approach is said to fail in modelling the relations between joints in separate limbs accurately.

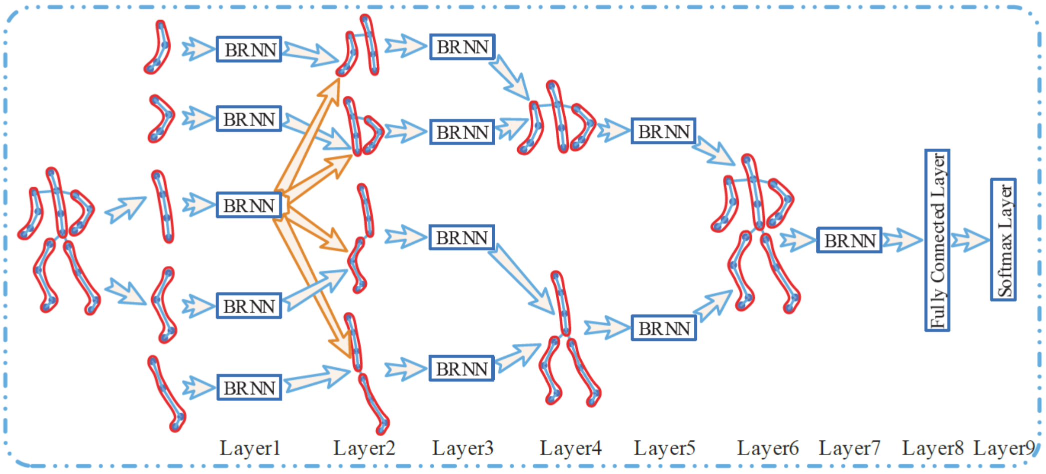


Figure 2.4: Illustration of the network proposed by Du et al. [16]. Only Layer 7 contains LSTM cells.

In contrast, Zhu et al. [73] used a simple three layered LSTM network and relied on a novel regularization scheme to learn the co-occurence of the skeletal joints. Their results consistently outperformed the work of Du et al. [16].

Finally, Zhang et al. [72] explored a wide range of geometric feature representations, seen in Figure 2.5, for skeleton based action recognition and outperformed both Du et al. [16] and Zhu et al. [73]. Based on their results, they conclude that using the right skeleton representation combined with using a simple 3-layer LSTM network achieves state-of-the-art results while additionally requiring less training data. Clearly, using the right skeletal representation can be essential. Integration of these relations on the feature level is the simplest approach and has shown the best results.

Maghoumi and LaViola [46] presented DeepGRU (Deep Gesture Recognition Utility), which aims to offer a robust and easily trainable gesture recognition network. As the name suggests, they use GRU units to reduce the parameter size of the network, combined with an attention

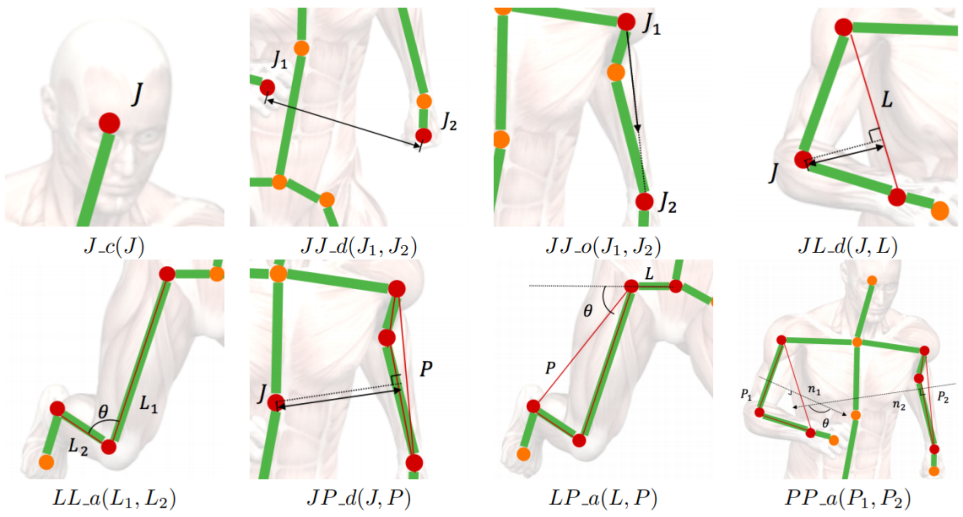


Figure 2.5: All geometric features explored by Zhang et al. [72]. Joint-line distance (JL_D) achieved the best results consistently.

module, which helps in focusing relevant temporal information to the classifier. The results they achieve are state-of-the art on a variety of datasets.

In conclusion, the field of skeleton based action recognition has advanced rapidly and significantly in recent years, and continues to do so. The proven methodologies of the works mentioned here can provide building-blocks to solve traffic gesture recognition.

2.4. Human pose estimation

The estimation of human poses (position and orientation of the skeleton) has been widely used for human computer interaction interfaces such as gaming, sport analysis, device control and many others. The previously mentioned Kinect camera is a prime example of these applications and relied on depth maps. However, these depth maps are obtained indoors only 2-4 meters from the camera, whereas self-driving vehicles will face interactions at much longer ranges in outdoor conditions. Therefore, pose estimation will most likely have to rely on RGB video feeds.

Use of pose estimation is not novel in the automotive domain. Pose information has been used to predict pedestrian intentions and movements in the work of Ghori et al. [18]. Fang et al [17] used 2D pose estimation to predict whether pedestrians are going to cross a road. Additionally, pose orientation also has been used for automotive applications by Kooij et al. [39] and Roth et al. [54].

Like most computer vision fields, current state-of-the-art pose estimation methods use deep learning approaches. Dang et al. [14] provide a survey of current advances based on deep learning in the field of pose estimation. They identify the approaches found in literature as the taxonomy overview found in Figure 2.6. Single person pipelines typically use either direct regression of the keypoints from feature maps or first generate heatmaps before predicting keypoints. Multi-person pipelines can also be split in two groups based on methodology: top-down approaches first use human detection after which a single-person pipeline is used, where bottom-up approaches start by locating all keypoints in an image, after which they are linked to the right persons.

Each of these configurations have their own advantages and disadvantages in regards to speed, scaling and ease of training, which Dang et al. [14] discuss in detail. Typically, bottom-up approaches scale better with more people visible in frame, as keypoint detection happens all at once.

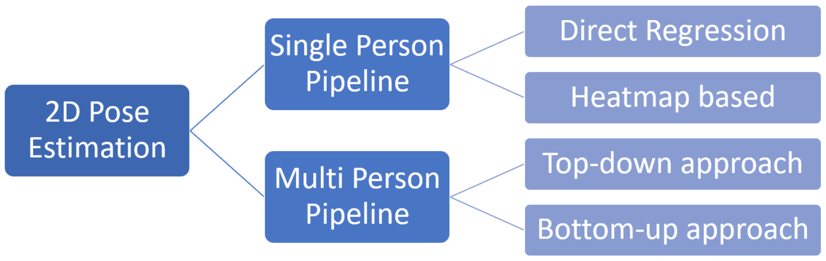


Figure 2.6: Structure of 2D human pose estimation approaches as identified by Dang et al. [14]

Selection criteria for traffic gesture recognition are accurate, fast and robust pose estimation. Most pose estimation methods rely on very large convolutional networks and as such are not fast enough for real-time pose estimation. The works of Cao et al. [9, 10] provided a real-time pose estimation solution that won the inaugural 2016 COCO keypoints challenge [43] and significantly outperformed the state-of-the-art on the MPII Multi-Person benchmark [4]. Their work can be classified as a multi-person pipeline using a bottom-up approach. Additionally, they released their work as the OpenPose framework with continued development and support. An example of human pose estimation using OpenPose can be seen in Figure 2.7. For the reasons mentioned above, it is a prime candidate for traffic gesture recognition.

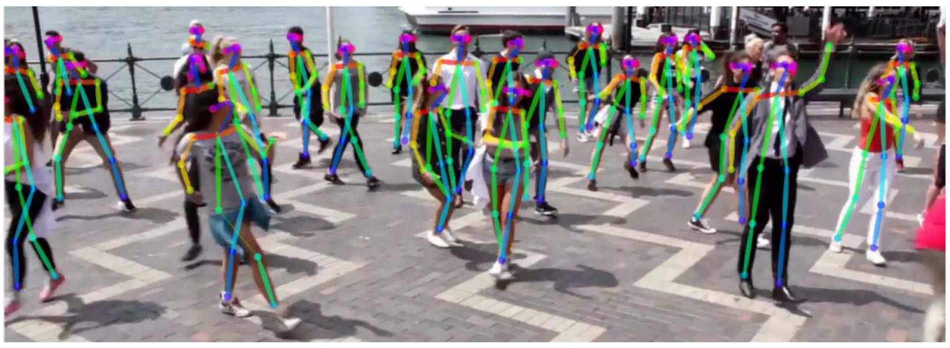


Figure 2.7: Sample image of human pose estimation using OpenPose of Cao et al. [10] in crowds.

Recently, Kreiss et al. [40] published a method aimed to specifically address robust pose estimation for the automotive domain called PifPaf, following similar design choices as OpenPose. Their results show accurate and robust classification in very dynamic environments, highlighting the possibilities and developments in the field of human pose estimation.

All methods mentioned up to now are limited to 2D human poses. The estimation of human poses in 3D space has also been a field of rapid research and developments with many interesting works [47, 48, 53]. Martinez et al. [47] even achieved real-time 3D pose estimation. Unfortunately, current results are not yet robust and accurate enough for use in real-world automotive application.

In this work, all pose estimation will be done using OpenPose. The arguments behind this choice are the accurate and fast estimation performance, combined with the availability of a supported code framework.

Using OpenPose for action recognition has been done before by several works [58, 66, 67] to test skeleton based classification methods on datasets that lacked skeletal labels. Angelini et al. [5] proposed a action recognition system based on OpenPose and is one of the few works that focuses on 2D skeletons specifically, compared to the more commonly used 3D skeletons. However, no works in literature have conducted a detailed analysis of the performance of OpenPose skeleton estimation combined with action recognition, to our knowledge. Additionally, insights on 2D versus 3D skeleton based action recognition are sparse in literature. In this work we will conduct experiments specifically set-up for this purpose.

2.5. Datasets

Based on the research questions posed in Chapter 1, the right datasets have to be selected to conduct the experiments needed to answer these questions. From the questions, the need arises for two separate datasets: a dataset consisting of traffic gestures and a dataset with accurate skeleton annotations and established state-of-the-art baselines and results.

Section 2.3 already mentioned some action recognition datasets currently prevalent in literature. However, the actions present in these database are broad human actions such as jump or wave, instead of traffic gestures. Section 2.1 outlined previous work specific to traffic gestures. However, up until recently none of these works had released their data. He et al. [26] were the first, to our knowledge, to release a traffic gesture dataset based on Chinese traffic gestures (seen in Figure 2.1 and 2.8). As mentioned, this work was released when this work was already in development and too late to be included in experiments. This work introduces an alternative traffic gesture dataset titled Greenscreen Police Gesture Dataset [3], which will be used to conduct experiments specific to traffic gestures.

However, as the Greenscreen Police Gesture Dataset lacks any form of skeletal ground-truth data, we have to look elsewhere to analyse performance of OpenPose for action recognition. Luckily, we can look to action recognition datasets for alternatives. Obdržálek et al. [50] investigated the accuracy and robustness of the Kinect sensor compared to a motion capture set-up. Their findings were that the Kinect performs accurate for controlled simple body postures, but has a variability of about 10 cm for general poses and struggles with occlusions. The conclusion they drew was that the Kinect sensor is useful to determine trends in movements, but lacks the accuracy for quantitative analysis. Kinect based datasets are thus a poor

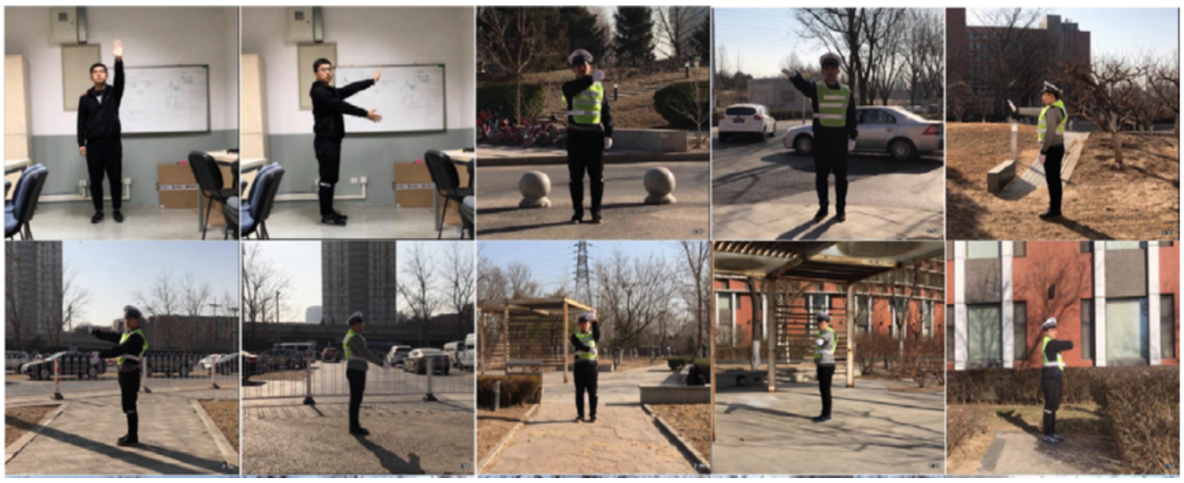


Figure 2.8: Frames from the recordings of the traffic gesture dataset of He et al. [26].

comparison for OpenPose based action recognitions. A much more accurate alternative way of extracting skeleton data is using motion-capture suits.

A dataset recorded in conditions close to real-world driving conditions would be ideal for the evaluation of robustness. However, this creates a contradiction with the requirement of motion-capture skeletons, as such set-ups (as well as Kinect datasets) are limited to studio environments. Therefore, studio recordings will have to suffice. The Berkeley Multimodal Human Action Database (MHAD) of Ofli et al. [51] seems a prime candidate. It's action range is of the desired size and the types of actions are relevant. It is recorded with both an accurate motion-capture system and a wide range of video camera's from a variety of viewpoints, including depth images. Additionally, its size is adequate for training RNN's as showed by Du et al. [16] and Zhu et al. [73], yet too large to become computationally expensive. These works also provide us with baselines to compare against.

There are some caveats present with using Berkeley MHAD. There is no clear evaluation protocol presented by the authors. They suggest training on 7 of the subjects and testing on the other 5, however in literature variations on this can be found, making direct comparisons to those works difficult or meaningless. There is also an absence of some form of validation set, something that is desirable and generally good practice. Also, considering both Du et al. [16] and Zhu et al. [73] managed to achieve 100% test accuracy based on the skeletal data relatively simply, one could argue that the dataset is not very challenging compared to the larger action recognition datasets, such as NTU-RGB+D of Liu et al. [45]. These caveats are something to keep in mind and account for when working with this dataset, but none of them are considered deal breakers. Additionally, Berkeley MHAD is one of the few action datasets that contains both motion-capture data and the image sequences needed for OpenPose estimation. Therefore it is selected as dataset to evaluate OpenPose.

2.6. Contributions

Compared to the existing literature outlined, this work presents the following contributions:

- Provide one of the first detailed explorations of traffic gesture classification using deep learning methods
- Evaluation of the use of OpenPose for 2D skeleton estimation and analysis of the importance of skeleton feature representation for action and gesture recognition
- Label design and creation of a custom label tool for the novel Greenscreen Police Gesture Dataset
- Evaluation and implementation of state-of-the-art RNN classification methods on traffic gestures and extending their use to a multi-label set-up

Chapter 3 will further explore the selected methods in detail.

3 Methods

Image based gesture recognition relies on the combination of various components to go from raw image sequences to gesture predictions. Each component comes with a range of design choices and parameters, which have to be chosen and where possible optimized to achieve the best performance possible. This chapter will discuss the theoretical frameworks and reasoning behind the selected and compared methods. The performance of these methods will be compared in a range of experiments outlined in Chapter 5.

In this work, the problem of gesture recognition is approached as a machine learning classification problem. This approach can be summarized as follows: given a sample gesture i, described by a sequence of feature vectors $\mathbf{x}^i = (\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_{t-1}, \mathbf{x}_t)$ the goal is to perform a prediction ($\hat{\mathbf{y}}^i$) of the correct label ($\mathbf{y}^i$). The design choices available are what features to use, feature representation and how to classify these features to predictions.

This chapter is structured in a similar order as the gesture recognition pipeline, depicted in Figure 3.1. First, Section 3.1 will discuss skeleton estimation with OpenPose. The feature representation of these skeletons and how they are fed to the classifiers is discussed in Section 3.2. Classification with recurrent neural networks and the architectures used will be discussed in Section 3.4, where Section 3.5 will elaborate on the details of the classification step. The exact implementation with relevant parameters and experiment set-up will be presented in Chapter 5.

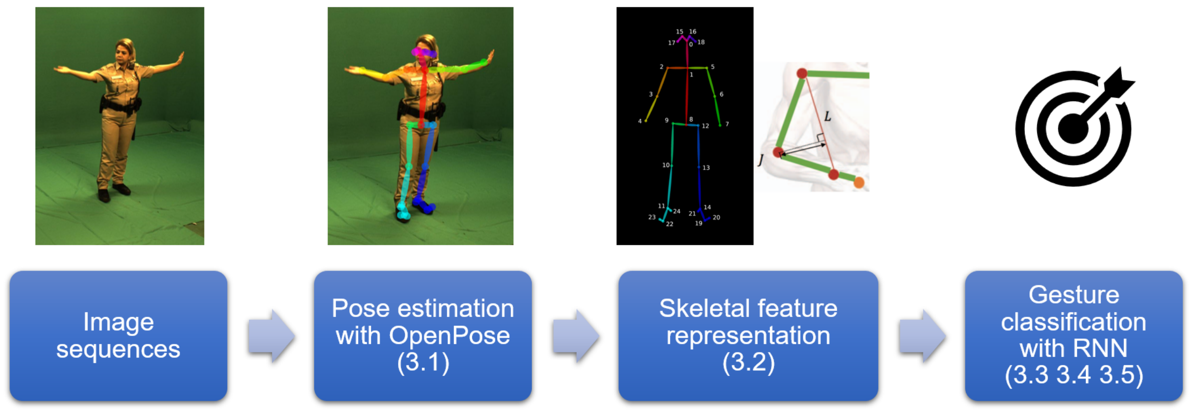


Image sequences
Pose estimation with OpenPose (3.1)
Skeletal feature representation (3.2)
Gesture classification with RNN (3.3 3.4 3.5)

Figure 3.1: Overview of the methods applied during gesture recognition and the corresponding Sections

3.1. Pose estimation with OpenPose

2D pose estimation from images is done using the OpenPose framework of Cao et al. [10]. This pipeline is visualized in Figure 3.2. OpenPose uses a bottom-up strategy where both body parts and their associated limbs are detected in an image. It uses a two-branch multi-stage convolutional neural network to jointly predict joint heatmaps and the so called Part Affinity Fields, which are vector fields that describe the connection to other joints in the image. Given the parts and their PAFs, individual skeletons are then constructed using Bipartite graph matching, which ensures matching of the right joints to the right person in crowds.

OpenPose based pose estimation is done on an image-by-image basis, without any form of tracking. For the skeleton output, the BODY_25 format configuration of OpenPose is used, seen in Figure 3.3. Part locations are returned as pixel locations scaled with respect to the original image size.

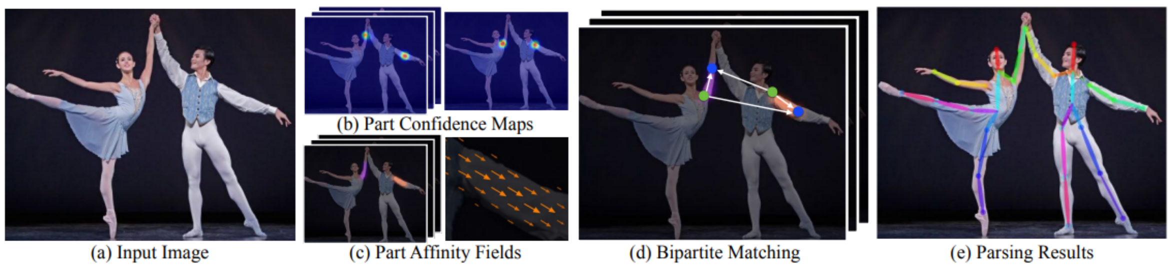


Figure 3.2: OpenPose pipeline from Cao et al. [10]. Given an input image (a), a CNN jointly predicts (b) Part Confidence Maps and (c) Part Affinity Fields. Body parts are matched using Bipartite matching (d) to construct the final poses for every person in the image (e).

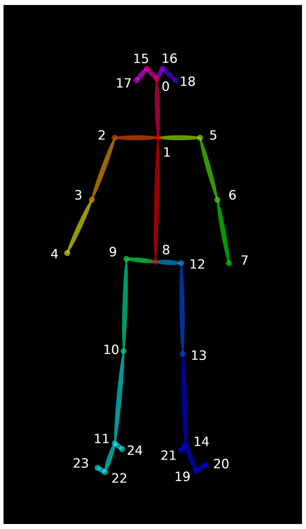


Figure 3.3: The BODY_25 skeleton format used by OpenPose

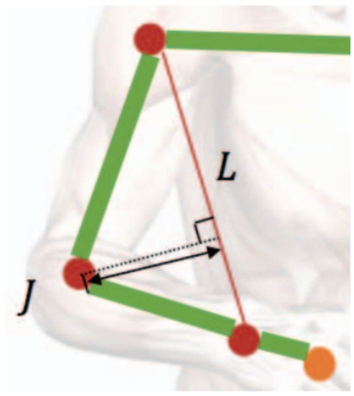


Figure 3.4: Joint-Line distance feature representation from Zhang et al. [72]

3.2. Skeletal representation

The skeletal features are fed as input to the networks in two ways. The first method simply feeds the the coordinates as a vector $\mathbf{x} \in \mathbb{R}^{N \times L}$, where $N = 25 \times 2 = 50$ for the BODY_25 skeleton format of OpenPose and L is the length of the sequence in frames. This approach leaves it up to the network to learn the relations between the joints.

The second approach follows the philosophy of creating hand-crafted features, in order to describe the relations of the joints to the network. For this, the joint-line distance method from Zhang et al. [72] is used. Joint-line distance was chosen because it showed the best performance from the representations explored by Zhang et al. [72] and is expected to translate well to application on 2D skeletons. From a given skeleton, a selection of lines is created based on the following three constraints:

1. J_1 and J_2 are directly adjacent in the kinetic chain
2. If either J_1 or J_2 are at the end of a skeleton chain (such as hands, feet or head), the second joint used for the line can be two steps away in the kinetic chain
3. If J_1 and J_2 are both at the end of a limb, $L_{J_1 \to J_2}$ is a line

For every joint in the skeleton, the distance to all lines is calculated, visualized in Figure 3.4. A visual representations of the lines given a sample skeleton is given in Figure 3.5. The dimension of this vector is logically dependent on the number of joints in the skeleton present. For the BODY_25 skeleton format from OpenPose, this gives us a feature vector of length 1012, which makes the input of the network $\mathbf{x} \in \mathbb{R}^{1012 \times L}$, where L is again the sequence length. For reference, the BVH skeleton of Berkeley MHAD used in Section 5.3.1 would have given a vector of dimension $\mathbf{x} \in \mathbb{R}^{1551 \times L}$.

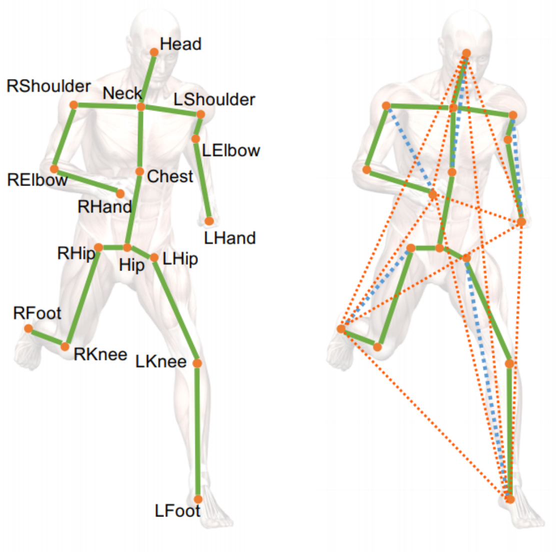


Figure 3.5: Example of line construction for a given skeleton from Zhang et al. [72]. Shown left is a skeleton model (note that this 16 joint example is different from the BODY_25 model) and right the different types of lines constructed for that model. Labeled by colour the lines represent the three constraints used (green: 1, blue: 2, orange: 3).

For both approaches, normalization of the skeleton is important. The exact implementation of this step is dependent on what skeleton format the dataset uses and the exact implementations are discussed in detail in Section 5.2. For both datasets, centering of the skeleton is important to ensure that the skeletal features are independent of their location in the frame. This is done by centering the skeleton around the average of the three hip joints as seen in Equation 3.1.

$$O = \frac{J_{\text{hip left}} + J_{\text{hip center}} + J_{\text{hip right}}}{3} \tag{3.1}$$

Additionally, scaling of the skeleton is important to ensure skeletal features are independent of variations such as agent size or distance to the camera. The spine length is used as scaling factor, as it is least vulnerable to estimation fluctuations and least susceptible to occlusions.

3.3. Recurrent neural network variations

Recurrent neural networks (RNN) are an expansion of simple feedforward networks by introducing feedback loops between the hidden states of previous timesteps. This structure allows RNNs to model temporal information of sequences of variable length and make it very powerful for a wide range of applications.

Given an input sequence $\mathbf{x} = (\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_{t-1}, \mathbf{x}_t)$, hidden states $\mathbf{h} = (\mathbf{h}_0, \mathbf{h}_1, ..., \mathbf{h}_{t-1}, \mathbf{h}_t)$ and outputs $\hat{\mathbf{y}} = (\hat{\mathbf{y}}_0, \hat{\mathbf{y}}_1, ..., \hat{\mathbf{y}}_{t-1}, \hat{\mathbf{y}}_t)$ the general structure of a single layer RNN can be given by the following equations:

$$\mathbf{h}_t = f\Big(\mathbf{W}_x \mathbf{x}_t + \mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{b}_h\Big) \tag{3.2}$$

$$\hat{\mathbf{y}}_t = f\Big(\mathbf{W}_{ho} \mathbf{h}_t + \mathbf{b}_{ho}\Big) \tag{3.3}$$

Where $\mathbf{W}_x$, $\mathbf{W}_h$ and $\mathbf{W}_{ho}$ are the connection weights between the input and hidden layer, hidden layer and hidden layer of the previous timestep and hidden layer and output layer. $\mathbf{b}_h$ and $\mathbf{b}_{ho}$ are the bias terms and f denotes the activation function, which in RNNs is typically the hyperbolic tangent function, as unbounded activation functions such as ReLu tend to result in exploding gradients.

Due to the recurrent nature of RNNs, they are able to handle a wide range of temporal inputs and outputs. Figure 3.6 illustrates the possible temporal configurations. This work is limited to the many-to-one approach which is most conventional in the field of action recognition, where a many-to-many approach is also possible in this scenario. This is done as this allows for simpler and more parallel training configurations and the main interests is the ability to accurately label a gesture, given the previous frames. Many-to-one networks can also generate a sequence of outputs, by using a sliding window over the input sequence, as will be done in Section 5.3.4.

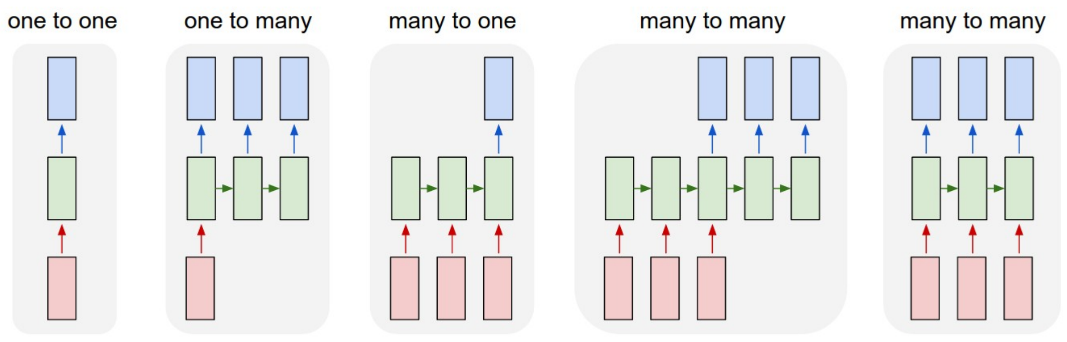


Figure 3.6: Possible temporal configurations for RNNs from Karpathy [35]. One-to-one is a traditional feedforward network without the recurrent step. One-to-many generates a sequence from a single input, such as image captioning. Many-to-one handles a sequence input and has a single output, such as sentiment analysis in text or action/gesture classification. Many-to-many that converts sequences to sequences of not necessarily the same length, such as in translation. Finally, many-to-many that has synced sequence input and output, such as in frame labeling.

RNNs traditionally suffer from the vanishing gradient problem, as explored by Hochreiter [29] and Bengio et al. [6]. This causes the network to struggle with modelling long term dependencies and makes it difficult and unstable to train. Expanding the RNN structure of Equations 3.2 and 3.3 with the so called Long Short Term Memory (LSTM) was first suggested by Hochreiter and Schmidhuber [30]. The function of LSTM can be summarized in Equations 3.4 through 3.9 and Figure 3.7a and 3.7b.

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f) \tag{3.4}$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i) \tag{3.5}$$

$$\tilde{\mathbf{C}}_t = \tanh(\mathbf{W}_C \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_C) \tag{3.6}$$

$$\mathbf{C}_t = \mathbf{f}_t * \mathbf{C}_{t-1} + \mathbf{i}_t * \tilde{\mathbf{C}}_t \tag{3.7}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o) \tag{3.8}$$

$$\mathbf{h}_t = \mathbf{o}_t * \tanh(\mathbf{C}_t) \tag{3.9}$$

In essence LSTMs maintain a flow of information through time in the form of the cell state ($\mathbf{C}_t$) and proposed cell state ($\tilde{\mathbf{C}}_t$), to which information is added, removed or outputted through the input ($\mathbf{i}_t$), forget ($\mathbf{f}_t$) and output ($\mathbf{o}_t$) gates. Using the sigmoid function (σ), each gate outputs a value between 0 and 1 which describes how much of this component should be let through to the recurrent states.

The introduction of LSTM also greatly increases the amount of trainable parameters in the network, as each gate adds its own set of weights and bias parameters. Gated Recurrent Units (GRU), introduced by Cho et al. [12], function similarly to LSTM cells but contain less parameters. This is done primarily by merging the cell state ($\mathbf{C}_t$) and hidden state ($\mathbf{h}_t$) and omission of the output gate ($\mathbf{o}_t$). Their function can be described by Equations 3.10 through 3.13:

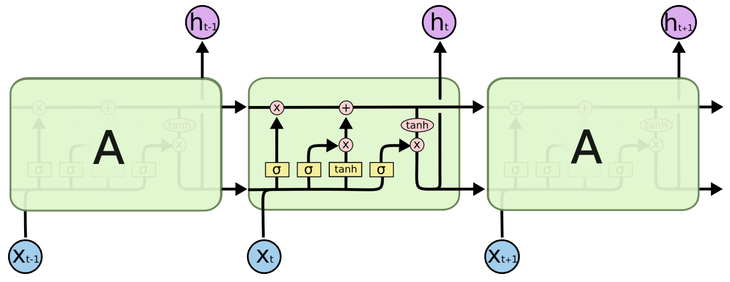
$$\mathbf{z}_t = \sigma(\mathbf{W}_z \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_z) \tag{3.10}$$

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_r) \tag{3.11}$$

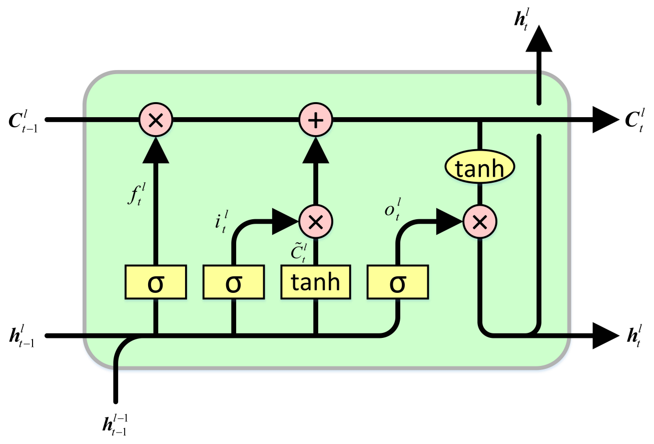
$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_h \cdot [\mathbf{r}_t * \mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_h) \tag{3.12}$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) * \mathbf{h}_{t-1} + \mathbf{z}_t * \tilde{\mathbf{h}}_t \tag{3.13}$$

where $\mathbf{z}_t$ denotes the update gate, $\mathbf{r}_t$ the reset gate, $\tilde{\mathbf{h}}_t$ and $\mathbf{h}_t$ the (proposed) hidden states and **W** and **b** the respective weights and biases.



(a)



(b)

Figure 3.7: (a): Information flow through an LSTM neuron over time (b): Detailed flow through a single LSTM neuron. From Olah [52], who provides a detailed explanation of the inner workings of LSTM.

Beside the addition of LSTM or GRU neurons, another option to aid RNNs in learning temporal relations are bidirectional layers. Bidirectional recurrent layers, introduced by Schuster et al. [56], aim to use both the past and future information at every timestep by presenting the sequence both forwards and backwards to two separate hidden layers, which are then merged. Bidirectional layers contain double the amount of neurons and thus parameters. The layout of such a bidirectional layer is illustrated in Figure 3.8. Use of bidirectional layers has been shown to help in classifying temporal patterns across many domains, where Du et al. [16] also illustrated its value for action recognition.

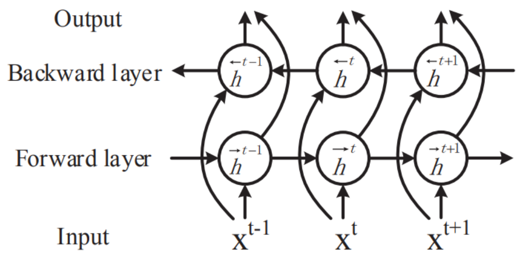


Figure 3.8: Illustration of a bidirectional hidden layer, from Graves [19]

3.4. Network architectures

The design of RNN architectures gives the user a lot of freedom in the amount of layers and neurons per layer used. Compared to large image recognition networks such as VGG-16 [60] or ResNet [27], action recognition networks are typically much smaller in size. This is because action recognition typically occurs in the stage after object recognition and in the case of skeleton based action recognition already works on higher level features (extracted using the larger OpenPose network).

There are limited clear guidelines on what kind of design works best per application. Due to the large computation cost of training networks, a grid search or alternative optimization search over the network parameters is computationally very expensive. Interesting work has been done by Zoph and Le [74] and Zoph et al. [75] regarding the efficient search for designing neural networks. In the scope of this work, we stick to the direct comparison of three different architectures, each following different principles.

Firstly, the DBRNN-L network from the work of Du et al. [16] is used as baseline network. Secondly, a simple and large LSTM network is used. Finally, a network inspired by the work of Maghoumi and LaVialo [46] is used, which combines a more complicated structure with use of GRU layers. A visual representation of the networks can be found in Figure 3.9. Table 3.1 provides an overview of the parameter size.

Evaluating the performance of these three networks will be done in Chapter 5. The goal of the comparison of these networks is to gain insight regarding which design methods work best for traffic gesture recognition.

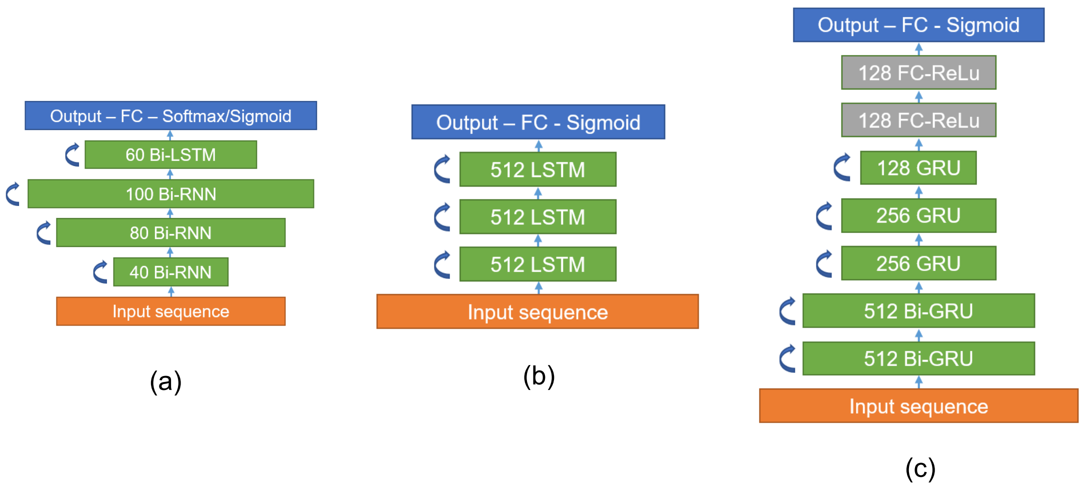


Figure 3.9: Overview of the RNN architectures, respectively DBRNN-L (a), LSTM based design (b), DeepGRU based design (c). Input/output layer size is data and feature dependent. The recurrent dimension is hidden for simplicity, but instead recurrent layers are indicated by the arrow to the left of the layer. Bidirectional layers have the Bi-prefix. Fully connected layers are depicted as FC with the corresponding activation function.

Network	Skeletal representation	Nr. of parameters
DBRNN-L	Coordinates/Joint-Line Distance	288.811
JL_D-LSTM	Joint-Line Distance	7.328.301
JL_D-GRU	Joint-Line Distance	10.967.341

Table 3.1: Overview of the architectures evaluated and their parameter size.

3.4.1. DBRNN-L architecture

The DBRNN-L network design of Du et al. [16] consists of 5 learnable layers. The first three layers are bidirectional conventional RNN neurons with a tanh activation function, the fourth layer is a bidirectional LSTM-RNN layer, followed by a fully connected softmax layer. Only one LSTM layer is used to reduce the number of parameters while maintaining a large and deep network structure, according to the authors. Figure 3.9a shows a visual representation of the neuron structure of the network.

3.4.2. LSTM architecture design

Similar to the work of Zhang et al. [72], which introduced the Joint-Line distance skeleton representation, a large LSTM network is implemented. This network consists of 3 LSTM layers, consisting of 512 neurons each. Batch normalization is applied between the last LSTM layer and the fully connected layer. Figure 3.9b gives a schematic view of the network.

The design philosophy behind this network is to create a large LSTM network, in which the network is relatively free to learn the higher-level features and gestures. As discussed in Section 3.3, this does significantly increase the parameter size of the network, illustrated by Table 3.1.

3.4.3. GRU architecture design

Inspired by the DeepGRU network of Maghoumi and LaVialo [46], a similar network was designed for evaluation on the Greenscreen Police Gesture Dataset. Main differences compared to previous networks are the use of GRU neurons instead of LSTM neurons and the addition of two fully connected layers after the recurrent part of the network. Additionally, this network is structured like a pyramid, increasing the depth of the network and allowing for more higher-level features, while keeping the network size reasonable. Table 3.1 shows that despite the larger network size and use of bidirectional layers, JL_D-GRU has roughly the same order of parameters as JL_D-LSTM, due to the use of GRU neurons over LSTM neurons.

Another technique used in the work of Maghoumi and LaVialo is the attention model, where certain parts of a sequence are deemed more relevant than others. Implementation of their attention module was omitted for now, due to implementation issues encountered in Keras [13]. Instead, the first two layers are used bidirectionally, similarly to the DBRNN-L network, which is another approach to aid temporal modeling. The inclusion of bidirectional layers does increase the network size more compared to attention modules, which has the resulting downsides of being more prone to overfitting and higher computational cost. However, additional methods for handling temporal information was deemed important for classification of dynamic gestures to offset these downsides.

3.5. Classification

Two separate classification methods are used in this work. Firstly, a multi-class classification set-up is used for the Berkeley MHAD dataset of the experiments in Section 5.3.1. For every input x_i, there is a single correct class y_i that is linked to that sample. This can be encoded as $y_i = \begin{bmatrix} 0, & 1, & 0 \end{bmatrix}$ for a three class problem, where sample x_i belongs to class 2 in this example. Multi-class classification is done by using a softmax activation function in the final layer in the network. The softmax function maps the input **z** of the function to the interval $(0, 1)$ using Equation 3.14, where the sum of the components of output vector $\sigma(\mathbf{z}_i)$ sum to 1. This essentially assigns a likelihood probability from the network to each class. The loss function used for optimizing the softmax function is the categorical cross-entropy (also known as log-loss) presented in equation 3.15, where N is the number of observations, M number of classes, y_{ij} a binary indicator whether the class label is correct and p_{ij} the model's predicted probability that a given observation belongs to class j.

$$\sigma(\mathbf{z}_i) = \frac{e^{z_j}}{\sum_{j=1}^{K} e^{z_i}} \text{ for } i = 1, \dots, K \tag{3.14}$$

$$\text{Categorical cross-entropy} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} y_{i,j} \log(p_{i,j}) \tag{3.15}$$

The second classification approach uses a multi-label approach. In a multi-label setting an sample can belong to multiple classes. This is applicable when classes are not mutually exclusive. For example, a sample can have the labels $\mathbf{y}_i = \begin{bmatrix} 0, & 1, & 1 \end{bmatrix}$, where this sample belongs to classes 2 and 3. The softmax activation function can not be used here, as the sum of the output probabilities can be greater than 1 for multi-label cases. Instead, a sigmoid activation function is used, given by Equation 3.16. This function maps the output of each class neuron in the final layer to the interval $(0, 1)$, essentially creating a probability per class.

Use of the sigmoid activation function as final layer has the implication that class output probabilities are independent from each other from the perspective of the final layer. This means that each class can be seen as a individual binary classification problem. This is also reflected in the loss function used for sigmoid layers, namely the binary cross-entropy seen in Equation 3.17.

$$\sigma(\mathbf{z}_i) = \frac{1}{1 + e^{-\mathbf{z}_i}} \text{ for } i = 1, \dots, K \tag{3.16}$$

$$\text{Binary cross-entropy} = \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M} -(y_{i,j}\log(p_{i,j}) + (1 - y_{i,j})\log(1 - p_{i,j})) \tag{3.17}$$

3.6. Summary

This chapter discussed the entire classification pipeline. Starting with feature extraction using OpenPose in Section 3.1, followed by how to input those features to the classifiers in Section 3.2. The inner workings of RNNs and the used architectures were outlined in Sections 3.3 and 3.4, after which the exact classification methodology was discussed in Section 3.5.

Now that the methods to be evaluated are established, Chapter 5 will delve further into the exact implementation of these methods and comparisons using experiments to answer the research questions posed in Chapter 1.

4 Greenscreen Police Gesture Dataset

The foundations of almost all gesture recognition methods are based on a dataset containing the right gestures for proper functioning and evaluation. For the exact purpose of traffic gesture recognition, gesture recordings were made in a studio greenscreen setting. This data was used to construct a novel dataset, called Greenscreen Police Gesture Dataset (GPGD) [3], and is a good starting point for evaluation existing action/gesture recognition methods, without introducing environmental challenges.

This Chapter will introduce this dataset and describe its specifications and content. The label design of this dataset and the custom labeling tools created are also outlined. GPGD consists of 5 actors, continuously executing traffic gestures, spread over 93 high-resolution recordings, for a total recording time of 70 minutes. The recordings contain variations in outfit and equipment, such as high-visibility jackets, police helmets and the use of batons. The gestures are executed in different directions and execution orientations (i.e. orientation of the agent and the gestures he is executing w.r.t. camera), as if directing traffic at a four way intersection with the camera as observer. The gestures are used to indicate the traffic from a certain direction to stop or start moving. Table 4.1 gives an overview of all data present in GPGD.

	Resolution	Recordings labeled	Total video length (hr:min:sec)	Framerate	Actors	Gesture types	Gesture directions
Primary Camera	2048x1080	93	1:10:37	21 fps	5	2 (Start/Stop)	4 (N/E/S/W)
Front Camera	3840x2160	26	0:19:21	23,98 fps	5	2 (Start/Stop)	4 (N/E/S/W)

Table 4.1: Greenscreen Police Gesture Dataset specifications

The traffic gestures were filmed from two viewpoints, one slightly offset front facing (as seen in Figure 4.1) and one directly front facing[1] and somewhat lower (as seen in Figure 4.2). The former is used as primary viewpoint in GPGD, the latter will be used in the cross-viewpoint evaluation experiments and explained in more detail in Section 4.4. In this work, viewpoint will be used to refer to the camera perspective used, whereas orientation is used to refer to the actors orientation with respect to that camera.

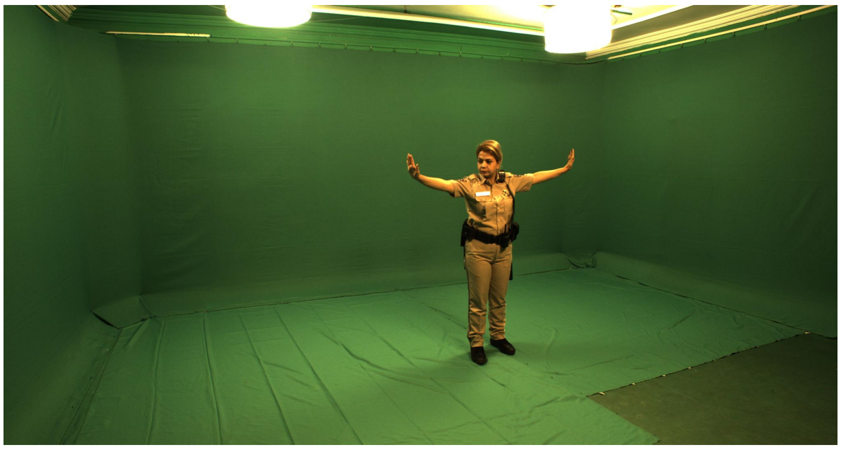


Figure 4.1: Sample frame from the primary camera viewpoint

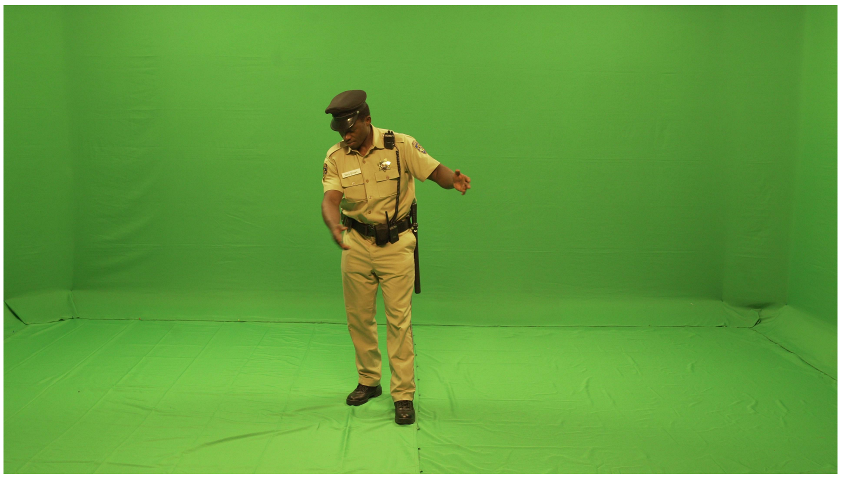


Figure 4.2: Sample frame from front facing camera viewpoint

4.1. Label design

There are two types of instructions present in GPGD: instruct traffic from a certain direction to stop and instruct traffic from a certain direction to start moving. Move instructions do not include a specific direction to move in. These instructions are not given one at a time, but most typically combined with another gesture (i.e. traffic from the south has to stop, while traffic from the west can go). The changes between instructions are independent of each other, meaning

[1]The recordings of the new viewpoint were unfortunately only available at a later time in the project and due to time-restraints only a subset of this viewpoint is labeled.

one instruction can stop being given, while the other is still given by the agent. As mentioned, there are four different agent orientations present in the dataset, creating another variable during the execution of gestures. In this work, the move instruction is referred to as Wave, whereas the stop instruction is referred to as Block.

Each actor in GPGD has their own gesture style. This means that the way instructions are combined and the speed at which they are changed varies from actor to actor. Some have a very calm, patient and clear style, where others have a quicker, more direct or sometimes casual style. The wide range in variation in the dataset is an accurate representation of the real-world problem.

The decision was made to label the high-level instructions themselves and their intended direction, independent of actor orientation or execution, and let the network learn the complete gesture implications. This ensures that the traffic implications for the observer are directly classified, compared to a set-up where the action and orientation would be labeled (e.g. extend arm while facing east), which needs additional processing/classification before traffic implications are reached. By considering the gestures not mutually exclusive, the problem moves from a multi-class classification problem to a multi-label classification problem, as described in Section 3.5.

The goal of labeling is to preserve as much of the gesture information per frame, allowing for more flexibility down the line. This is done by introducing the binary gesture vector, which consists of 8 binary values depicting the following gestures:

- Wave North
- Wave East
- Wave South
- Wave West
- Block North
- Block East
- Block South
- Block West

A ninth label called 'Other' is added during label parsing, which contains all sequences without any traffic gesture. While in a multi-label set-up the inclusion of a rest class can also be achieved by imposing a threshold on the class probabilities, it is useful to evaluate the ability of the classifier to actively classify when an agent is doing nothing, for example to mark transition points when an agent stops a gesture and does nothing, with the corresponding traffic implications. As the label probabilities are independently computed, this has no effect on the training of traffic gestures.

This label design has several advantages. Firstly, it allows for quick and intuitive labeling, where the user can quickly label the gesture using keyboard shortcuts. Secondly, it aims to preserve as much lower level gesture information as possible. To complete the label, the actor number is added for actor based train/test splitting.

Using the unique combinations of gestures (i.e. 'Wave South and Block East') as class labels was tried but encountered practical problems. The vast differences in gesture distribution resulted in a very imbalanced and spread out dataset with 27 classes. This made any form of cross-actor evaluation in this set-up impossible and some classes arguably not trainable, given they had less than 10 samples in the entire dataset. The multi-label setup proved to be much more useful and practical, both in training and network output.

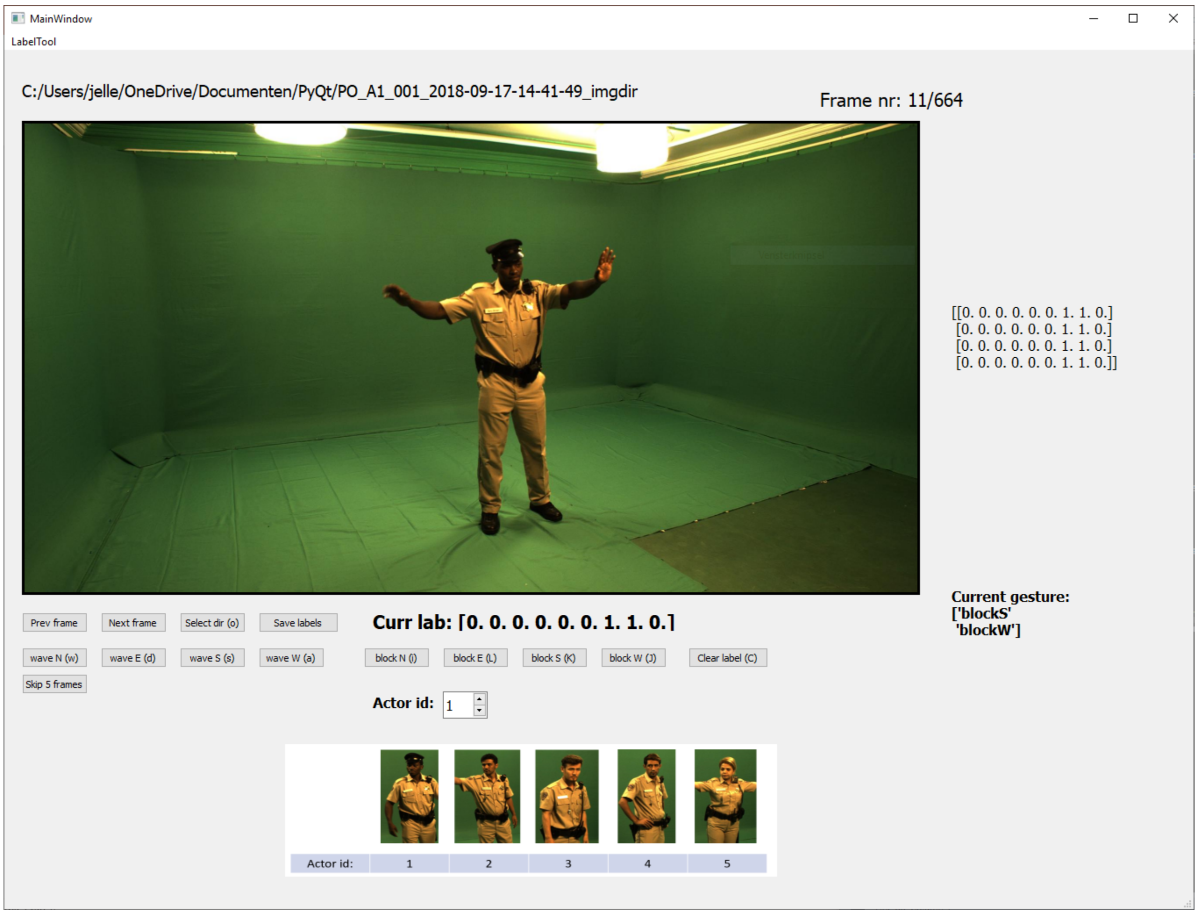
4.2. Label tool

All GPGD recordings are presented unlabeled, therefore manual labelling has to be done. As each recording contains a multitude of gestures in varying order, frame-by-frame labeling of the gestures is necessary. Research yielded no adequate existing labeling tools for this purpose. Most image/video based tools are focused on object labeling and lack time-series labeling capabilities, whereas time-series based labeling tools are mainly focused on sound and text labeling. Therefore, a custom label tool and label design was implemented specifically designed for this dataset.

The coding of the custom label application is done in Python, using PyQt5. Qt5 is a C++ based framework for the development of graphical user interfaces based applications. PyQt5 bridges the gap between this framework and python, allowing seamless coding transition between the label tool and further analysis and modelling code, also done in Python. The GUI is designed in Qt Creator 4.10.2, which allows for quick prototyping of the interface and a graphical design methodology, rather than having to hard-code every widget. The final GUI used can be found in Figure 4.3.

The workflow is designed such that the user selects the current gesture and direction using intuitive keyboard shortcuts (w,a,s,d and i,j,k,l) for the 2 gesture types, both in 4 directions, and can quickly scrub through the recording. With this workflow, the user effectively only has to label the transition point.

One issue encountered while processing the image sequences for GPGD was that the recording frequency of the images was not always constant, but decreased at some moments for some recordings due to ringbuffer overflow. Generating videos using the constant 21 fps of the non-affected image sequences results in videos that seem sped up slightly at some points. Roughly 30% of recordings are affected to some extent. Fortunately there seems to be no reason to assume that these frames are unusable, but is something to keep in mind during evaluation.



C:/Users/jelle/OneDrive/Documenten/PyQt/PO_A1_001_2018-09-17-14-41-49_imgdir

Frame nr: 11/664

```
[[0. 0. 0. 0. 0. 0. 1. 1. 0.]
 [0. 0. 0. 0. 0. 0. 1. 1. 0.]
 [0. 0. 0. 0. 0. 0. 1. 1. 0.]
 [0. 0. 0. 0. 0. 0. 1. 1. 0.]]
```

Curr lab: [0. 0. 0. 0. 0. 0. 1. 1. 0.]

Current gesture:
['blockS'
'blockW']

Actor id: 1

Figure 4.3: Screenshot of the label tool in action. The Image in the center shows the current frame, which number is also depicted in the top right. Current working image directory is found above the image frame. All necessary buttons and selectors are found beneath the image frame and can also be operated using keyboard shortcuts. Center bottom is a reference image for the actor ID's. The current label vector and an overview of the adjacent frames can be found respectively beneath and to the right of the frame. For ease of use, there is also an textual representation of the current gesture displayed.

4.3. Dataset analysis

The labeled recordings are split into individual sequences of uniform labels to create a segmented dataset. This is done by splitting the recording at all points where any label changes. In total 1288 sequences are extracted, which are used for training and testing. Train/test splitting is done using leave-one-out cross validation, meaning that every actor is evaluated by a classifier trained on the sequences from all other actors. All actors are evaluated using this scheme. Due to the small size of the dataset and limited amount of actors, no validation set is created, as the remaining training data is expected to be too small for all methods used.

Individual label occurrence in the extracted sequences can be seen in Figure 4.4. One can observe an imbalance between wave, other and block gestures. This can be explained by the fact that wave gestures tend to be longer gestures, whereas block gestures are typically shorter instructions. Due to the way the sequences are split, this introduces an imbalance in the amount of samples per label. The total amount of samples per actor is also mentioned in

Figure 4.4. Variation in sequence count between the actors can be explained by variation in gesture length and the simple fact that some actors occurred in more recordings. Figure 4.5 shows the distribution of the frame length per sequence. The dataset consists predominantly of shorter gestures with framecounts under 100 (roughly 5 seconds), with the majority even shorter.

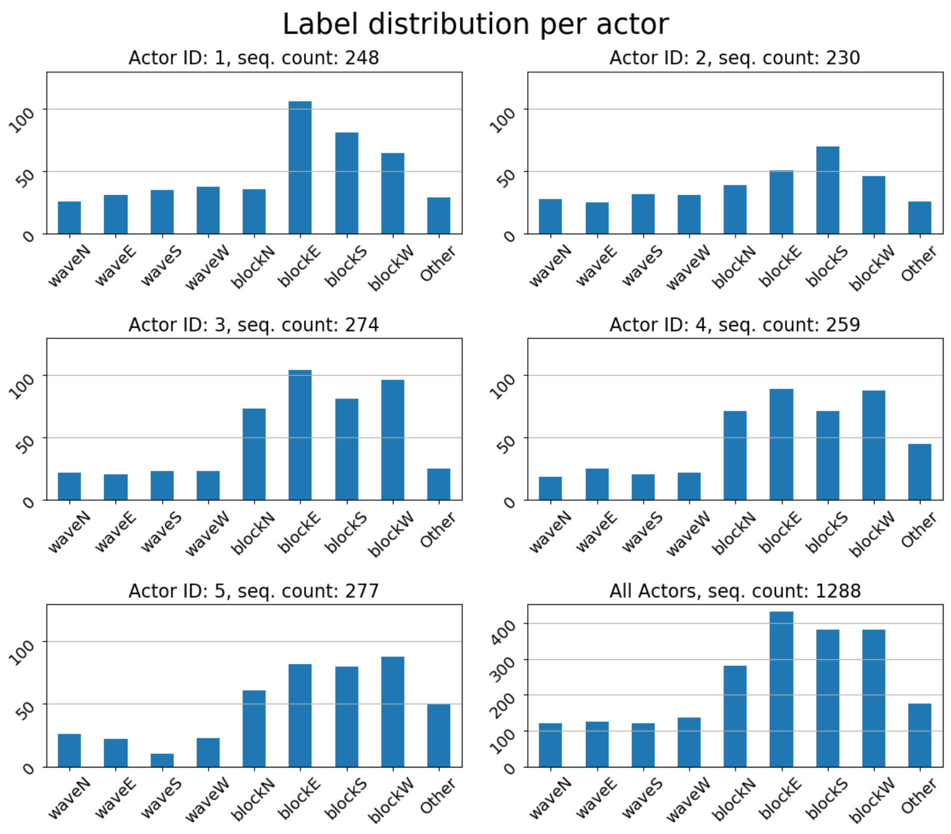


Figure 4.4: Gesture label distribution per actor and of all actors.

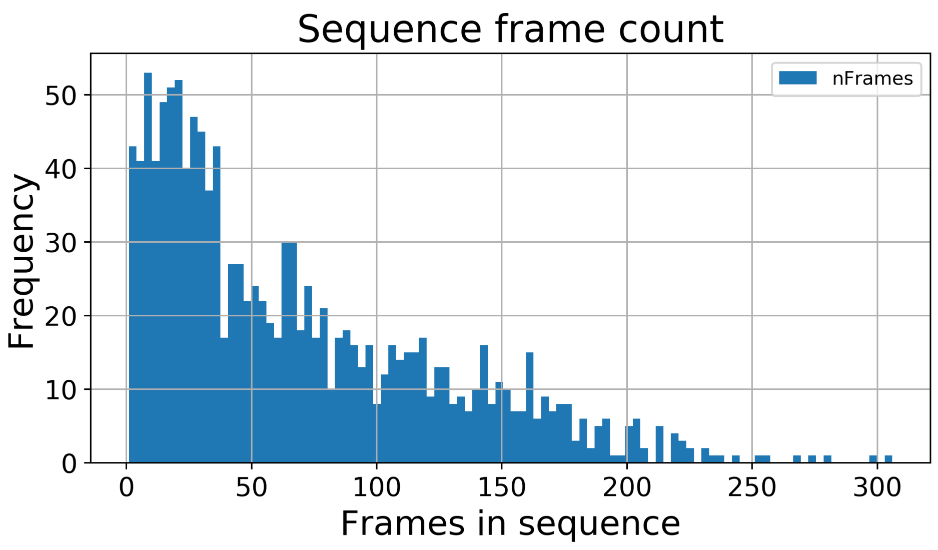


Figure 4.5: Histogram of the frame count per gesture sequence of the entire dataset

4.4. Viewpoint variation subset

The gesture sequences executed for GPGD were also recorded from a second camera viewpoint, directly facing the actor as can be seen in Figure 4.2, similar to the perspective of an approaching car. Gesture recordings from two viewpoints allows for a range of experiments, such as cross-viewpoint variation to evaluate classification and feature robustness and to evaluate transfer learning to new view points.

These recordings are not time synchronized with the primary viewpoint and recorded at a different framerate (23,98 fps vs 21). Also, the overlap between the different viewpoints is not exact, meaning some recordings have less or more frames captured compared to the primary viewpoint. This makes label transfer unfeasible and required separate labeling of this viewpoint. As the execution of the gestures was not changed, the same label and tools set-up are used.

When considering occlusions, this viewpoint is considerably harder than the original viewpoint. Gestures in the north direction are often blocked entirely by the body, especially block gestures. Accurate human labeling in slow-motion is already difficult in most cases and has to be inferred from the rest of the body. 405 new sequences are labeled in total. The distribution of the labels can be seen in Figure 4.6. Due to the lower sample count, imbalances are worse compared to the full original dataset.

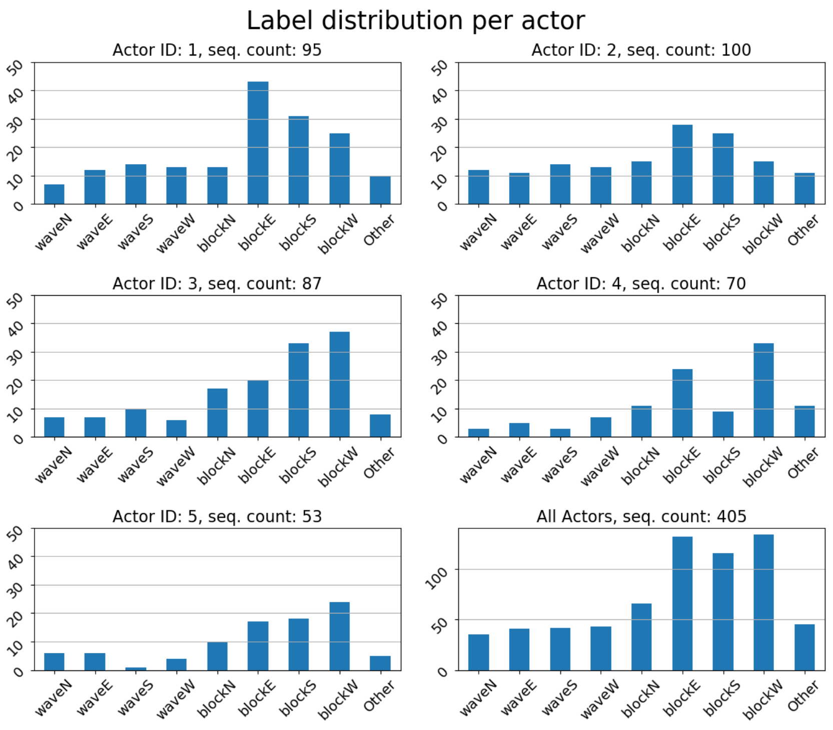


Figure 4.6: Gesture label distribution per actor and of all actors for the front viewpoint subset.

5 Experiments

This chapter will present the experiments executed using the methods outlined in the previous chapter. The experiments are designed in such a way to provide answers to the research questions posed in Chapter 1. The metrics used to quantify the results and answer the research questions are explained in Section 5.1. Section 5.2 will describe the datasets used in the experiments, namely Berkeley MHAD and the novel Greenscreen Police Gesture Dataset [3]. Dataset specific preprocessing is also outlined in these sections.

Section 5.3 will discuss the five experiments conducted. The first experiment conducted evaluates the performance of OpenPose based action recognition on Berkeley MHAD. Skeletal features generated using OpenPose are compared to motion capture data to determine the extent to which these features can describe gestures/actions.

The second experiment conducted in this work analyses the novel Greenscreen Police Gesture Dataset using a variety of state-of-the-art action recognition methods, as outlined in the previous chapter. The goals of this experiment are twofold; firstly, to gain insight on challenges within the dataset and provide a baseline on the novel dataset. Secondly, to analyse performance differences among several network architectures.

The third experiment conducted evaluates viewpoint robustness by evaluating the models generated in the previous experiment on a new viewpoint from the Greenscreen Police Gesture Dataset. Additionally, the value of transfer learning for gesture recognition is evaluated in this experiment using two different transfer learning schemes.

The ability of the gesture classification to generalize to a continuous gesture sequence will be evaluated in the fourth experiment. This will be done using the configuration that showed the best performance in the previous experiments. Additionally, qualitative analysis of the classification will be performed by comparing the continuous performance to the corresponding video frames.

Finally the computation time will be evaluated. Two configurations are compared, which each represent two ends of the spectrum regarding network size.

5.1. Metrics

Choosing the right evaluation metric is important for accurate evaluation of the experiments. The most obvious metric, and common in gesture datasets, is to use the accuracy of the classifier, defined as the fraction of samples in the dataset that is correctly labeled by the classifier. This can give a good overall indication of the performance of the classifier, but has its limitations. For example, in the case of very imbalanced datasets, accuracy gives skewed results. Fortunately, the Berkeley MHAD dataset used for evaluating OpenPose feature representation is balanced and can simply be evaluated using accuracy. Additionally, confusion matrices are suitable to provide a good in-depth view on classification performance and where classifiers struggle, by giving a clear view on what classes are difficult to distinguish.

Metrics for the multi-label Greenscreen Police Gesture Dataset [3] are less straightforward as accuracy gives poor detailed insight for multi-label problems. Where samples that belong to only one class can only be correct or wrong, samples that can belong to several classes at once can also be partially correct/wrong. Using only accuracy fails to capture the degree to which samples are mislabeled, as it only accounts for completely correct labels. Precision, Recall and F1-score (Equations 5.1 to 5.3) can provide more detailed insight in this case:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5.1}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5.2}$$

$$\text{F1-score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{5.3}$$

where TP, FP and FN are true positive, false positive and false negative respectively. These metrics are calculated per class. Overall performance is calculated by averaging the class scores (also called macro-average) for each actor. The resulting metrics are again averaged with equal weight. This assigns equal weight to each class in the dataset and prevents class imbalances from skewing the results. Use of these metrics is common across many fields such as information retrieval, medical science and machine learning.

Additionally to these numerical metrics, ROC curves are computed for each class, which provide a clear visual view of classification confidence and performance. This is done by plotting the true positive rate (TPR, shown in Equation 5.4) against the false positive rate (FPR, shown in Equation 5.5) at increasing decision thresholds. In a multi-label configuration, confusion matrices can only be calculated on a per class basis and thus don't capture class confusion. They are therefore less relevant and ommited for this configuration.

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5.4}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{5.5}$$

These metrics give a more accurate view of per class performance, but fail to capture the accuracy of the entire predicted label and the correct handling of negative samples. Therefore, further metrics used for evaluation of the multi-class case are the Hamming loss, average precision and label ranking average precision. The Hamming loss (also called Hamming distance, introduced by Richard Hamming [23]) between two sets of labels is given in Equation 5.6,

$$L_{\text{Hamming}}(y, \hat{y}) = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \oplus(\hat{y}_{i,j} \neq y_{i,j}) \tag{5.6}$$

where N is the number of samples, M the number of classes, $\oplus$ denotes the XOR function and y and $\hat{y}$ are respectively the true labels and predicted labels. This is a measure of the average distance from the correct label and absolute predictions. It can be used both on a frame-by-frame basis given a single sequence and on a set of final predictions. A lower Hamming loss is desirable.

The mean Average Precision (mAP) summarizes precision-recall performance by averaging the precision for each recall at increasing decision thresholds. Mean Average Precision is calculated per class using Equation 5.7, after which it is averaged on a macro level. In Equation 5.7 tr is the current decision threshold count, R_{tr} and R_{tr-1} recall at the current and previous threshold and P_{tr} precision at the current threshold.

$$\text{AP} = \sum_{\text{tr}} (R_{\text{tr}} - R_{\text{tr}-1}) P_{\text{tr}} \tag{5.7}$$

Label ranking average precision asks for each ground truth label in a sample what fraction of higher-ranked labels are correct. The final score domain is between 0 and 1, where 1 is the best value. This metric scores the ability of the classifiers to correctly rank true labels over false labels. Given the true labels $y \in \{0, 1\}^{N \times M}$ and predicted scores $\hat{f} \in \mathbb{R}^{N \times M}$, its computation is shown in Equation 5.8:

$$\text{LRAP}(y, \hat{f}) = \frac{1}{N} \sum_{i=0}^{N-1} \frac{1}{||y_i||_0} \sum_{j: y_{ij}=1} \frac{|\mathcal{L}_{ij}|}{\text{rank}_{ij}} \tag{5.8}$$

where $\mathcal{L}_{ij} = \left\{k : y_{ik} = 1, \hat{f}_{ik} \geq \hat{f}_{ij}\right\}$, $\text{rank}_{ij} = \left|\left\{k : \hat{f}_{ik} \geq \hat{f}_{ij}\right\}\right|$, $|\cdot|$ computes the number of element in the set and $||\cdot||_0$ is the ℓ_0 norm that computes the number of nonzero elements in the vector.

5.2. Datasets

As mentioned, the methods outlined in Chapter 3 are evaluated on two different datasets: Berkeley Multimodal Human Action Dataset captured by Ofli et al. [51] (Berkeley MHAD) and the novel Greenscreen Police Gesture Dataset (GPGD), which was introduced in Chapter 4. This section will, in the case of Berkeley MHAD describe the datasets and for both outline the data-specific preprocessing steps taken with each dataset.

5.2.1. Berkeley Multimodal Human Action Dataset

Berkeley MHAD consists of 11 actions, performed by 12 subjects (of which 7 males and 5 females of varying ages). An overview of the dataset, including the repetitions and recording duration is provided in Table 5.1. In total, the dataset contains 82 minutes of recordings, spread over 660 action sequences. A wide range of modalities was recorded:

- Optical Motion Capture
- Multi-view video
- Kinect depth imaging
- Accelerometer data
- Audio

For our research, only the motion capture data and video images are used and discussed, as our interests lies in skeletal features and the visual estimation of those features.

For the motion capture ground truth, Ofli et al. [51] used the Impulse motion capture system of PhaseSpace [31], which captures the 3D position of 43 active LED markers on the capture suit at a frequency of 480Hz. The marker trajectories were post-processed by Ofli et al. [51] to extract the skeletons. The skeleton data was provided in the BVH file format and is visualized in Figure 5.2b. Compared to the BODY_25 format used by OpenPose, this skeleton format contains more data points, especially more between joint locations, and less facial and foot keypoints.

A total of 12 camera's, spread over 4 clusters, were used to capture the actions from different viewpoints. Two of the clusters had two camera's for stereo vision, whereas the other two had four camera's for multi-view capture. The cameras used (Dragonfly2 cameras of Point Grey Research [32]) recorded images at a resolution of 640 by 480 pixels at a frequency of 22Hz. In this work, only data from camera 1 from cluster 1 was used, to ensure that the training/test set of the OpenPose and motion capture experiment were of the same size. Some sample frames of all actions can be seen in Figure 5.1. All sensor data was temporally synchronized. Figure 5.2a provides an overview of the total capture setup used.

	Action	Repetitions per Recording	Number of Recordings	Approximate Length per Recording
1	Jumping in place	5	5	5 sec
2	Jumping jacks	5	5	7 sec
3	Bending - hands up all the way down	5	5	12 sec
4	Punching (boxing)	5	5	10 sec
5	Waving - two hands	5	5	7 sec
6	Waving - one hand (right)	5	5	7 sec
7	Clapping hands	5	5	5 sec
8	Throwing a ball	1	5	3 sec
9	Sit down then stand up	5	5	15 sec
10	Sit down	1	5	2 sec
11	Stand up	1	5	2 sec

Table 5.1: Overview of action classes and recordings of the Berkeley MHAD dataset.

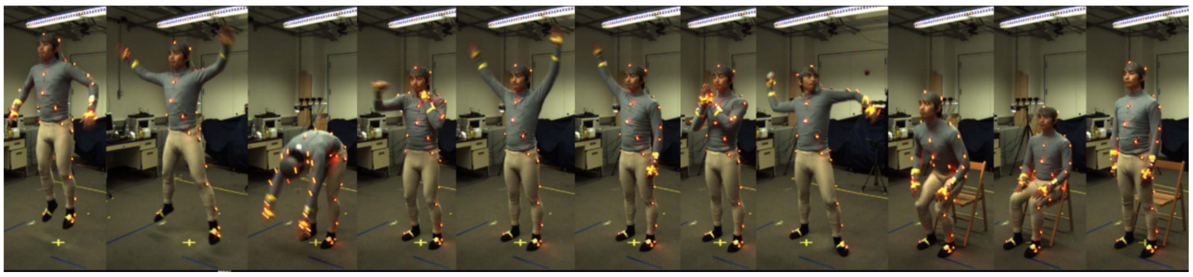
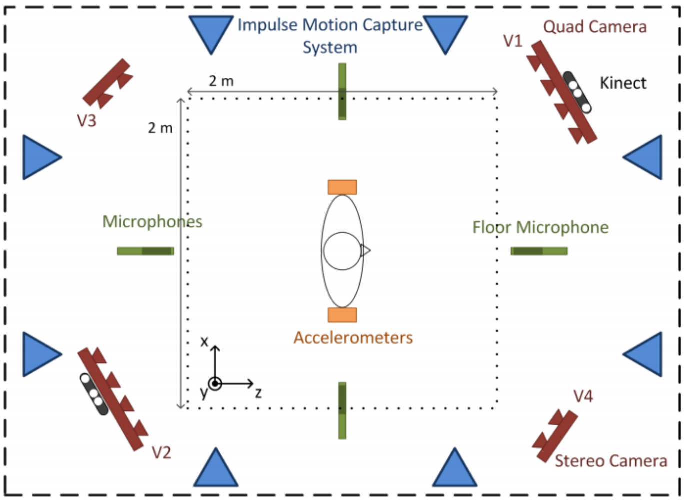
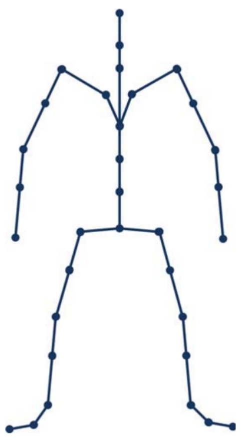


Figure 5.1: Examples from all the actions in the work of Ofli et al. [51] as seen from cluster 1



(a)



(b)

Figure 5.2: (a): Overview of the capture setup of Ofli et al. [51] (b): Skeleton construction of Berkeley MHAD, figure from Chaudry et al. [11]

5.2.1.1 Preprocessing

For validation of the DBRNN-L network against the results of Du et al. [16], the data preprocessing pipeline is designed similar to theirs. First, the original motion capture data, which was recorded at 480 Hz, is smoothed using the Savitzky-Golay smoothing filter [55], represented in Equation 5.9. Subsequently, the data was downsampled to 22 Hz to match the camera framerate and thus the OpenPose skeletons, by simply selecting 1 in every 22 frames. Finally, the coordinates were recentered around a new origin defined as the average of the three hip joints, as was seen in Equation 3.1. Similar to Du et al. [16], no scaling of the joint locations is applied, leaving the coordinates in centimeters w.r.t the skeleton center. Joint information is fed to the network as concatenated input vector $\mathbf{x} \in \mathbb{R}^{N \times L}$, where $N = 35 \times 3 = 105$.

$$f_i = \frac{-3x_{i-2} + 12x_{i-1} + 17x_i + 12x_{i+1} - 3x_{i+2}}{35} \tag{5.9}$$

To evaluate the amount of motion information that is preserved in 2D, classification based on purely 2D motion capture data is also tested. This is done by simply dropping the depth information from the joint coordinates. The result is a 2D view of the skeleton from the front. This leaves us with the input vector $\mathbf{x} \in \mathbb{R}^{N \times L}$, where $N = 35 \times 2 = 70$. It should be noted that this is not the same viewpoint as the cameras used, as illustrated in Figure 5.3. This is both a deliberate choice and practical limitation, as for the use-case of intelligent vehicles the main interest lies in classification from straight-on. Also, exact camera positions are unknown so exact mapping to camera perspective would have been impossible. This difference is not a hindrance, but still something to take in account when making direct comparisons to the OpenPose 2D data.

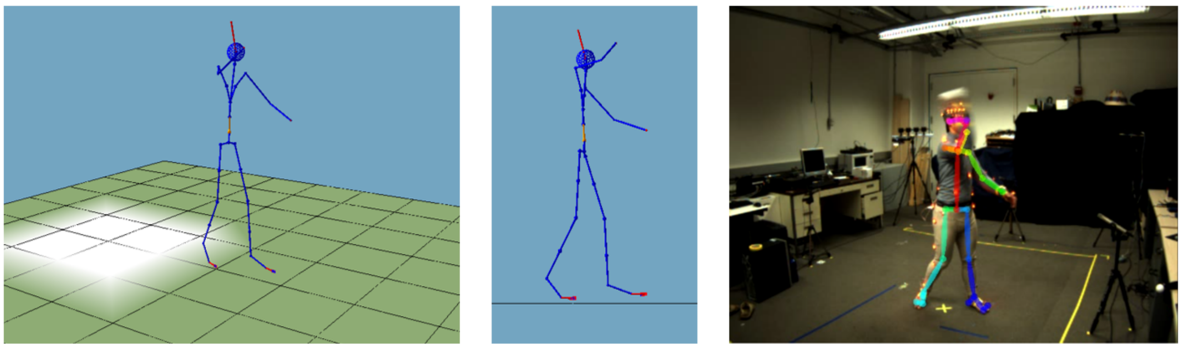


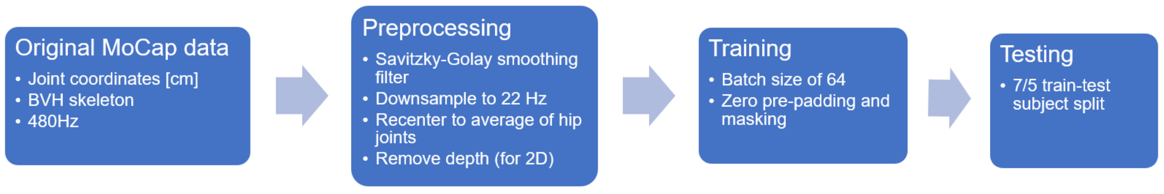
Figure 5.3: From left to right: motion capture datapoints in 3D space, seen from approximately the camera perspective, flattened 2D from the front and the OpenPose detection. All for the same frame of the action ”throwing ball”.

For the OpenPose based classification, image sequences from only 1 camera viewpoint are used (camera 1, viewpoint 1 in Figure 5.2a). The default OpenPose network resolution aspect ratio and scale is used, resulting in a network resolution of 492 by 368 given the 4 by 3 ratio of input images. OpenPose coordinates are preprocessed with the same hip centering of Equation 3.1. No downsampling or temporal smoothing is applied, due to the already low temporal resolution of 22Hz.

OpenPose based classification is tested with two skeletal representations. Firstly, simply feeding the coordinates as concatenated vector $\mathbf{x} \in \mathbb{R}^{N \times L}$, where $N = 25 \times 2 = 50$. Secondly, using the Joint-Line Distance skeletal representation outlined in Section 3.2 of Zhang et al. [72]. As shown, this results in the input vector $\mathbf{x} \in \mathbb{R}^{1012 \times L}$. Joint-Line Distance is not evaluated on the motion capture data in this work as this was already done in the work of Zhang et al. [72] and is not the focus of this experiment. No changes are made to the size of the network to accommodate the larger input vector, as this proved unnecessary during prototyping.

Following the conventional approach for this dataset, the first 7 subjects are used for training and the final 5 for testing. For the training sequences a slight change is made with respect to Du et Al.[16], namely recordings containing multiple repetitions are split into single repetitions. This allows for better parallelisation in the form of larger batch sizes and significantly faster training.

Training samples are pre-padded with zeros to ensure uniform sequence length across the batch. Zeros are filtered from the input by the network with a masking layer over the input. Test samples are not split and also used zero pre-padding. Evaluation of the test data is done based on the final class label assigned at the end of the recording. This means that some of the test recordings contain 5 repetitions. An overview of the entire evaluation pipeline is given in Figure 5.4.



(a)

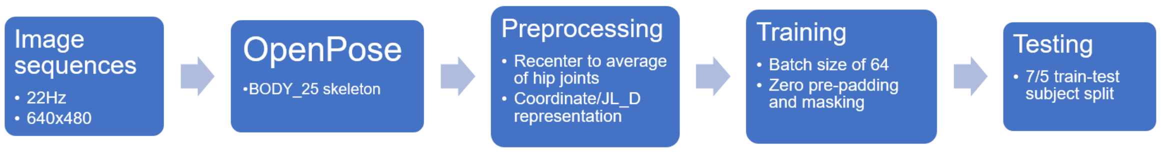


Image sequences
• 22Hz
• 640x480
OpenPose
•BODY_25 skeleton
Preprocessing
• Recenter to average of hip joints
• Coordinate/JL_D representation
Training
• Batch size of 64
• Zero pre-padding and masking
Testing
• 7/5 train-test subject split

(b)

Figure 5.4: Complete evaluation pipeline used for Berkeley MHAD for motion capture data (a) and OpenPose (b)

5.2.2. Greenscreen Police Gesture Dataset

The Greenscreen Police Gesture Dataset is used to determine how well the state-of-the-art action recognition methods outlined in Chapter 3 translate to the specific application of traffic gesture recognition. Additionally, this are the first experiments conducted on this dataset and will thus function as initial baseline and can help in identifying challenges presented within this dataset.

Visual output results of OpenPose on the Greenscreen Police Gesture Dataset show good and accurate detection of skeletons. Examples of detections can be seen in Figure 5.5. Due to the absence of ground-truth joint data, quantitative analysis of detection performance is not possible, nor is it the focus of this study. Limb occlusions can sometimes be inferred by OpenPose, but especially on frames without any hints of a limb OpenPose struggles. Detection also works well for dark and low contrast police outfits.

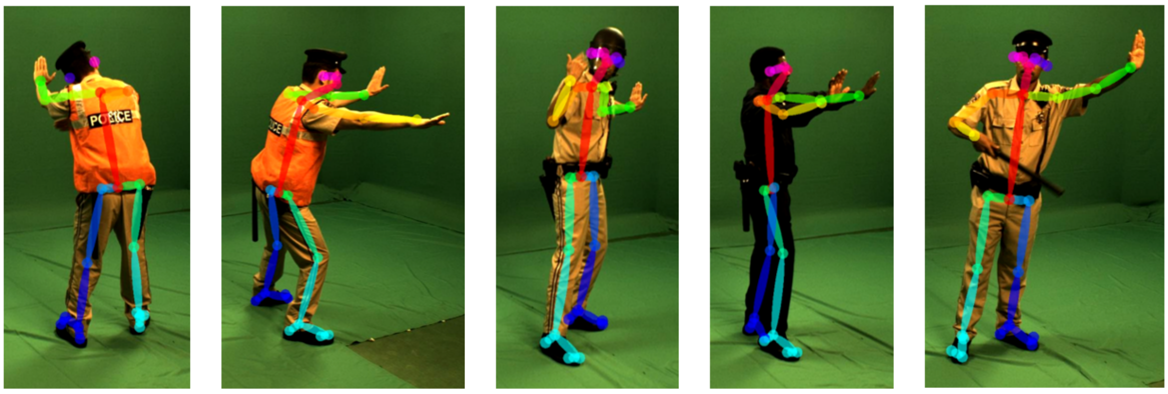


Figure 5.5: Qualitative analysis of OpenPose detection on GPGD. Examples of success and failure cases of OpenPose handling occlusions, successful face detection while wearing helmets, detection of dark outfits, lack of baton detection.

5.2.2.1 Preprocessing

Data preprocessing for the methods evaluated on GPGD is very similar to the Berkeley MHAD dataset preprocessing. OpenPose is run on the whole gesture sequence for 2D skeleton estimation, using the same settings as for Berkeley MHAD, resulting in a network resolution of 698 by 368. The resulting keypoints are centered around the hip joints, using Equation 3.1. An addition to the MHAD pipeline is scaling all skeleton coordinates by the spine length. This is done to make the features invariant to the distance from the camera and subject length, which is more inconsistent in this dataset compared to Berkeley MHAD and also a challenge for real-world applications. The spine was chosen as scaling factor, because due to its length it has the least relative fluctuation introduced by keypoint detection. Additionally, it is visible from all orientations and should suffer the least from viewpoint variation. As the joints were already centered to a skeleton center, normalization was done simply by:

$$\mathbf{x}_{\text{norm}} = \frac{\mathbf{x}}{L_{\text{spine}}} \tag{5.10}$$

where $\mathbf{x}$ and $\mathbf{x}_{\text{norm}}$ are the original and normalized joint coordinates and L_{spine} is the spine length scaling factor.

Outlier removal is done by imposing simple thresholds on the distance joints can be from the body center and replacing outliers with 0's. This proved effective enough to not warrant the use of more advanced statistical approaches. An overview of the complete data pipeline is given in Figure 5.6.

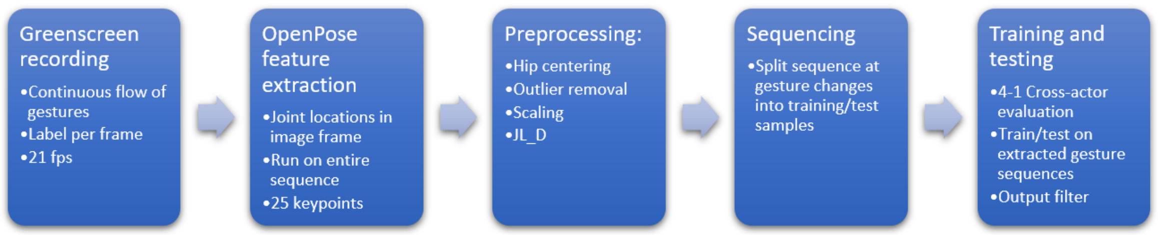

Greenscreen recording
•Continuous flow of gestures
•Label per frame
•21 fps
OpenPose feature extraction
•Joint locations in image frame
•Run on entire sequence
•25 keypoints
Preprocessing:
•Hip centering
•Outlier removal
•Scaling
•JL_D
Sequencing
•Split sequence at gesture changes into training/test samples
Training and testing
•4-1 Cross-actor evaluation
•Train/test on extracted gesture sequences
•Output filter

Figure 5.6: Data processing pipeline for GPGD

5.3. Experiments

In this section, the experiment are explained and their results evaluated in the order presented in the introduction of this chapter.

5.3.1. Evaluation of OpenPose as feature extraction

The goal of this experiment is to determine how well 2D pose estimation works for action and gesture recognition and what advantage feature representation can offer. The potential gap in performance to 3D skeletons is determined by feeding both accurate motion capture skeleton data and OpenPose features to the same baseline network and analysing the results. OpenPose based classification is evaluated using coordinate features and the joint-line distance feature representation. From these results, we can determine if and how much improvement hand-crafted skeletal features offer and if OpenPose based action recognition can get close to motion capture based action recognition.

5.3.1.1 Training

Our implementation is created using Keras [13] with a Tensorflow [1] back-end, running on a laptop with an i7-8750H 6 core CPU and a NVIDIA Quadro P1000 GPU. The model is trained using the Adam optimizer, of Kingma and Ba [37], which has been shown to be robust and fast. Learning rate was initialized at 1×10^{-3}. The default β parameters as presented in the original paper are used and proved to be effective enough to not warrant any further tuning. As discussed in Section 3.5, the categorical loss is used as loss function.

L2 regularization is applied across all layers with a regularization value of $1,5 \times 10^{-3}$. A batch size of 64 is used. Early stopping is applied on the test loss for lack of a validation set. This practice will be discussed more in 6.3.2.

5.3.1.2 Results

Table 5.2 shows classification accuracy results of all combinations of pose extraction and skeletal representation used. Figure 5.8 to 5.11 show the confusion matrices for these configurations. Similar to the work of Du et al. [16], the DBRNN-L baseline method based on motion capture data is able to achieve very high results on the Berkeley MHAD dataset. By achieving 98,55% accuracy, it nearly solves the test set. The only four missclassifications occur in the classes 'Waving - one hand', 'Clapping hands' and 'Sit down', as can be seen in the confusion matrix of Figure 5.8. Noteworthy is the fact that the 2D coordinate representation (seen in Figure 5.9) achieves identical accuracy as 3D coordinates, even the same number of errors. The only difference between 2D and 3D motion capture data are the type of errors, however no noteworthy trends can be seen in the error types. Depth information does not seem to be essential for accurate action recognition given accurate joint coordinates.

Network	**Pose extraction**	**Skeletal representation**	**Accuracy**
DBRNN-L	Motion capture	Coordinates - 3D	0,9855
DBRNN-L	Motion capture	Coordinates - 2D	0,9855
DBRNN-L	OpenPose	Coordinates	0,9345
DBRNN-L	OpenPose	Joint-Line Distance	0,9709

Table 5.2: Accuracy results of different skeleton estimation methods and skeleton representations on Berkeley MHAD. Identical accuracy values are caused by the same number of errors made on the test set.

Classification based on OpenPose coordinates achieves an accuracy of 93 %, which is quite a bit lower, especially for the relatively simple Berkeley MHAD dataset. Interesting is the confusion matrix of this configuration, seen in Figure 5.10. Classes with a lot of vertical or lateral movements, such as jumping, bending, waving and the sit/stand classes, perform nearly identical compared to the motion capture performance of Figure 5.8.

However, classes that show large amounts of joint overlap, such as punching, clapping hands or throwing a ball, show a clear decrease in performance compared to the motion capture results. Figure 5.7 shows some sample frames of the struggling classes. The common denominator in these detections and action types is the fact that many joints in the arms and shoulders are positioned roughly in the same place, close to the body. Occlusions in the frames seem to be handled well by OpenPose. Based on the confusions found, the network fails to distinguish the different classes with hands in similar places, where also the relation of joints to other joints is important.

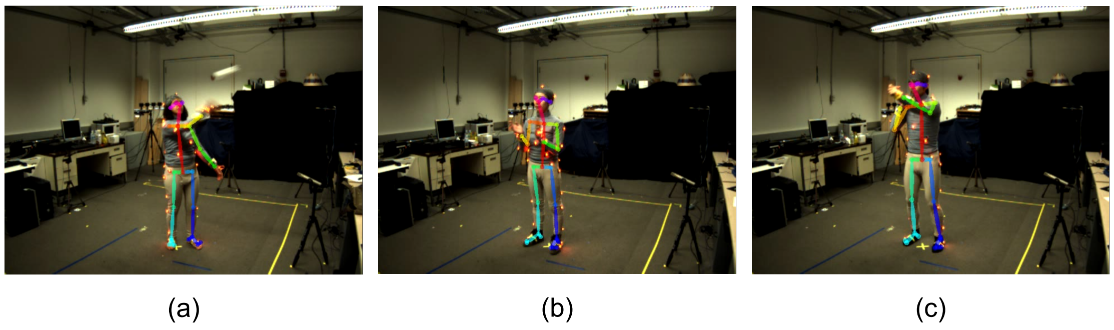


Figure 5.7: Sample frames of OpenPose keypoint estimation for poor performing classes. (a) subject 7, throwing ball (b) subject 9, clapping hands (c) subject 10, boxing

This makes the result of OpenPose based classification using JL_D representation all the more interesting. Use of the JL_D skeletal representation yielded a roughly 4% performance gain with respect to joint coordinates for OpenPose based classification. The accuracy achieved using more advanced skeletal representations is nearly identical to that of motion capture data. Clearly, better feature representation helps significantly in the case of 2D action recognition. Figure 5.11 shows that the classes with much joint overlap improved significantly.

Given these findings, it seems that the OpenPose BODY_25 skeleton, when fed as joint coordinates, proves to be not very distinguishable when joints are positioned closely together and fails to capture joint relations. Adding the JL_D representations appears to help substantially to capture the connections of the limbs, when joint coordinates are overlapping more, resulting in better classification. This is in line with previous research into skeleton representation, outlined in Section 2.3.

Noteworthy is the fact that the 2D motion capture coordinate configuration did not seem to struggle with the same problem. An explanation for this could lie in the different skeleton formats used. Compared to the BODY_25 format, the motion capture data contains much more descriptive limb information with multiple datapoints per limb. The presence of the extra intermediary datapoints better defines the exact limb orientation and might make it easier for the network to learn the joint relations. Additionally, the different viewpoint of the OpenPose detections compared to the 2D motion capture skeleton result in different viewpoints with regard to limb overlap.

Figures 5.12 and 5.13 visualize the output of the softmax-layer per timestep for two test samples, generated by the joint-only OpenPose configuration and 3D coordinate motion capture configuration respectively. The output at timestep t is based on all feature inputs from $x_{t=0}$ to x_t. The fraction of the total probability assigned to a class can be interpreted as the estimated probability for that class. Accuracy metrics of Table 5.2 are calculated using the final label $y_{t=N}$.

Both figures give a good indication of what the network 'thinks' while looking at the test sequence. Figure 5.12 shows that throwing the ball is first confused with clapping hands, which is in fact a logical link. In this case, the ball is thrown by first bringing the hands together, followed by throwing with one hand. In Figure 5.13, the network shows confusion between the different sit/stand classes, before finally converging on the right class. Looking at the sequence in question shows no clear 'logical' confusions in this case. Given the fact that the network does finally converge to the right class, one could conclude that in this case the network struggles to model the sequence with limited temporal information, but given enough temporal information does end at the right solution.

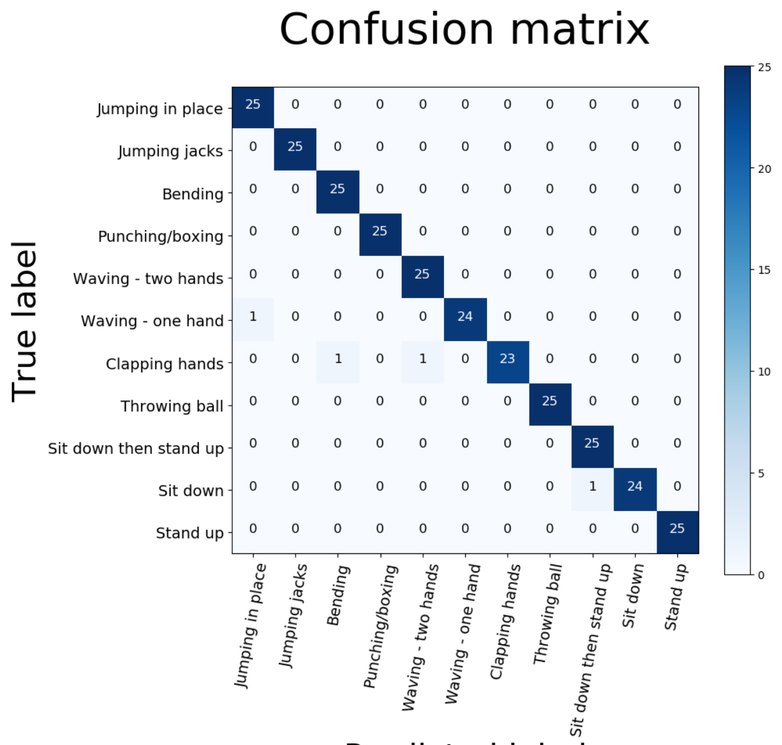


Figure 5.8: Confusion matrix of DBRNN-L with 3D motion capture coordinates

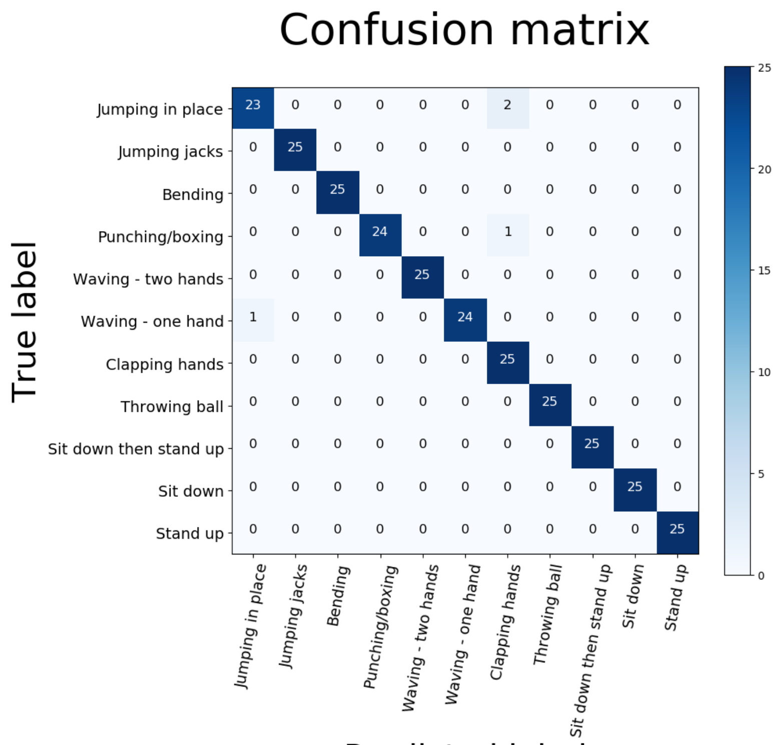


Figure 5.9: Confusion matrix of DBRNN-L with 2D motion capture coordinates

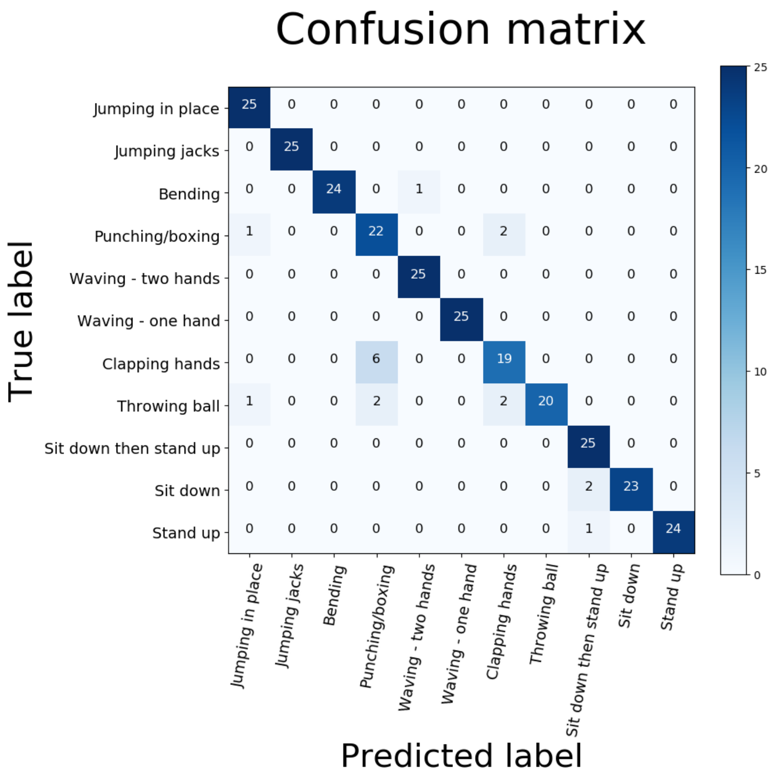


Figure 5.10: Confusion matrix of DBRNN-L-OpenPose with coordinates

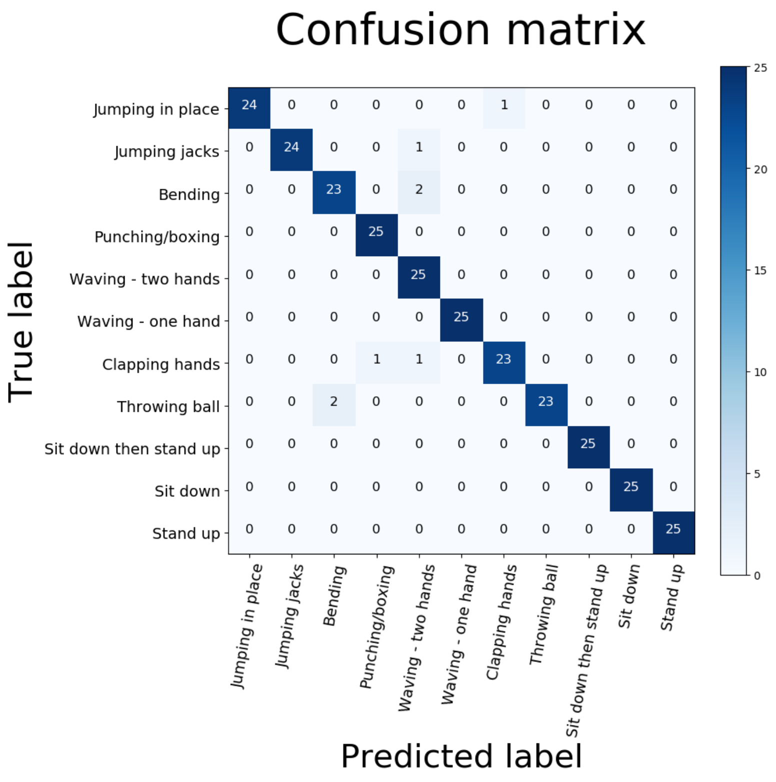


Figure 5.11: Confusion matrix of DBRNN-L-OpenPose with JL_D representation

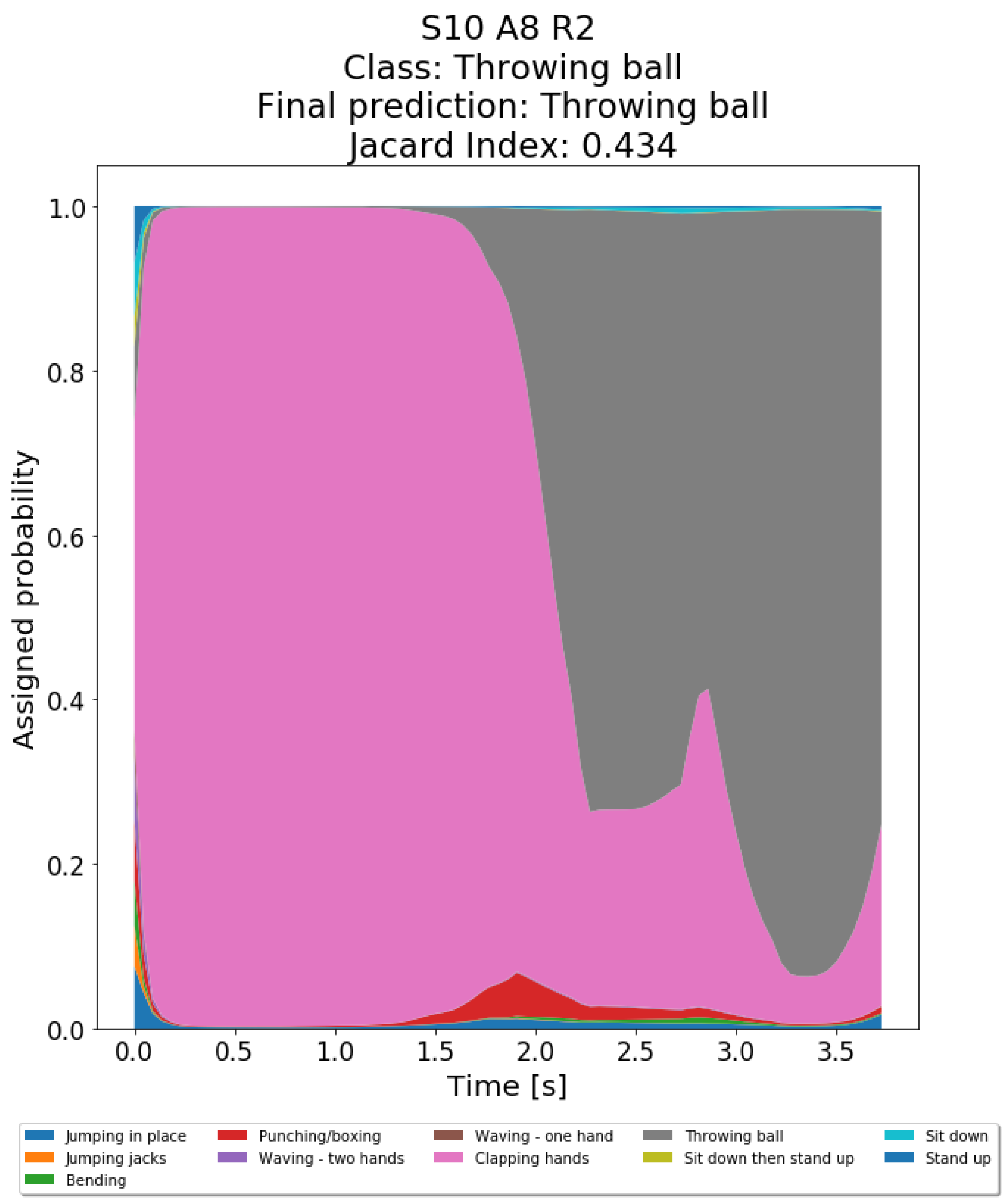


Figure 5.12: Timeline visualisation of the coordinate based OpenPose network classification output during a sequence, including the Jacard index of similarity for the specific sequence.

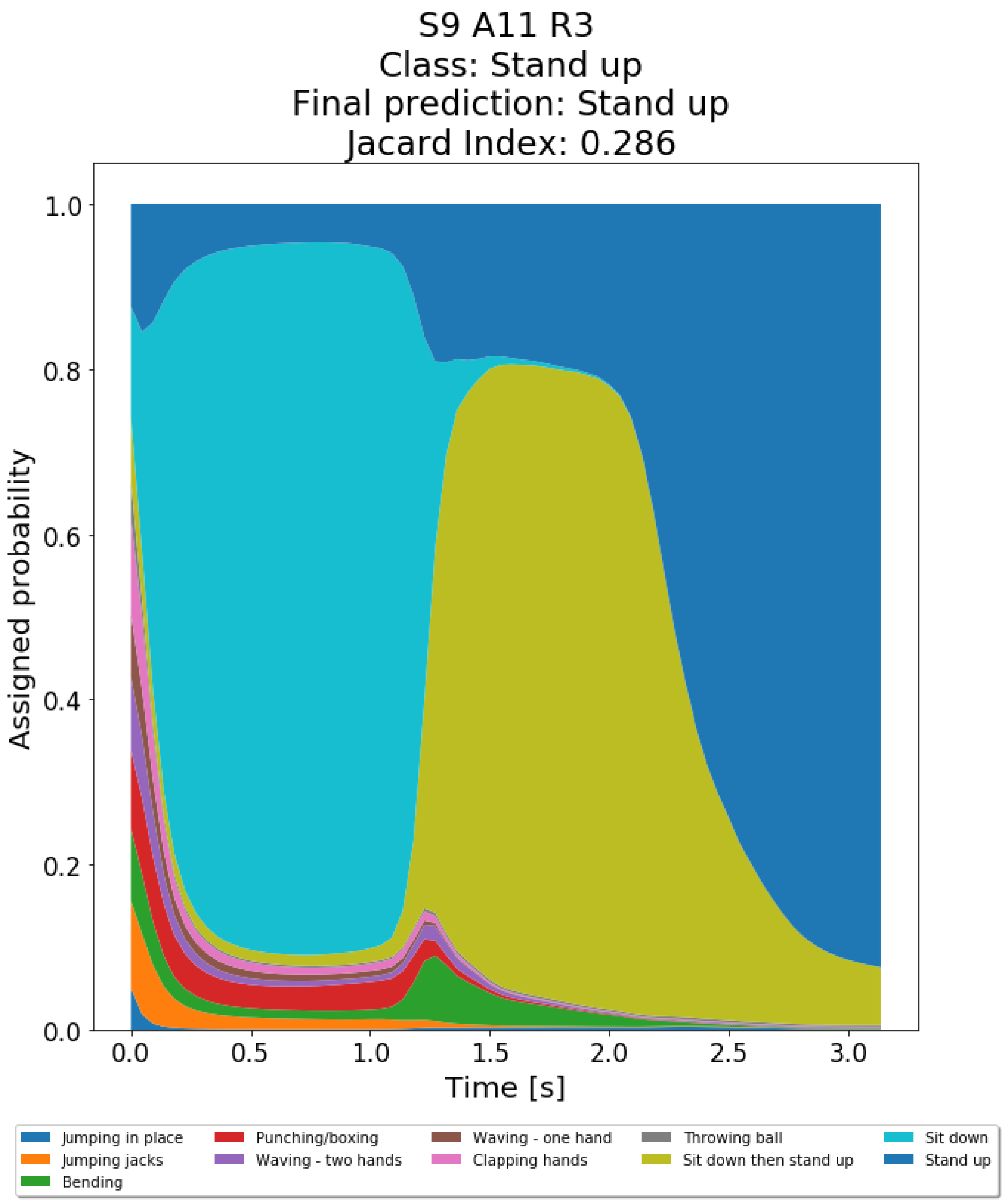


Figure 5.13: Timeline visualisation of the 3D motion capture network classification output during a sequence, including the Jacard index of similarity for the specific sequence.

5.3.2. Evaluation of traffic gesture recognition

The goal of this experiment is to determine how well the existing state-of-the-art action recognition methods outlined in Chapter 3 perform on the Greenscreen Police Gesture Dataset. Using the metrics outlined in Section 5.1, we gain the first insights on classification performance on this novel dataset.

5.3.2.1 Regularization

Good regularization proves to be challenging for GPGD. Therefore a wide range of regularization techniques are tested and applied during prototyping. A zero-mean Gaussian noise with a standard deviation of 0.05 is applied[1] over the network inputs. Secondly, dropout, a widely adapted technique introduced by Hinton et al. [28], is used between every layer. L2 regularization is applied on all layers, with a value of 1×10^{-4}. Finally, batch normalization, introduced by Ioffe et al. [33], is used at several points throughout the networks. Using batch normalization primarily accelerates training, but can also help with regularization.

During prototyping several additional methods were explored. Data augmentation in the form of skeleton mirroring to increase the training sample size was tested. This, however, only showed worse generalization performance and was not used. The expected cause of this decrease was the fact that gesture recordings were made at a slight angle, which meant that there was no symmetry in the skeletons due to perspective. Additionally, the Focal loss function of Lin et al. [44] was tested briefly, which aims to address class imbalances. However, this yielded no performance increase and increased the time to converge and was therefore omitted.

5.3.2.2 Training

Evaluation is done using a cross-actor evaluation protocol, where one actor is used for testing and the remaining four for training. For every network, each actor is evaluated, meaning each network is trained and tested 5 times. Early stopping is applied on the test loss with a patience of 10 with a maximum of 150 epochs. Again, this will be discussed further in Section 6.3.2. A batch size of 150 is used. The loss function used is binary cross-entropy, as discussed in Section 3.5. The decision threshold was set at 0.5. Tuning of this threshold is possible to achieve a higher recall/precision at the trade-off of lower performance in the other. In practice, 0.5 proved to be the right balance point, which makes sense given it is the middle of the sigmoid curve.

The Adam optimizer of Kingma and Ba [37] is used again for GPGD. Learning rate was set at 1×10^{-3} with the default β parameters. Training is run in the Google Colab environment, which provides free virtual machines with workstation grade graphic cards for educational

[1]For scale reference, the scaled OpenPose skeletons are roughly 3.0 in length top to bottom and JL_D ranges from 0 to 1.6. This measure is unitless, but can be interpreted as the amount of spine lengths.

purposes. Training time per actor evaluation averaged around 1-1,5 hours, for a total of around 5-6 hours per network.

5.3.2.3 Results

Tables 5.3 and 5.4 show the summarized numerical performance metrics of the configurations evaluated on GPGD. These tables give a numerical indication of overall performance on the metrics outlined in Section 5.1. The ROC-curves of the configurations are presented in Figure 5.14 through 5.16, which give a visual indication of individual and overall class performance. Table 5.5 contains an in detail overview of precision, recall and F1-score per actor and class. It is important to note that in this table, these metrics are based on the binary predictions with the decision threshold at 0.5.

The performance difference for the DBRNN-L classifier with and without the joint-line distance feature representation is much smaller than it was for Berkeley MHAD. Overall, performance can be considered slightly improved, however the average recall is worse with joint-line distance. Noteworthy is that both DBRNN-L classifiers have quite similar Hamming losses as the more advanced classifiers, which perform better on most other metrics. Considering the definition of the Hamming loss, this means that the distance of the predicted labels to the correct labels is similar/better than the other classifiers. A low Hamming loss correlates to low false positive count considering each prediction has 9 labels, which is also reflected in the highest precision score of JL_D-DBRNN-L. However, the Hamming score does not penalize a low recall as heavily and fails to capture the low recall that is shown in Table 5.3.

In general, the JL_D-GRU configuration shows the best performance across the board. It has the highest hard accuracy score by a noticeable margin and performs well in label ranking average precision. Furthermore, classification performance shows a good balance between precision and recall. JL_D-LSTM also shows improvements over the baseline, but by a smaller margin.

Classifier	Feature representation	Accuracy	Hamming loss	mAP	Label ranking average precision
DBRNN-L	Coordinates	0,400	0,125	0,658	0,781
DBRNN-L	Joint-Line Distance	0,401	**0,109**	0,692	**0,814**
JL_D-LSTM	Joint-Line Distance	0,439	0,128	0,666	0,757
JL_D-GRU	Joint-Line Distance	**0,490**	0,113	**0,700**	0,801

Table 5.3: Performance metrics of the different configurations evaluated on GPGD. All metrics are macro-averaged by class and actor.

Network	Skeletal representation	Average precision	Average recall	Average F1-score
DBRNN-L	Coordinates	0,648	0,565	0,580
DBRNN-L	Joint-Line Distance	**0,727**	0,512	0,569
JL_D LSTM	Joint-Line Distance	0,621	0,650	0,608
JL_D GRU	Joint-Line Distance	0,665	**0,658**	**0,640**

Table 5.4: Results on GPGD, comparing several classifiers. All metrics are macro-averaged by class and actor and based on the binary predictions. Note that the average precision here is the average of class precision, rather than the mean Average Precision metric.

	DBRNN-L			JL_D DBRNN-L			JL_D LSTM			JL_D GRU		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Actor 1	0,71	0,51	0,57	0,75	0,47	0,53	0,65	0,57	0,59	0,70	0,60	**0,63**
Actor 2	0,67	0,54	0,56	0,79	0,52	0,58	0,58	0,63	**0,59**	0,58	0,60	0,58
Actor 3	0,66	0,62	0,62	0,72	0,55	0,61	0,66	0,71	0,66	0,74	0,72	**0,72**
Actor 4	0,49	0,44	0,44	0,60	0,39	0,45	0,47	0,52	0,46	0,55	0,50	**0,47**
Actor 5	0,71	0,71	0,70	0,78	0,62	0,67	0,74	0,81	0,75	0,76	0,86	**0,80**
waveN	0,55	0,35	0,43	0,71	0,33	0,44	0,55	0,63	**0,57**	0,60	0,52	0,54
waveE	0,54	0,40	0,43	0,71	0,19	0,30	0,57	0,55	0,51	0,63	0,59	**0,56**
waveS	0,44	0,28	0,32	0,43	0,19	0,23	0,40	0,34	0,33	0,40	0,40	**0,38**
waveW	0,68	0,62	0,59	0,81	0,48	0,57	0,52	0,78	0,60	0,70	0,60	**0,63**
blockN	0,63	0,57	0,58	0,72	0,51	0,59	0,68	0,58	0,61	0,69	0,62	**0,63**
blockE	0,81	0,66	0,72	0,80	0,79	**0,79**	0,76	0,70	0,72	0,83	0,78	0,78
blockS	0,55	0,56	0,54	0,69	0,61	0,63	0,61	0,55	0,57	0,63	0,70	**0,65**
blockW	0,84	0,85	0,84	0,85	0,76	0,79	0,84	0,86	**0,85**	0,80	0,88	0,83
Other	0,79	0,79	**0,79**	0,83	0,74	0,78	0,64	0,86	0,72	0,70	0,82	0,75
macro avg	0,65	0,57	0,58	**0,73**	0,51	0,57	0,62	0,65	0,61	0,66	**0,66**	**0,64**
weighted avg	0,70	0,62	0,63	**0,76**	0,60	0,64	0,68	0,67	0,65	0,71	**0,70**	**0,69**

Table 5.5: Precision (P), recall (R) and F1-score (F1) per actor and class, averaged over all 5 actors based on the binary predictions. The Highest F1-score per class is bold. The last two rows report the macro and weighted average of all classes. Here the highest average precision, recall and F1-score are again bold.

Table 5.5 allows for direct comparison of actor scores and the ROC-curves provide a more visual impression of the results. The cross validation scores are a measure of how well a classifier trained on gestures by the other four actors works on this unseen actor. Uniform trends can be seen across all classifiers. Evaluation on actor 5 achieves the best results, followed by actor 3, while actor 4 performs consistently bad. This is also reflected in the ROC curves of this actor, which for some classes actually move below the diagonal and are thus performing worse than random guessing at certain points.

Another observation that can be made about the actor scores is that increases in performance are largest in well performing actors. The difference in highest and lowest F1-score for actor 5 is 0,13 (from 0,67 to 0,80), compared to 0,03 for actor 4 (from 0,44 to 0,47). Comparing the ROC curves also illustrates again the performance differences among actors. The curves for actor 3 and 5 are grouped nicely towards the ideal upper-left corner, while the other actors curves are much more scattered. This is especially evident in the ROC curves of JL_D-GRU in Figure 5.16.

Looking at scores per class again show consistent trends across all configurations.
’BlockE’ and ’BlockW’ are well classified, where ’BlockS’ and ’BlockN’ are more difficult to classify. Waves are typically performing worse than block gestures. ’WaveS’ especially is scoring quite low. This is also reflected in the ROC plots of ’WaveS’, in which they typically trend the least to the upper-left corner.

The larger classifiers of JL_D-LSTM and JL_D-GRU show better classification performance especially for the waves gestures, compared to DBRNN-L, which scores low for these classes. Performance increase for the block gestures is not as big.

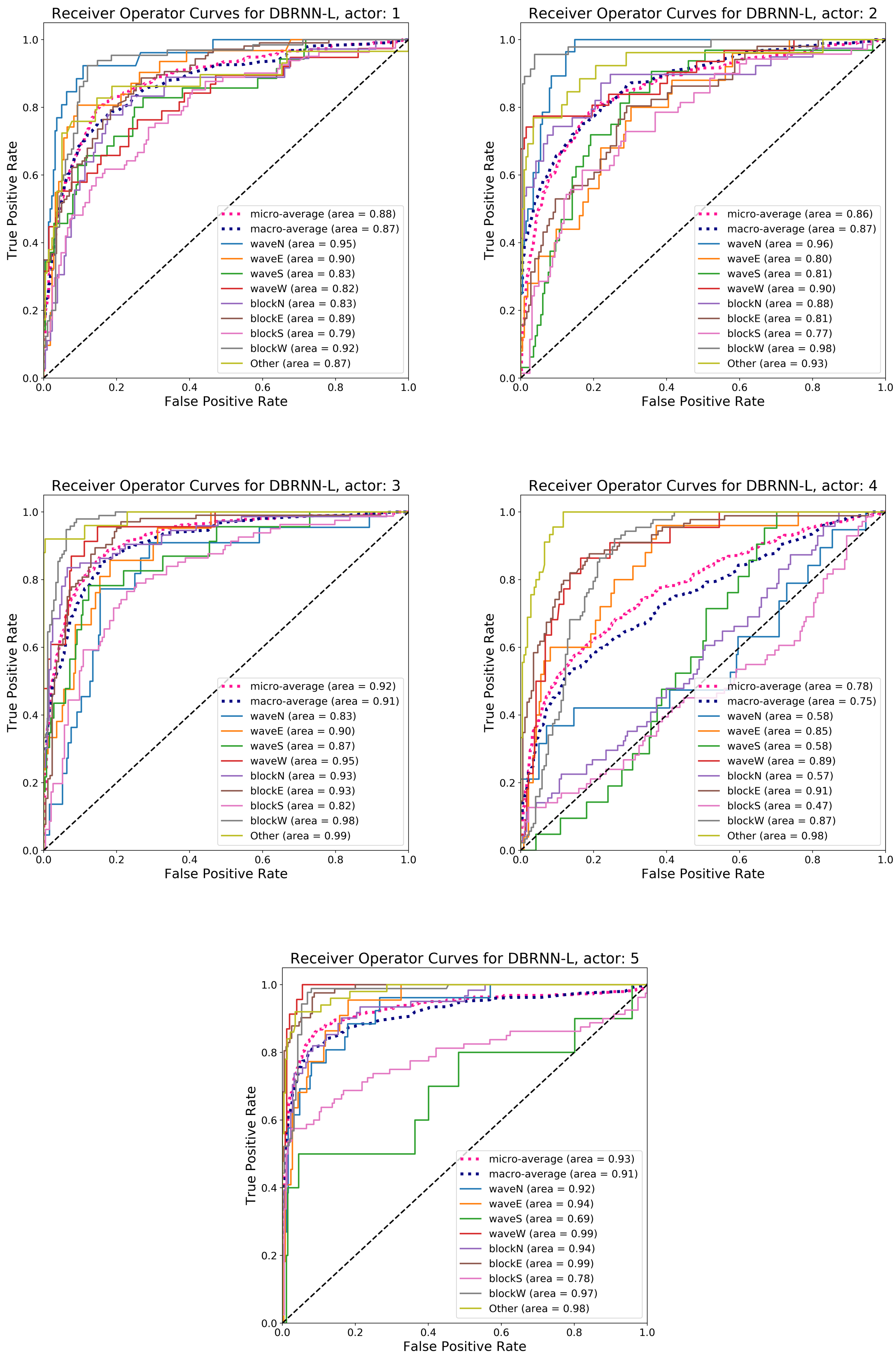


Figure 5.14: Receiver operator curves for DBRNN-L per actor

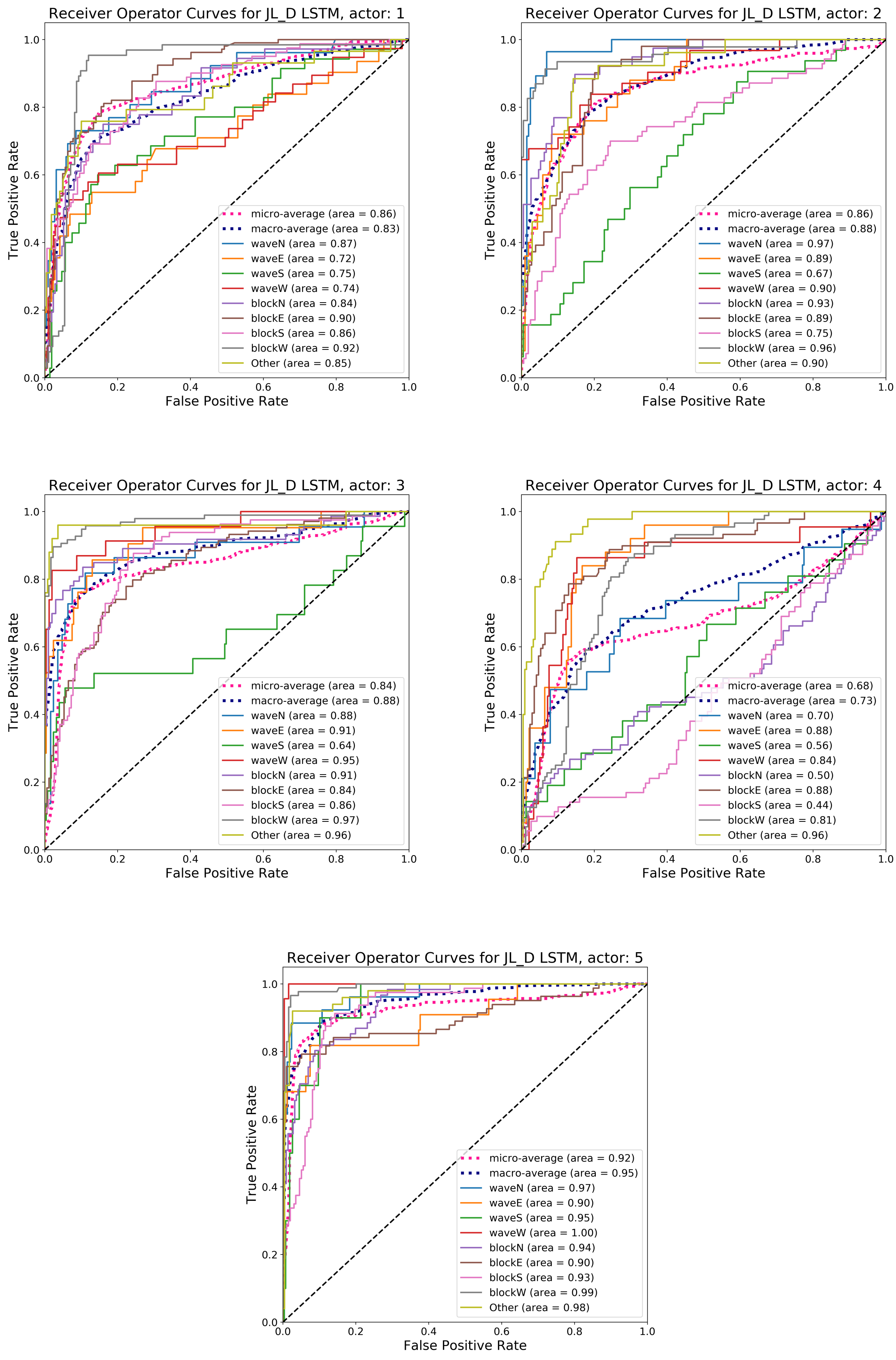


Figure 5.15: Receiver operator curves for JL_D LSTM per actor

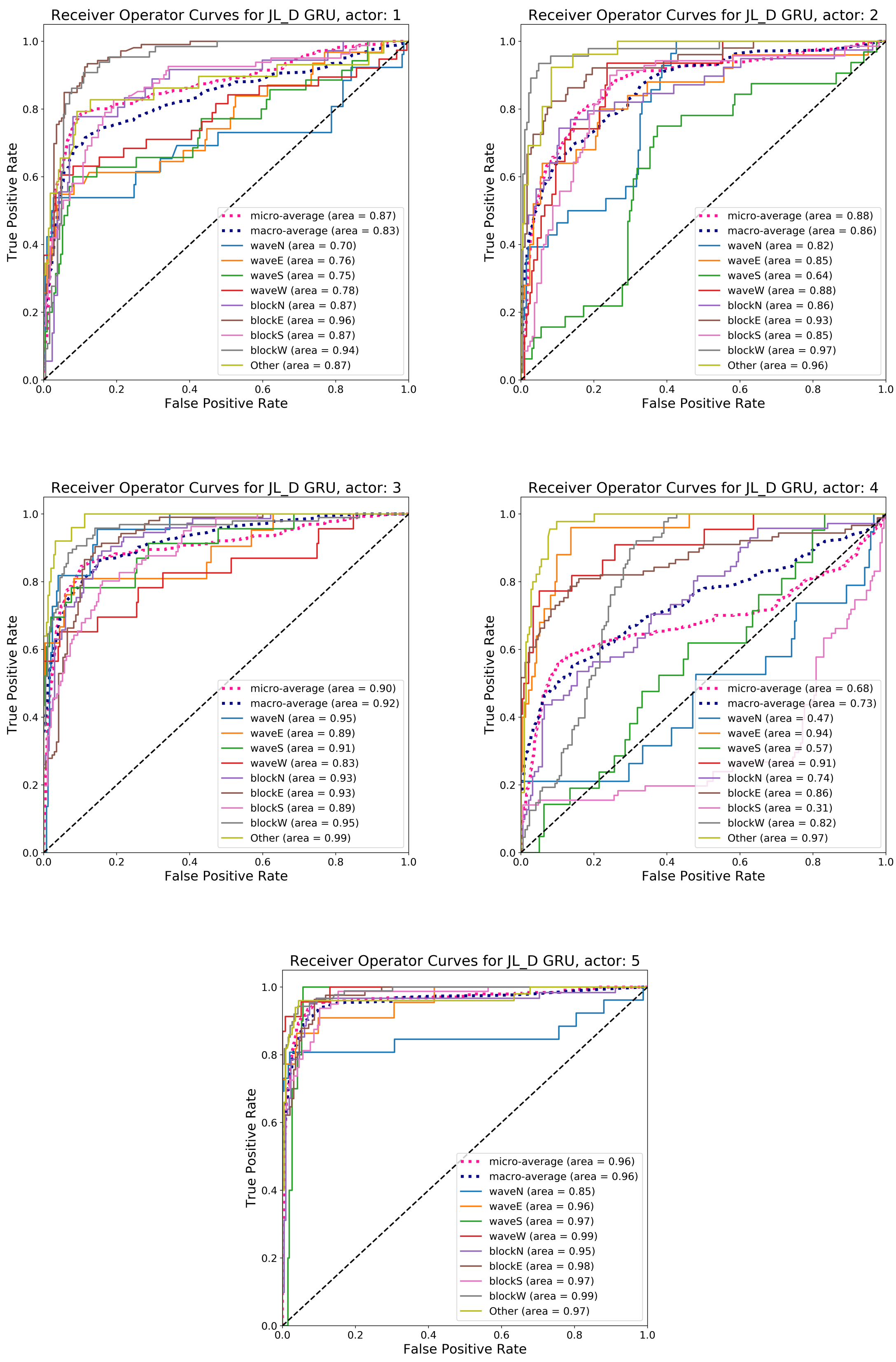


Figure 5.16: Receiver operator curves for JL_D GRU per actor

5.3.3. Cross-viewpoint evaluation and transfer learning

The robustness of the system to viewpoint variation and the potential of transfer learning is explored in this experiment. This is done using the labeled subset of the front viewpoint from the Greenscreen Police Gesture Dataset. All four configurations tested in the previous section are again evaluated.

First, the viewpoint robustness is evaluated by using the models trained on the complete viewpoint in the previous section on the new viewpoint without any adaptations. Comparing the numerical results to the results achieved in the previous section gives us an indication of how robust the classification system is to variations.

Secondly, the potential of transfer learning for traffic gestures is explored. This is done by comparing a classifier trained from scratch only on the front viewpoint to a classifier that uses the weights from the previous section as starting point before training. Two different transfer learning set-ups are evaluated.

5.3.3.1 Training

A training setup identical to the previous experiments is used for the cross-viewpoint evaluation. Again, cross-actor evaluation is used and the actor split is maintained to the new viewpoint.

For the transfer learning, two transfer schemes are evaluated. Opening all layers to retraining and only allowing the last two layers to be retrained. The argument for only retraining the last two layers are that the lower layers in a network in theory are uniform lower to higher level features that don't have to be retrained for similar data, a practice that is common with large image classification networks. This leaves much fewer parameters to be optimized, which is attractive given the limited sample size. However, by fixing the weights of such a large part of the network, the classifier might underfit to the data. Therefore, it is compared against optimizing all layers in this experiment.

5.3.3.2 Results

Table 5.6 shows the results of the cross-viewpoint evaluation and transfer learning experiment. The cross-viewpoint evaluation achieves results very similar to classifiers trained from scratch for all configurations. Performing similar to networks that have been trained on that viewpoint specifically is a good result and testament to the ability of the networks to generalize. However, the training data for these classifiers was substantially less, which is something to take into account during this evaluation.

Furthermore, the results on the front viewpoint lack behind compared to those achieved in Table 5.3, especially for accuracy. However, as explained in Section 5.1, accuracy gives a skewed representation of classifier performance for multi-label classification. Given that the average precision and LRAP are still quite close to the original viewpoint, we can conclude

Classifier	Transfer configuration	Accuracy	Hamming loss	mAP	Label ranking average precision
DBRNN-L	CV evaluation	0,27	0,17	0,55	0,68
	Train from scratch	0,24	0,15	0,58	0,72
	Transfer - retrain 2	**0,35**	**0,13**	**0,62**	**0,76**
	Transfer - retrain all	0,33	0,14	0,61	0,74
JL_D DBRNN-L	CV evaluation	0,29	0,15	0,58	0,70
	Train from scratch	0,31	0,14	0,59	0,76
	Transfer - retrain 2	**0,35**	0,12	**0,68**	0,78
	Transfer - retrain all	0,32	**0,11**	0,64	**0,80**
JL_D-LSTM	CV evaluation	0,27	0,18	0,54	0,62
	Train from scratch	0,28	0,16	0,60	**0,74**
	Transfer - retrain 2	0,33	**0,14**	**0,63**	0,71
	Transfer - retrain all	**0,36**	**0,14**	**0,63**	0,73
JL_D-GRU	CV evaluation	0,31	0,17	0,55	0,62
	Train from scratch	0,30	0,15	**0,62**	0,74
	Transfer - retrain 2	0,31	0,15	0,59	0,70
	Transfer - retrain all	**0,38**	**0,12**	0,61	**0,76**

Table 5.6: Viewpoint variation and transfer learning results.

that viewpoint variation does introduce more noise in the predicted labels.

Both transfer learning schemes show improvement over classifiers trained from scratch. For the smaller network of DBRNN-L, only retraining two layers gives the best result. In contrast, for the larger networks of JL_D-LSTM and JL_D-GRU opening all layers to training achieves better results. Transfer learning shows performance gains predominantly in accuracy and Hamming loss. This means that the overall quality of the output labels produced by these classifiers is more accurate, considering the nature of these metrics and noise is reduced.

Comparing the results presented in Table 5.6 to those in Table 5.3 shows that classification performance on the front viewpoint generally performs slightly worse compared to the side-front viewpoint of Section 5.3.2. As expected, this viewpoint is more difficult to classify due to the presence of more occlusions.

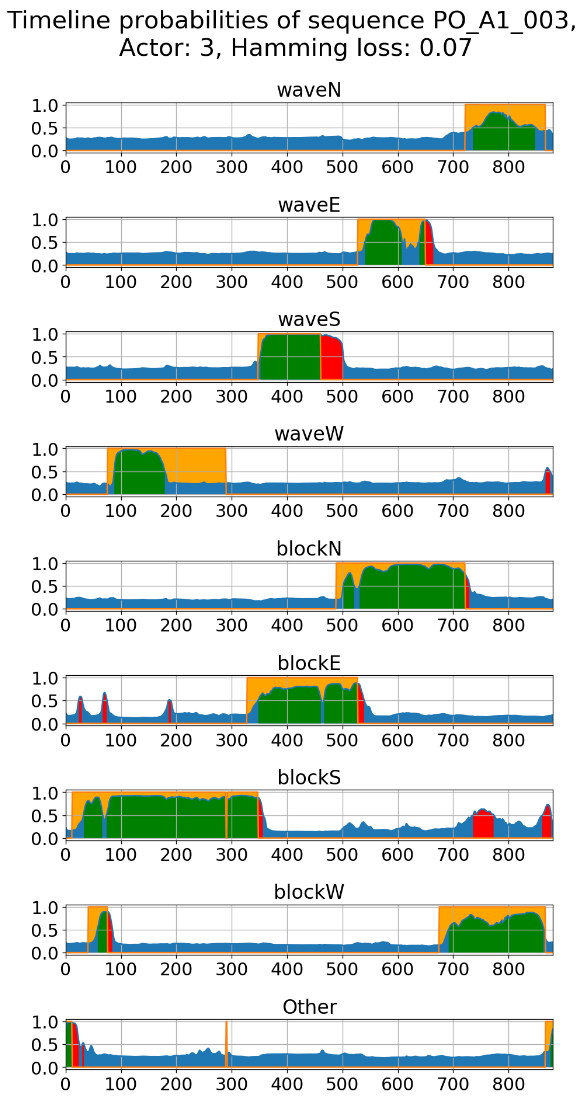
5.3.4. Generalization to continuous data and video analysis

Training and evaluating the networks on GPGD in the previous sections was done using split sequences. However, in a real-world scenario, data will not be presented in segmented sequences, but in a continuous stream. This experiment will evaluate the generalization of the best gesture recognition system from the previous sections (JL_D-GRU) to continuous input streams.

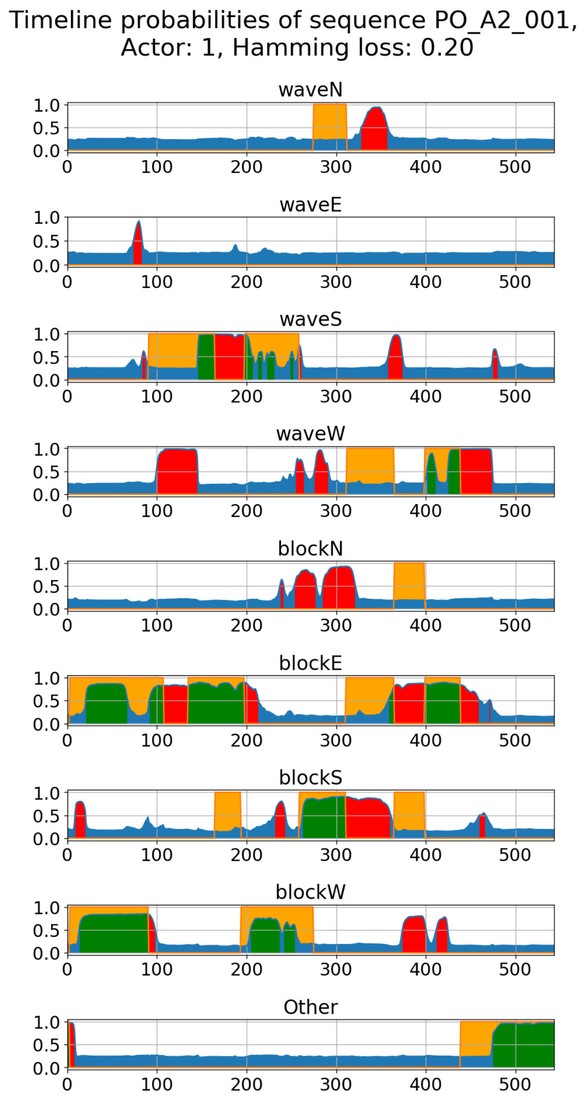
Evaluation is done on the same gesture sequences from GPGD, however in their complete and unsegmented form. As the network is structured in a many-to-one way, where a sequence results in a single labels prediction, these sequences have to be fed to the network in the right way. Input skeletal features are fed to the network using a sliding-window principal, where the last 60 frames are fed to the network, equal to roughly 3 seconds. This means that the label at frame y_t is based on the features from x_{t-60} to x_t. This window cutoff is used as features from more than 60 frames ago are not relevant to the current prediction and memory use increases significantly for very long sequences. Zero-padding is used for the first 60 frames.

Visualization of the output probabilities over time are presented in Figure 5.17. Figure 5.17a is an example of a well classified sequence. False positives are limited and the classifier responses quick to gestures. Furthermore, as can be seen from the label distribution over the sequence, actor 3 has a much more structured gesture style compared to actor 1 from Figure 5.17b. Figure 5.17b shows that the classifier struggles to respond to the quickly changing gestures and also confuses gestures. Quick transitions show greater difficulty and errors.

Figure 5.18 show frames from the video sequences linked to Figure 5.17. The image sequence in the top row shows an accurate classification of the gestures occuring, while the actor is facing away from the camera. The failure to classify 'WaveE' around frame 600 can be attributed to the prolonged occlusion of the right arm. The second image sequence in Figure 5.18 shows confusion between South and East in the first frame. Additionally, a delay in output is observed between raising the arm and classification in the final two frames. The third image sequence explains the struggles of the classifier seen in the timelines. The actor is waving to the north, however the waving is mainly done using the hand and wrist. Therefore, the classifier mistakes it for a block gesture, as the hand and wrist are not included in the skeleton. Furthermore, this sequence is affected by the speed-up issue mentioned in Section 4.1, which causes the perceived execution speed of the sequence to be higher. This could also further complicate classification of the wave gesture. Additionally, due to the positioning of the actor and arm, confusion is seen with 'WaveW'.



(a)



(b)

Figure 5.17: Timeline probability visualization per class of two gesture recordings using a sliding window of 60 frames and JL_D GRU. Ground truth labels are presented by orange, correct predictions are green, incorrect red. X-axis is the frame count. Included also is the Hamming loss over the entire sequence.

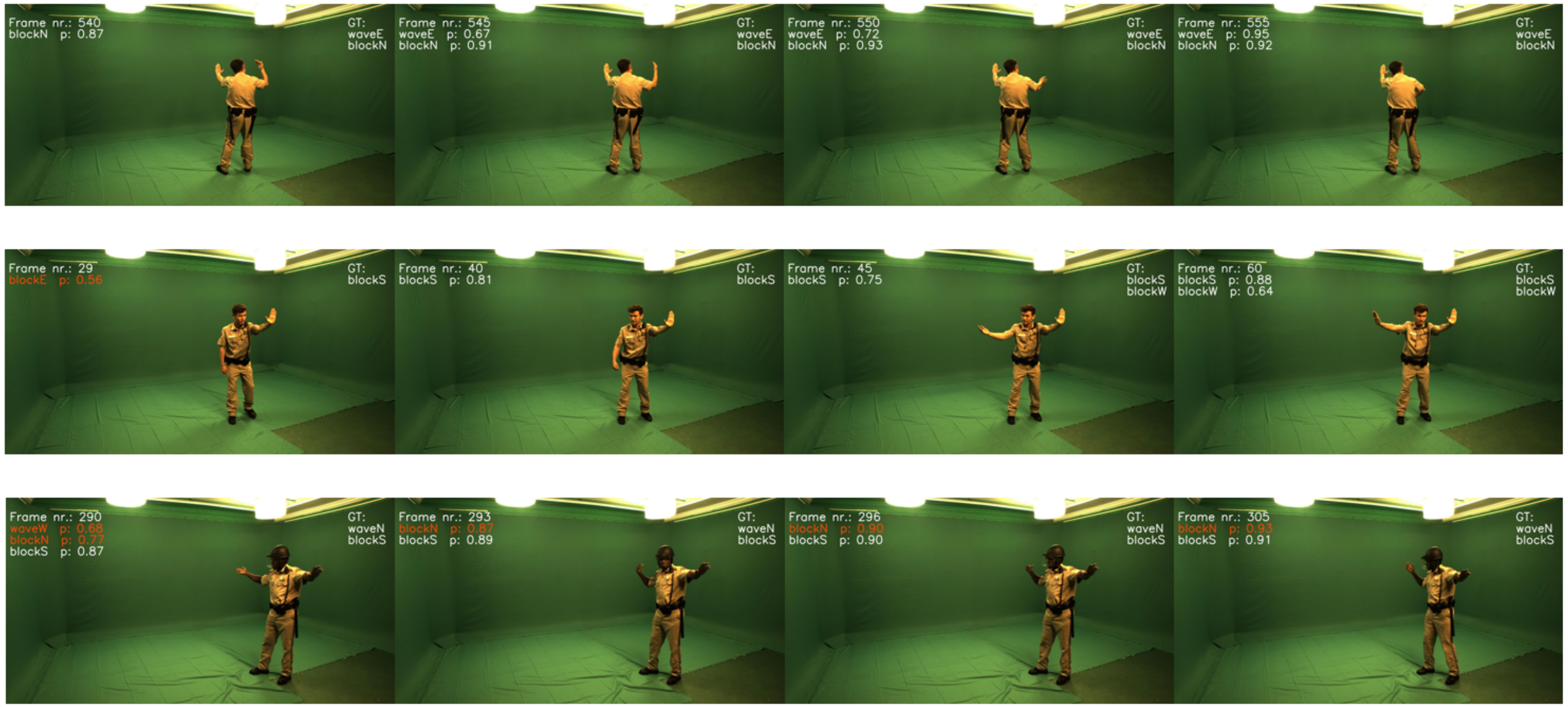


Figure 5.18: Video still frames taken several frames apart with corresponding output probabilities and ground truth, linked to the timelines of Figure 5.17. Text on the left shows frame number the output of the network, with correct predictions in white and incorrect in red. Groundtruth labels are shown on the right. Images on the top row show correct classification, even during brief occlusion of the right hand. First frame on the second row shows confusion in direction between East and South, the third frame shows detection delay of the wave gesture. The third sequence shows confusion between Wave and Block in the North direction, likely due to the hand waving instead of the whole arm.

5.3.5. Computational efficiency

For real-world use, the runtime of a gesture recognition system has to be fast enough for real-time processing of gestures. As we currently lack a complete real-time completely integrated application of the gesture recognition pipeline, measurements are made of the primary subsystems in order to asses the speed of the system. We compare the largest network, JL_D-GRU, to the smallest network, JL_D-DBRNN-L. For OpenPose, no separate measurements are used. OpenPose speed is highly dependent on the detection configuration and resolution. Depending on accuracy and speed trade-offs, a range between 4 and 15 fps is possible. We use the reported 6,8 frames per second processing speed using the net resolution of 656x368 in this calculation, which is a balance between accuracy and speed.

All time measurements are done using the same laptop equipped with a i7-8750H 6 core CPU and a NVIDIA Quadro P1000 GPU. The computation time per prediction is calculated by processing a batch of frames and using the elapsed system time to estimate the time per prediction. This is done using the same sliding window approach on an entire sequence as introduced in Section 5.3.4, meaning every evaluation processes 60 frames for a single prediction. Model inference of the classification architectures was done using a batch size of 1, similar to frame by frame processing. The first 15 gesture sequences are processed this way, frame by frame. The final average processing times are reported in Figure 5.19, including the found standard deviation in the measurements. The current set-up does not measure any other code overhead, but is also not representative to real-world overhead in that regard anyway. The reported OpenPose computation time is the processing time for pose estimation of the latest frame.

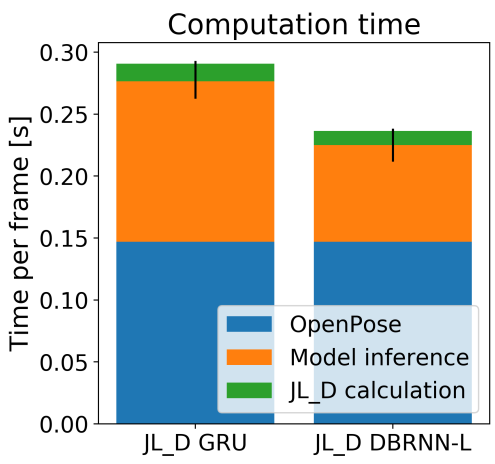


Figure 5.19: Comparison of Computation time. Error bars denote standard deviation of time measurements with N=15 complete sequences measured.

Computation time for JL_D-GRU is on average 0.290 seconds per frame (3.44 fps). For JL_D-DBRNN-L on average 0.236 (4.23 fps) is measured. As found in Figure 5.19, OpenPose pose estimation and model inference make up a majority of the computation time. JL_D feature calculation is much smaller, and quite fast. For these measurements, it should be noted that the computations are being done on a laptop. OpenPose performance is highly GPU dependent an can achieve much higher speeds with the right hardware. The same applies to a lesser extent to the network inference times for both configurations. Noteworthy is that the inference time for JL_D-DBRNN-L is comparatively not much faster than JL_D-GRU, given their respective network and parameter sizes.

Overall performance shows that these methods have the ability to translate to real-time classification. Given the hardware used for evaluation, acceptable speeds are achieved. For proper real-time use, however, performance of at least 20 fps is desirable.

6 Discussion

Chapter 5 presented all experiments executed and results found in this work. This chapter will discuss the interpretation and implications of these results with regards to the research questions posed in Chapter 1.

6.1. OpenPose as feature extraction for traffic gesture recognition

The main interest regarding the research question formulated around OpenPose was to evaluate the expected performance gap between image based skeleton estimation and the maximum potential in information captured by skeletal features recording using motion capture technology. The goal was then to investigate how much this gap can be minimized using the right skeletal feature representation.

Based on the findings presented in Section 5.3.1, it seems that OpenPose functions well as feature extraction methods for action/gesture recognition, provided that the right feature representation is used. Using OpenPose with coordinate representation works well for classes with large lateral movements, but performs significantly worse than the motion capture baseline for classes that contain more depth, such as 'boxing' or 'clapping hands'. For application in self-driving cars, gestures towards the car are both important and occur often. Poor performance in this direction is therefore worrisome.

When using OpenPose with the joint-line distance skeletal feature representation, this shortcoming is addressed. Performance of this set-up was significantly better for classes containing a lot of joint overlap and where the joint relations were important. The fact that this set-up comes very close in performance to motion capture data is a very promising result, both for potential classification performance, but also for the flexibility it allows when generating training data for gesture recognition. The accuracy of these methods allow for the recording of training data in situations outside studio settings.

The fact that 2D skeleton action recognition performs so similar to 3D skeleton action recognition is another noteworthy finding. The field of 3D skeleton action recognition has been explored greatly, however 2D skeleton action recognition has received marginal attention in comparison. The fact that both the 2D motion capture data and OpenPose with JL_D perform so similar to 3D motion capture data provides good perspective for extending skeleton based action recognition to self-driving cars. Being able to perform gesture recognition purely on 2D images greatly extends the operating domain of the classification system and validates the viability of classifying gestures based on a video feed.

The accuracy results shown on Berkeley MHAD in this work are very high (up to 0,9855), which is consistent with the results seen in literature, where a multitude of works have achieved perfect scores. Perfect classification scores typically arise suspicion and several explanations can be given for these results. First, it is a very synthetic and carefully fabricated dataset. It is recorded in a controlled studio setting with very consistent and controlled action definitions. This makes generalization to the test set simple, but arguably unrepresentative of real-world challenges. Additionally, as this test set is the only measure for performance and a validation set is absent in the evaluation protocol of Ofli et al. [51], some extent of overfitting on the test set is expected. This is discussed in more detail, also regarding GPGD, in Section 6.3.2. Finally, the actions contain some landmark positions, to which the network can easily fit the action, rather than using the action sequence itself. An example can be found in the throw ball class: every subject ends this recording in the relatively same position of extended arm after throwing for an extended time. The network could base the prediction purely on this final pose, rather than use the movement of the throw itself. This would generalize poor to real-world applications.

The performance gain observed while using joint-line distance feature representation is much smaller for GPGD in Section 5.3.2.3 compared to Berkeley MHAD. Given that the camera perspectives used are very similar and the same classifier was used, the most likely reason for this difference lies in the gesture execution. As discussed in Section 5.3.1, Berkeley MHAD struggled with distinguishing actions that existed in the same location in space and relied heavily on the skeletal relations. The JL_D representation provided a very good solution for this specific problem. However, the gestures in GPGD show less spatial overlap compared to Berkeley MHAD and are spread over four orientations. Furthermore, the traffic gestures are less dependent of other joints for their definition than 'boxing' and 'clapping with two hands', in which the position of the hands w.r.t each other are essential for separation.

An important consideration about the way OpenPose was implemented in this work is that there is only one keypoint for the end of the arm, namely the wrist. This means that subtle hand movements and flicks of the wrist are not captured in the skeletal data. These subtle movements can contain significant and useful information for the classifier. The OpenPose framework does have a more extended face and hands detection system based on the work of Simon et al. [59], but this works mainly up close and cannot be expected to work well from far away in real-world scenarios. Additionally, adding an entire hand skeleton with a large number of joints contains redundant input data for the purpose. Instead, an additional keypoint could be added to the skeleton at the top of the fingertips and/or using the palm. This would add hand gesture information, without introducing too many new dimensions or becoming undetectable in realistic scenarios. Similarly, sequences where a baton was used encounter the same issue, as OpenPose does not detect these objects, which would also be a problem for agents using stop signs.

Another finding from the experiments of Section 5.3.1 can be found in the sequence visualizations presented. These sequences illustrate some important caveats of evaluating the performance of classifiers on segmented sequences. Numerical assessment of the metrics and loss calculation is done only based on the final label assigned to the sequence, which does not capture the accuracy on previous frames in the sequence. The sequence in Figure 5.13 is missclasssified for roughly 73% of the frames, yet as the final label is correct, it is counted as good classification. Furthermore, as the loss of the sequence is also calculated based only on the final frame prediction, the network has little incentive or guidance to increase accuracy in the earlier frames. This means that the response time to actions is not penalized or optimized for. This part of sequence evaluation is often overlooked when working with segmented sequences in the field of (skeletal) action recognition, yet is important for traffic gesture recognition, where response time and the accuracy during the sequence are essential. While these experiments were evaluated on Berkeley MHAD specifically, this limitation is an inherent consequence of the combination of using segmented training samples and the many-to-one network configuration. This limitation is thus something to also keep in mind while using this set-up on other datasets, such as GPGD.

6.2. Greenscreen Police Gesture Dataset observations

This work conducted the first experiments on the Greenscreen Police Gesture Datasets [3]. As such, the results of the experiment in Section 5.3.2 present a wide range of interesting findings and identification of challenges. All methods showed difficulties with the same challenges. Overall, wave gestures performed poorer compared to block gestures. Additionally, large intra-actor performance gaps are seen. Both of these challenges are in-line with the expectations stated in Chapter 1.

The intra-class performance variation between block and wave are closely connected to the challenges outlined in Chapter 1. Wave gestures contain a broad range of execution speed, style and length. In comparison, block gestures are typically either short extension of an arm or keeping an arm statically extended. There is much less temporal information that has to be classified in these block gestures, typically only the general pose. The difference in classification performance is visible across all classifiers. The execution of wave gestures showed a much larger variation than the block gestures in the dataset among the actors. Examples of which can be found in Figures 6.1 and 6.2.

Noteworthy are the especially poor results when evaluating actor 4 across all classifiers. The performance gap shown here is significantly large compared to the other actors. The hypothesis for this performance gap is twofold. First, this actor consistently positioned himself on a different angle with respect to the camera compared to the other actors. This meant

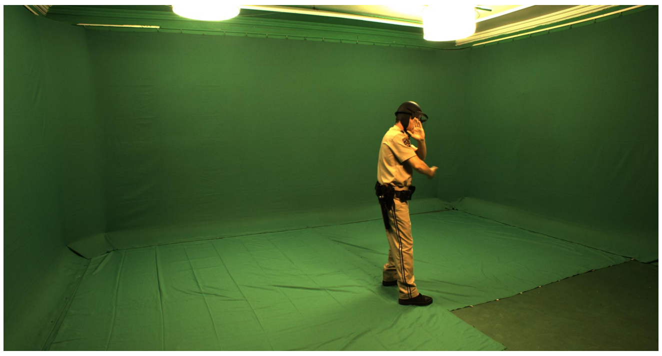


Figure 6.1: Example of a hard gesture of actor 3. The correct label in this case is 'BlockW', 'WaveS'.

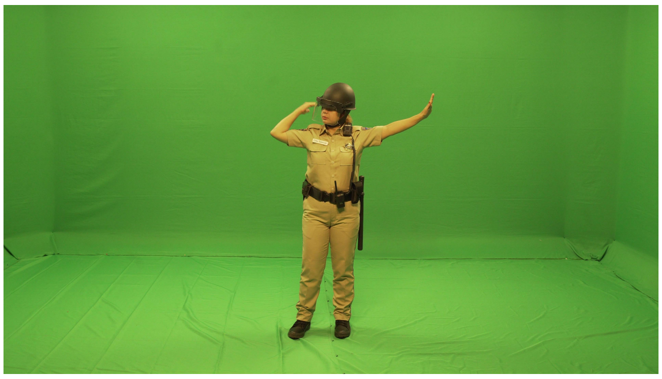


Figure 6.2: Example of clear gestures from actor 5, seen from the front. Gesture is 'BlockE', 'WaveW'.

that the gestures were executed at slightly different angles with regard to the primary gesture directions that were defined. This means that the labels and features do not contain exactly the same correlations as for the other actors. This hypothesis is supported by the significantly lower performance hit found for the initial viewpoint variation comparison, when looking in depth at the per actor metrics found. Figure 6.3 shows the difference in actor positioning.

The second expected cause of the poorer performance is the execution style of the actor. While this cannot be objectively quantified, the gesture execution style of actor 4 could be considered more vague and at times ambiguous compared to the other actors. For comparison, actor 5 was extremely clear in gesture execution and obtained the highest classification scores consistently. This exact problem was expected and something that a good traffic gesture recognition system needs to be able to address. However, the fact that one of the actors performed all gestures at different angles compared to the other actors does seem to introduce undesired noise in the dataset. A way to resolve this issues, would be to switch from a studio frame of reference to an agent frame of reference for the labels to achieve invariance to agent orientation with respect to the camera. In such a label set-up one would label the orientation of the agent and the direction of the gestures with respect to that orientation.

An important note on the gestures present in the Greenscreen Police Gesture Dataset is that there is limited overlap with the officially defined gestures found in Chapter 2. The gesture styles portrayed are executed in a more natural and intuitive manner, compared to the strict

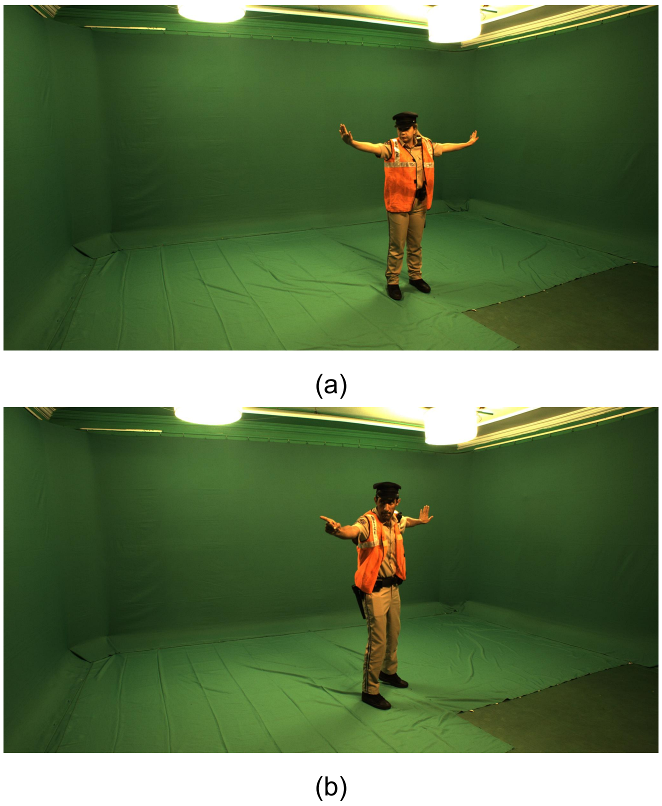


Figure 6.3: Disparity of actor orientation for the exact same gesture. (a) shows typical angle of actor 5, (b) shows actor 4's angle. This orientation disparity is consistent through all recordings and estimated to be in the range of 30-45 degrees.

guidelines outlined by legislation. This is an accurate reflection of reality, including the wide range of gesture execution. It is, however, something important to keep in mind during further development.

6.3. Classification performance

The methods explored in this work have established good performance on existing action recognition datasets. The main question posed was how well this performance transfers to the problem of traffic gesture recognition. A detailed analysis of the classification performance of the used methods was presented in Sections 5.3.2 and 5.3.4. Based on these results, the JL_D-GRU configuration showed the best performance. Given the overall findings, it seems that GPGD responds better to deeper networks, such as JL_D-GRU, rather than broad and shallower nets as used in JL_D-LSTM. The ablation study of Maghoumi and LaViola [46] showed that their DeepGRU network responded well to the addition of 2 fully-connected layers between the recurrent section and classification layer. This could very well be the case for this dataset as well. Use of GRU units showed their practical value by allowing for the creation of larger networks for similar training parameter counts. The multi-label outputs of the network proved to be an effective design choice.

The inclusion of more LSTM/GRU layers compared to the DBRNN-L architecture seems to help especially for classification of the wave gestures. This validates the expectation that the classification of traffic gestures is highly dependent on the way the temporal domain is handled. Additionally, the GRU network which used bidirectional layers shows improvement compared to the LSTM network which lacked bidirectional layers. Of course, direct comparison here is ambiguous as several other factors were changed across these networks, but bidirectional layers are already a well established concept for handling temporal data. This still suggests that handling of the temporal data is essential and could be a focus point for further work.

Curious is the fact that the larger networks primarily gain performance in already well performing actors. A likely explanation for this is that a clearer execution of gestures results in more easily separable samples, to which the larger networks are then able to fit. More vague gestures are then expected to be less separable and the larger networks are still unable to correctly separate them. Two strategies could potentially address this, either add more training data or make the samples more separable by the addition of the right temporal or spatial features or manipulation. Feature manipulation in the form of JL_D skeletal representation already showed its merits in the results of Section 5.3.1.

Evaluation on continuous gesture sequences combined with video analysis showed that the classifiers struggled with both missclassifications, and transitions of gestures. The missclassifications are closely related to the issues outlined above. Lack of hand information can explain most errors made in wave classifications. Additionally, the large range of execution styles make waves hard to classify. Confusion in direction can be attributed to difficulties with estimating the orientation of limbs based on only 2D information. Errors in this regard were more apparent when joints were located in-between the logical extremes (bent v.s. stretched arm). In these regions there is more overlap between gestures, making these spatial regions troublesome for classification. This is seemingly a contradiction with the findings discussed in Section 6.1. However, that experiment only evaluated the ability to classify the action class, rather than both class and direction at the same time. It seems that 2D skeletons capture enough information for class identification, but struggle comparably more with direction identification. Exploration of additional feature representations, specifically aimed at capturing the direction, could help here. Finally, occlusions are a large contributor of classification errors. Frames with occluded and missing detections showed consistently poorer classification results.

The difficulties and late responses shown during transitions could be linked to the segmentation of the training data. As the classifiers were trained on segmented gesture recordings, they have no notion of how gesture transitions happen in continuous gesture streams. Exploration of different training set-ups where transitions are included could offer better results in this regard. An example would be to switch to a many-to-many network configuration, which showed good results in the work of He et al. [26], where each frame is assigned an output label, rather than a single label to a sequence of frames. This allows for training on transition points explicitly.

The impact of the dataset imbalance is hard to quantify. As mentioned, segmentation of gestures creates an over-representation of block labels in the samples, compared to wave labels. Dataset imbalances for multi-label problems are especially hard to address, as conventional techniques such as under- or oversampling don't work when samples can have multiple labels. Addressing this was briefly explored by using the focal loss function, but this offered no clear benefit and slowed convergence during training. Alternatively, removing the gesture segmentation from the training pipeline would remove the imbalance almost entirely. This can be combined with a many-to-many network configuration as discussed before.

The relatively low numerical scores achieved on GPGD seem to suggest that it is a difficult dataset, considering the high scores previously achieved by the methods used on other action recognition datasets. However, most datasets limit themselves to class classification, whereas the gestures present in GPGD also require determination of the right orientation. It should be noted that direct comparison of metrics (such as accuracy) of multi-label and multi-class classification datasets is not possible for the reasons outlined in Section 5.1. However, given the performance achieved and shown in video analysis, this work provides a good starting point. The methods are able to correctly classify a wide range of traffic gestures, independent of agent orientation and show robustness against the range in execution styles with the leave-one-out cross validation protocol. While there is room for improvement, this works highlights the possibilities of a deep learning approach to the problem of traffic gesture recognition.

6.3.1. Viewpoint variation robustness & transfer learning

As described in Chapter 1, one of the problems encountered in traffic gesture recognition are viewpoint variations from the car with respect to the agent. The main argument behind skeletal features are that they are invariant to appearance and viewpoint changes. The robustness of the methods to this challenge were explored by cross-evaluating the models against the secondary viewpoint present in GPGD. Additionally, transfer learning was explored.

The results in Section 5.3.3 show the advantages of using skeletal features for classification. Classifiers are still able to classify the gestures presented from a different viewpoint, not included in the training data. Additionally, models evaluated in the cross-viewpoint style generally outperform models trained from scratch on this viewpoint. It should be noted that models trained from scratch had less training data to work with. That is, however, a constraint that is very representative of reality.

However, there is a decrease in classification performance noticeable in the evaluation metrics compared to the results of Section 5.3.2. Two reasons are expected to cause this. First, the front-facing viewpoint is a more difficult viewpoint as occlusions are more prevalent and the estimation of depth is harder compared to the more side-facing viewpoint of Section 5.3.2.

Second, given that the models still show improvement when given the ability to finetune their weights to the new dataset, we expect that the classifiers still require some viewpoint specific tuning. Changing factors such as camera height, camera distance and the focal length of the lens used all have impact on the 2D representation of the skeleton and thus the input of the classifier. Using JL_D feature representation aimed to combat this. Comparison performance of DBRNN-L with and without JL_D shows that JL_D feature representation did preserve more classification performance across viewpoints, albeit marginally. This seems to suggest that in contrast to 3D skeletons, 2D skeletons are less invariant to viewpoint variations, which is expected.

The viability of transfer learning has been shown by significantly increasing classification performance. The performance differences between networks trained from scratch are large. Transfer learning has shown here to be a viable method to address the limited gesture data, specifically for viewpoint variations. The best way to retrain a network seems to be mostly dependent on the network architecture. Additional research of the effectiveness of transfer learning to new, unseen traffic gestures is still to be done, but is expected to be possible based on these findings. This would have the implication that once a larger (traffic) gesture dataset has been established, new traffic gestures can be successfully added with limited training data.

6.3.2. Evaluation protocol

Both Berkeley MHAD and GPGD lack a validation set and only consist of training and test data. The evaluation protocol of Berkely MHAD was structured in this way by Ofli et al. [51], so in order to maintain comparability to literature this protocol was also used. For the Greenscreen Police Gesture Dataset [3], cross-actor evaluation was deemed important, as testing on similar gestures as performed in the training data is undesirable. Combined with the small sample and actor count (combined with the poor data of actor 4) made the inclusion of a separate validation set not feasible. A leave-one-out cross-validation protocol was used instead.

The lack of validation set effectively meant that hyperparameter tuning and further optimizations were done using the test set as evaluation point. This means that to some extent the test set is not a pure test set and there is a danger that classifiers have been overfit to the test data and generalize poorer to unseen data. In an optimal scenario, new gesture sequences from at least 3 or more actors are added to the dataset. This would allow a dataset split in the form of, for example, 5 train, 2 validation and 3 test actors.

In the case of Berkeley MHAD, overfitting on the test set is suspected as discussed. Expected causes are the carefully curated nature of the data and actions, combined with the landmark poses present in the actions. For GPGD, given the generalization performance currently shown by the classifiers, the current set-up is not expected to have been problematic. However, it is important to be aware of the constraints of this split. Unfortunately, a better option was not available at this time.

6.4. Generalization to real-world traffic gesture recognition

This works aims to be a starting point towards real-world traffic gesture recognition in the automotive domain. Recordings in the datasets in this work are limited to indoor recording studio's, and are as such not a completely accurate representation of the real-world domain. Several points remain to be addressed before implementation of gesture recognition in real-world scenario's becomes possible.

Difficult lighting conditions and other environmental factors will be the primary challenge for skeletal feature extraction with OpenPose. Detection results in studio settings are of a good quality, however some initial real-world test images, as shown in Figure 6.4 showed mixed performance. Especially long distance detection struggles. Robust skeletal detection is fundamental for accurate classification and warrants further evaluation for applications in the automotive domain. Skeletal features showed to be robust against agent and viewpoint variations, especially when using proper feature representation. The work of Kreiss et al. [40] already explored this specifically and showed promising results.

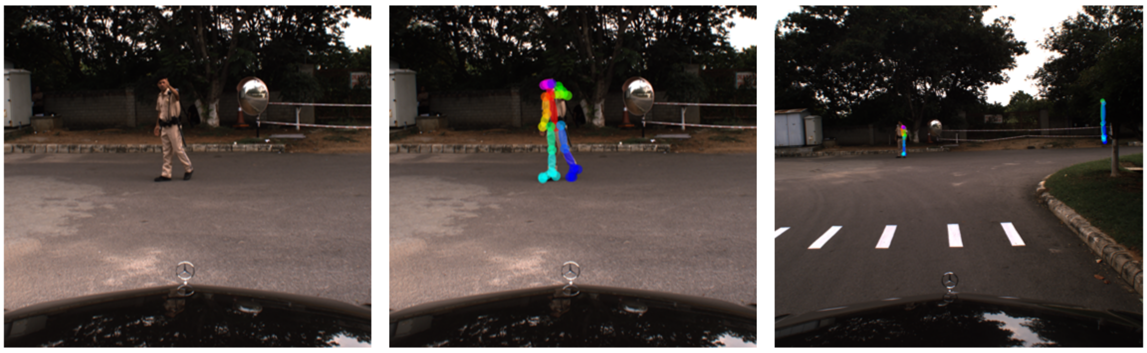


Figure 6.4: Examples of real world skeleton detection using OpenPose. Left image is the input, resulting in the detection of the middle image. Right image shows poor detection and false positives at longer range. Taken from an unlabeled part of GPGD [3]

Current classification performance has been evaluated using data recorded in a studio setting. To properly asses the robustness and generalization of the classifiers, evaluation on real-world traffic gesture recordings have to be done. Based on the findings in this work, we assume that given a robust skeletal detection, the system proposed in this work could generalize well to real-world scenarios. Current classification performance is not yet up to par for implementation in consumer vehicles, but with more work and research, future iterations could achieve the performance needed for successful deployment of gesture recognition for (autonomous) vehicles.

As discussed in Chapter 2, international differences in traffic gesture definitions exist. Therefore it is important that there is a good match between the local gestures used and the gestures the classifier has actually been trained on. Use of transfer learning could help to learn the classifiers new gestures faster and more accurately. Further local research into what gestures are used in practice is important to ensure gestures encountered are known to the classifier.

The results of 5.3.5 showed promising results for the feasibility of real-time implementation. Given the proper hardware and optimization, the deep learning based methods explored in this work should be able to classify traffic gestures in real-time. Interestingly, the time penalty for using significantly larger networks is limited, allowing for use of larger architectures with more performance potential. Examples of potential methods to reduce the computation cost of the model inference are weight pruning, which is based on removing unnecessary connections from the network and explored in the work of Han et al. [25] and weight sharing/quantization, where similar weights are grouped in bins of smaller bit size. The work of Han et al. [24] showed that using such compression methods can result in a 3-4x speed-up and 3-7x reduced energy use with no loss to accuracy on large existing networks such as Alexnet [41] and VGG-16 [60]. Additionally, the classification pipeline does not exist in a vacuum, and pose information can be used by a wide array of subsystems. Integration of the gesture system in the entire automation system can thus result in further optimization. An example would be to use existing pedestrian detections in combination with the pose estimator.

Based on the findings in this work, the pipeline established is expected to generalize well to real-world performance. OpenPose pose estimation combined with skeleton based traffic gesture recognition is a very viable approach with potential for deployment in intelligent vehicles with more development and the right adaptations.

7 Conclusion and further work

Intelligent vehicles continue to increase in automation levels. Self-driving cars, which not very long ago were deemed a futuristic vision, are getting increasingly close to being a reality. However, one of the hurdles to overcome for full automation are interactions with traffic agents. In this process, correct classification of the traffic gestures used for communicating instructions is essential for safe and accurate decision making. This work aimed to provide a working traffic gesture pipeline by leveraging the latest developments in computer vision and machine learning. The classification performance of multiple configurations based on different skeletal feature representations estimated using OpenPose and classified with different architectures based on recurrent neural networks have been evaluated. Each subsystem has been investigated in detail to answer the research questions originally posed in Chapter 1.

OpenPose based gesture classification

The practice of using OpenPose as feature extraction method for skeleton based action recognition was evaluated by comparing its performance against motion capture based action recognition on the Berkeley MHAD dataset. The goal was to investigate how much performance is lost by using visual skeleton estimation and how to minimize this gap. OpenPose based action recognition with coordinate feature representation showed difficulties with addressing spatially overlapping actions where inter-joint relationships are important. By using the joint-line feature representation of Zhang et al. [72], this was addressed resulting in significantly better performance. By achieving an accuracy of 0,9709 it comes very close to the 3D motion capture performance of 0,9855. Additionally, performance of 2D motion capture data was identical to 3D motion capture based classification. These results are promising for the application of OpenPose as 2D skeleton estimation for traffic gesture recognition.

Greenscreen Police Gesture Dataset

A new traffic gesture dataset was presented, containing dynamic and challenging gestures recorded in a greenscreen studio setting called Greenscreen Police Gesture Dataset [3]. This dataset includes challenges relevant to traffic gesture classification such as variations in actor orientation resulting in limb occlusions and variations in gesture execution and gesturing speed. The gestures used by the agents are not exclusive, meaning agents can use two different gestures simultaneously.

A multi-label design was implemented, where the higher level implications of the instructions were labeled, independent of actor orientation or gesture execution. A custom labeling tool was designed and used specifically for this dataset. Finally, the recordings were split by label into train/test samples to achieve a segmented dataset. In addition to the primary viewpoint, recordings were also made from a separate, front facing camera. These recordings were used to create a separate subset, which was used in experiments evaluating cross-viewpoint performance and transfer learning.

Traffic gesture classification performance

The performance of state-of-the-art action recognition methods based on recurrent neural networks was evaluated on the novel Greenscreen Police Gesture Dataset [3] in a cross actor fashion using leave-one-out cross validation. Three RNN network architectures, varying in complexity and size, were evaluated using the primary viewpoint of the dataset. Analysis of this dataset showed consistent trends in the dataset for all configurations. Classification of the dynamic wave gestures proved to be more difficult than the block gestures. Expected causes are the wide range in execution style and the increased complexity of the gesture. These challenges are in-line with the real-world challenges identified back in Chapter 1. Additionally, consistent large intra-actor variations in achieved performance were seen. Possible causes are the different gesture styles used by the actors and in the case of one actor a variation in actor positioning. Overall, the Greenscreen Police Gesture Dataset has proven to be both an effective and challenging starting point for traffic gesture recognition.

The combination of joint-line distance and a large net using GRU-layers showed the best performance. Compared to the other configurations it showed the best rounded performance in its ability to retrieve traffic gestures and produce accurate predictions without too many false positives by achieving a accuracy of 0,49 and mAP of 0,70. The results showcase the power of the combination of skeleton features and RNNs with LSTM/GRU units. The methods are able to classify traffic gestures despite the large execution variations and orientations found in the dataset.

There is, however, still room for improvement. Based on error qualitative analysis there are points to address specifically in future work. Classification of wave gestures proved to be difficult. An expected major cause is the fact that the OpenPose skeleton does not capture wave nuances in the hand. Addition of extra keypoints is expected to improve the ability to classify these gestures significantly. Based on the confusions observed, the networks sometimes struggles with assigning the right direction (N/S/E/W) to the gestures based on the 2D input skeleton. Based on these findings, combined with the results found with Berkeley MHAD, 2D skeletons appear to contain enough information for class identification but struggle with direction identification with the current representation. Continuous evaluation showed that transition points between gestures were difficult to accurately classify. The expected cause lies in the segmentation of gesture recordings during training.

Robustness to viewpoint variation and transfer learning

Insight about the robustness to changes in camera viewpoint was gained by cross-viewpoint evaluation. The previously trained networks were evaluated on a camera viewpoint originally not included in the training data in two ways: No retraining of the networks and transfer learning using samples from the new viewpoint. The were compared against the same architectures trained from scratch on this viewpoint.

Performance of the networks without retraining was very similar to better compared to networks trained from scratch on the alternative viewpoint. Based on this result, the methods show good viewpoint robustness and were able to classify gestures without any retraining. It should be noted that the network trained from scratch had comparably less training data to work with. However a decrease in performance obtained compared to the primary viewpoint was observed. This was partly due to the increased occlusions in this viewpoint, partly due to the viewpoint variation.

The use of transfer learning to the new viewpoint showed substantial improvements. These improvements show that some viewpoint specific optimization was necessary and the classifiers and features are not completely invariant to viewpoint changes. Additionally, it showcases the value of transfer learning for traffic gesture recognition while dealing with small datasets.

Computational efficiency

An estimation of the computation time per frame was made. An estimated computation time of 0.29 seconds per frame (3.44 fps) for the JL_D-GRU configuration including OpenPose keypoint estimation was found on laptop hardware without specific optimization. This result shows that, given the right hardware and optimization, the methodology proposed in this work is very suitable for real-time application in intelligent vehicles.

Future work

The viability to transfer networks to completely new datasets and gestures could be researched using the dataset of He et al. [26] in further work. This direction of research is interesting, because of the implication it has on the sample sizes required to implement new traffic gestures. Efficient transfer learning would allow for quick expansion into new international markets with different traffic gesture definitions. As the Greenscreen Police Gesture Dataset [3] is not without flaws and limited in gesture executions, there is still room for further development of traffic gesture datasets.

Future work to address the shortcomings in classification performance could include the expansion of the detection keypoints with more hand/palm information and research in better feature representation specifically aimed at direction identification. For improving the transition points, an interesting alternative is found in the many-to-many approach to traffic gesture recognition of He et al. [26]. This would have the added benefit of removing the class imbalance from the dataset. Additionally, this would allow for direct comparison to their work on their gesture dataset.

The metrics used in this work have been mainly limited to classification performance. Further work is still to be done on the evaluation and optimization of response time to gestures. Visualization of the action sequences generated using Berkeley MHAD showed an inherent limitation of the current set-up, where training and evaluation is only done based on the final prediction of a sequence. A possible way to address this would be to modify the loss function to penalize slow responses to gestures, such as the work of Koch et al. [38]. Alternatively, a many-to-many approach would circumvent this problem as it optimizes the loss of all frames and predictions in a training sequence, rather than only the final prediction.

The classifiers in this work have been limited to different implementations of recurrent neural networks, such as LSTM and GRU. Evaluation of other configurations of temporal classifiers such as CNN methods or convolutional graph networks could yield further performance gains. The latter has showed impressing results in the work of Shi et al. [58].

Finally, the scope of this work has been limited to the classification of traffic gestures. There are however more aspects to this problem. Determining whether a gesture was performed by a traffic agent with jurisdiction or just a random pedestrian waving on the street, or the integration of the traffic gestures in the further decision making progress in traffic are both examples of venues that remain unresolved in this work. Additionally, there are more aspects besides classification in the interaction with traffic agents. Clear communication to the agent by the vehicle what decision or interpretation was made based on the gesture is also important, which could be done using external human-machine interfaces. Depending on the automation level and if the driver is still in the loop, communication with the driver is also important regarding the gesture classification.

Conclusion

Traffic gesture recognition is a complicated problem that still has to be solved in before full automation in intelligent vehicles can be achieved and has received limited attention in literature to date. This work is one of the first explorations of an end-to-end traffic gesture classification pipeline based on deep learning methods. Several classification configurations using skeletal features estimated from image data with OpenPose and classified with RNN architectures were presented and evaluated in detail. The final results are promising and show the power and validity of the approach chosen in this work. Hopefully, the work presented in this thesis can function as foundation for further work and bring humanity just a tiny step closer in the process of reducing the 1.35 million yearly traffic deaths to zero.

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

[2] National Highway Traffic Safety Administration. Critical reasons for crashes investigated in the national motor vehicle crash causation survey. Technical report, U.S. Department of Transportation, 2015.

[3] Mercedes Benz AG. Greenscreen Police Gesture Dataset, January 2020. Internal research usage granted to TU Delft, Intelligent Vehicles Group.

[4] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2D human pose estimation: New benchmark and state of the art analysis. In *Proceedings of the 2014 IEEE Conference On Computer Vision And Pattern Recognition*, pages 3686–3693, 2014.

[5] Federico Angelini, Zeyu Fu, Yang Long, Ling Shao, and Syed Mohsen Naqvi. ActionX-Pose: A novel 2D multi-view pose-based algorithm for real-time human action recognition. *arXiv preprint arXiv:1810.12126*, 2018.

[6] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[7] Aaron Bobick and James Davis. The recognition of human movement using temporal templates. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (3):257–267, 2001.

[8] Zixing Cai and Fan Guo. Max-covering scheme for gesture recognition of chinese traffic police. *Pattern Analysis and Applications*, 18(2):403–418, 2015.

[9] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2D pose estimation using part affinity fields. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition*, pages 7291–7299, 2017.

[10] Zhe Cao, Gines Hidalgo Martinez, Tomas Simon, Shih-En Wei, and Yaser A Sheikh. Openpose: Realtime multi-person 2D pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[11] Rizwan Chaudhry, Ferda Ofli, Gregorij Kurillo, Ruzena Bajcsy, and Rene Vidal. Bio-inspired dynamic 3D discriminative skeletal features for human action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 471–478, 2013.

[12] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[13] François Chollet et al. Keras, 2015. URL `https://keras.io`.

[14] Qi Dang, Jianqin Yin, Bin Wang, and Wenqing Zheng. Deep learning based 2D human pose estimation: A survey. *Tsinghua Science and Technology*, 24(6):663–676, 2019.

[15] New York City Police Department. An unexpected rain storm doesn't stop an NYPD traffic enforcement agent from directing traffic. URL `https://www1.nyc.gov/site/nypd/media/photo-gallery/photo-gallery.page?galleryScript=082016_tea`. Accessed: 2020-5-1.

[16] Yong Du, Wei Wang, and Liang Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *2015 IEEE Conference on Computer Vision and Pattern recognition*, pages 1110–1118, 2015.

[17] Zhijie Fang, David Vázquez, and Antonio López. On-board detection of pedestrian intentions. *Sensors*, 17(10):2193, 2017.

[18] Omair Ghori, Radek Mackowiak, Miguel Bautista, Niklas Beuter, Lucas Drumond, Ferran Diego, and Björn Ommer. Learning to forecast pedestrian intention from pose dynamics. In *Proceedings of the 2018 IEEE Intelligent Vehicles Symposium*, pages 1277–1284. IEEE, 2018.

[19] Alex Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*. Studies in Computational Intelligence. Springer, Berlin, 2012. doi: 10.1007/978-3-642-24797-2.

[20] Fan Guo, Zixing Cai, and Jin Tang. Chinese traffic police gesture recognition in complex scene. In *Proceedings of the 2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 1505–1511. IEEE, 2011.

[21] Surabhi Gupta, Maria Vasardani, and Stephan Winter. Conventionalized gestures for the interaction of people in traffic with autonomous vehicles. In *Proceedings of the 9th ACM SIGSPATIAL International Workshop on Computational Transportation Science*, pages 55–60. ACM, 2016.

[22] Isabelle Guyon, Vassilis Athitsos, Pat Jangyodsuk, Ben Hamner, and Hugo Jair Escalante. Chalearn gesture challenge: Design and first results. In *2012 IEEE Computer*

Society Conference on Computer Vision and Pattern Recognition Workshops, pages 1–6. IEEE, 2012.

[23] Richard W Hamming. Error detecting and error correcting codes. *The Bell system technical journal*, 29(2):147–160, 1950.

[24] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[25] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing systems*, pages 1135–1143, 2015.

[26] Jian He, Cheng Zhang, Xinlin He, and Ruihai Dong. Visual recognition of traffic police gestures with convolutional pose machine and handcrafted features. *Neurocomputing*, 390:248–259, May 2020.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[28] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[29] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[30] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[31] PhaseSpace Inc. *Impulse optical motion capture system*. San Leandro, CA USA, . Specification sheet available at `http://www.phasespace.com/impulse-motion-capture.html`.

[32] Point Grey Research Inc. *Dragonfly2 cameras*. Richmond, BC Canada, . Specification sheet available at `http://www.lustervision.com/UpLoadFile/20141214/dragonfly2-datasheet-GSM.pdf`.

[33] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[34] Gunnar Johansson. Visual perception of biological motion and a model for its analysis. *Perception & psychophysics*, 14(2):201–211, 1973.

[35] Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks, May 2015. URL `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`. Accessed: 2020-3-20.

[36] Tae Soo Kim and Austin Reiter. Interpretable 3D human action analysis with temporal convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1623–1631. IEEE, 2017.

[37] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[38] Philipp Koch, Huy Phan, Marco Maass, Fabrice Katzberg, Radoslaw Mazur, and Alfred Mertins. Recurrent neural networks with weighting loss for early prediction of hand movements. In *2018 26th European Signal Processing Conference*, pages 1152–1156. IEEE, 2018.

[39] Julian F P Kooij, Nicolas Schneider, Fabian Flohr, and Dariu M Gavrila. Context-based pedestrian path prediction. In *European Conference on Computer Vision*, pages 618–633. Springer, 2014.

[40] Sven Kreiss, Lorenzo Bertoni, and Alexandre Alahi. PifPaf: Composite fields for human pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 11977–11986, 2019.

[41] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances In Neural Information Processing Systems*, pages 1097–1105, 2012.

[42] Ivan Laptev. On space-time interest points. *International Journal of Computer Vision*, 64 (2-3):107–123, 2005.

[43] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[44] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988, 2017.

[45] Jun Liu, Amir Shahroudy, Mauricio Lisboa Perez, Gang Wang, Ling-Yu Duan, and Alex Kot Chichung. NTU RGB+ D 120: A large-scale benchmark for 3D human activity understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1010–1019, 2019.

[46] Mehran Maghoumi and Joseph J LaViola Jr. DeepGRU: Deep gesture recognition utility. In *International Symposium on Visual Computing*, pages 16–31. Springer, 2019.

[47] Julieta Martinez, Rayat Hossain, Javier Romero, and James J Little. A simple yet effective baseline for 3D human pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2640–2649, 2017.

[48] Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. Vnect: Real-time 3D human pose estimation with a single RGB camera. *ACM Transactions on Graphics*, 36(4):44, 2017.

[49] News-Times. Dancing traffic guard spreads the joy, 2014. URL `https://www.newstimes.com/local/article/Dancing-traffic-guard-spreads-the-joy-5754319.php#photo-6842002`. Accessed: 2020-5-1.

[50] Štěpán Obdržálek, Gregorij Kurillo, Ferda Ofli, Ruzena Bajcsy, Edmund Seto, Holly Jimison, and Michael Pavel. Accuracy and robustness of Kinect pose estimation in the context of coaching of elderly population. In *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 1188–1193. IEEE, 2012.

[51] Ferda Ofli, Rizwan Chaudhry, Gregorij Kurillo, René Vidal, and Ruzena Bajcsy. Berkeley MHAD: A comprehensive multimodal human action database. In *2013 IEEE Workshop on Applications of Computer Vision*, pages 53–60. IEEE, 2013.

[52] Christopher Olah. Understanding lstm networks. `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`, August 2015. Accessed: 2020-3-20.

[53] Georgios Pavlakos, Xiaowei Zhou, Konstantinos G Derpanis, and Kostas Daniilidis. Coarse-to-fine volumetric prediction for single-image 3D human pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7025–7034, 2017.

[54] Markus Roth, Fabian Flohr, and Dariu M Gavrila. Driver and pedestrian awareness-based collision risk analysis. In *Proceedings of the 2016 IEEE Intelligent Vehicles Symposium*, pages 454–459. IEEE, 2016.

[55] Abraham Savitzky and Marcel JE Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, 36(8):1627–1639, 1964.

[56] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[57] Science and Technology Select Committee. Connected and autonomous vehicles: The future? Technical report, House of Lords, London, 2017.

[58] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12026–12035, 2019.

[59] Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *Proceedings Of The IEEE Conference On Computer Vision And Pattern Recognition*, pages 1145–1153, 2017.

[60] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Conference Track Proceedings of the 3rd International Conference on Learning Representations*, 2015.

[61] World Health Organization Violence and injury prevention team. Global status report on road safety 2018. Technical report, World Health Organization, Geneva, 2018.

[62] Ben Wang and Tao Yuan. Traffic police gesture recognition using accelerometer. In *IEEE SENSORS Conference, Lecce-Italy*, pages 1080–1083, 2008.

[63] Pichao Wang, Wanqing Li, Philip Ogunbona, Jun Wan, and Sergio Escalera. RGB-D-based human motion recognition with deep learning: A survey. *Computer Vision and Image Understanding*, 171:118–139, 2018.

[64] Caleb Williams. A minneapolis police department traffic officer directing traffic in down-town minneapolis, June 2009. URL `https://commons.wikimedia.org/wiki/File:090610-Minneapolis-Traffic-Officer-2.jpg`. Accessed: 2020-5-1.

[65] Lu Xia, Chia-Chih Chen, and Jake K Aggarwal. View invariant human action recognition using histograms of 3D joints. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 20–27. IEEE, 2012.

[66] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Thirty-second AAAI Conference on Artificial Intelligence*, 2018.

[67] Zhengyuan Yang, Yuncheng Li, Jianchao Yang, and Jiebo Luo. Action recognition with spatio–temporal visual attention on skeleton image sequences. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(8):2405–2415, 2018.

[68] Angela Yao, Juergen Gall, Gabriele Fanelli, and Luc Van Gool. Does human action recognition benefit from pose estimation? In *Proceedings of the 2011 British Machine Vision Conference 2011*, 2011.

[69] Alper Yilmaz and Mubarak Shah. Actions sketch: A novel action representation. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 984–989. IEEE, 2005.

[70] Tao Yuan and Ben Wang. Accelerometer-based chinese traffic police gesture recognition system. *Chinese Journal Of Electronics*, 19(2):270–274, 2010.

[71] Kiwon Yun, Jean Honorio, Debaleena Chattopadhyay, Tamara L Berg, and Dimitris Samaras. Two-person interaction detection using body-pose features and multiple instance learning. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 28–35. IEEE, 2012.

[72] Songyang Zhang, Xiaoming Liu, and Jun Xiao. On geometric features for skeleton-based action recognition using multilayer LSTM networks. In *2017 IEEE Winter Conference on Applications of Computer Vision*, pages 148–157. IEEE, 2017.

[73] Wentao Zhu, Cuiling Lan, Junliang Xing, Wenjun Zeng, Yanghao Li, Li Shen, and Xiaohui Xie. Co-occurrence feature learning for skeleton based action recognition using regularized deep LSTM networks. In *Thirtieth AAAI Conference On Artificial Intelligence*, 2016.

[74] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[75] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8697–8710, 2018.